IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 21 June 2023, accepted 23 July 2023, date of publication 1 August 2023, date of current version 25 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3300376

SURVEY

Dependable DNN Accelerator for Safety-Critical Systems: A Review on the Aging Perspective

IRAJ MOGHADDASI, SAEID GORGIN, (Senior Member, IEEE), AND JEONG-A LEE, (Senior Member, IEEE)

Department of Computer Engineering, Chosun University, Gwangju 61452, South Korea

Corresponding authors: Iraj Moghaddasi (moghaddasi@chosun.ac.kr) and Jeong-A Lee (jalee@chosun.ac.kr)

This work was supported in part by the Brain Pool Program funded by the Ministry of Science and Information and Communication Technology (ICT) through the National Research Foundation of Korea under Grant NRF-2021H1D3A2A02040040, and in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant NRF-2020R1I1A3063857.

ABSTRACT In the modern era, artificial intelligence (AI) and deep learning (DL) seamlessly integrate into various spheres of our daily lives. These cutting-edge disciplines have given rise to numerous safety-critical applications such as autonomous driving with a paramount concern on ensuring a high promise of dependability because of the high risk of human injury in the case of malfunction. Even the dependability becomes more crucial as shrinking CMOS technology feature size enhances resilience concerns due to factors like aging. In the context of DL accelerators, which heavily rely on the efficiency and speed of computations, addressing the effects of aging is of utmost significance to ensure their optimal design and performance. This paper addresses the overarching dependability issue of advanced deep neural networks (DNN) accelerators from the aging perspective. Especially, a comprehensive survey and taxonomy of techniques used to evaluate and mitigate aging effects are introduced. We cover different aging effects like permanent faults, timing errors, and lifetime issues. We review research by the layer-wise approach and categorize several resilience classes to bring out major features. The concluding part of this review highlights the questions answered and several future research directions. This study is expected to benefit researchers in different areas of DNN deployment, especially the dependability of this emergent paradigm.

INDEX TERMS Deep learning, AI accelerator, dependability management, resilience, safety-critical systems, fault tolerance, aging, lifetime, permanent error, timing error, graceful degradable system, NBTI.

I. INTRODUCTION

Artificial intelligence (AI) and machine learning (ML) are becoming pervasive in today's applications with intelligent autonomy. The deep neural networks (DNNs) deployment in these applications has prohibitive requirements of throughput, latency, cost, and power consumption, while a significant number of these applications are positioned in resource-constrained embedded domains. Herein, the DNN processing unit may be an ordinary commercial off-the-shelf embedded edge device, including limited power and computing resources. Accordingly, the ever-increasing complexity of emergent DNN models and algorithms poses challenges in deploying them on edge devices that struggle to support the complex models because of increasing computing demand and the number of parameters. Thus, edge devices need support from efficient special-purpose accelerators to process DNNs which are gaining popularity due to their broad applications, excellent performance, and energy efficiency.

On the other hand, DNNs are increasingly employed in safety-critical disciplines like healthcare [1], automotive [2], avionics [3], space [4], railway [31], and industry [6] to extract useful information from complex raw data. Hence, the dependability of deployed accelerator-based systems is rising in importance. Their failure can result in catastrophic consequences such as the loss of lives or severe environmental damage [7]. Moreover, the continuous scaling of CMOS technology feature size significantly threatens the dependability of emergent DNN accelerators due to factors like aging. In other words, aging-induced permanent stuck-at faults

The associate editor coordinating the review of this manuscript and approving it for publication was Jinhua Sheng.

(PSFs) and timing errors (TEs) dominate the dependability of advanced AI chips under the nanoscale manufacturing process.

The importance of considering aging in DNN accelerators lies in the following aspects:

- **Reliability and Lifetime**: Ensuring the reliability and longevity of these accelerators is essential to avoid unexpected failures that can result in costly downtime and data loss. Aging mechanisms must be analyzed to estimate the hardware's lifetime and improve reliability.
- **Performance Degradation**: Over time, digital circuits can degrade due to various aging mechanisms e.g., bias temperature instability (BTI). These aging effects can decrease the performance of DNN accelerators, affecting the efficiency of training and inference tasks.
- **Energy Efficiency**: As aging impact the performance of accelerators; it can result in higher power consumption to maintain the same level of performance. Understanding and mitigating aging effects can help maintain energy efficiency over the lifetime.
- **Thermal Management**: Aging can exacerbate thermal issues in DNN accelerators. Higher temperatures due to accelerated aging can lead to thermal throttling, reducing the overall performance and causing stability problems.
- **Workload Balancing**: Over time, different parts of the DNN accelerator may age differently, leading to performance variations across different hardware parts. Being aware of this difference can help developers distribute the workload effectively and maximize performance.
- **Economic Implications**: Designing DNN accelerators to account for aging can have economic benefits by extending the lifetime. It can reduce the need for frequent hardware replacements.
- **Design Optimization**: Considering aging in the design of DNN accelerators allows hardware engineers to incorporate techniques like error correction codes or adaptive resource allocation. These approaches can help mitigate the impact of aging and prolong the lifetime.

To address these issues, researchers work on developing aging-aware DNN accelerator architectures, robust algorithms, and adaptive techniques that can effectively manage aging effects and ensure the continued efficiency and reliability of hardware accelerators over time. The primary focus of the prior research works have been on optimizing DNN algorithms and architecture to improve performance and efficiency based on compression and approximation [8], [9]. However, these methods may lead to reduced resilience due to intrinsic redundancy reduction. In emergent intelligent edge devices utilized in safety-critical domains, dependability takes precedence as a first-class citizen. Thereby, it is essential to highlight that dependability requires special attention, as it has received comparatively less investigation in this particular discipline. This paper provides a survey of design-time and runtime considerations, techniques, and tools presented to improve the dependability of DNN accelerators from the aging perspective.

A. CONTRIBUTIONS AND PAPER OUTLINE

Compared to related survey studies, the major contributions of this paper are as follows:

1) The first survey of aging studies in DNN accelerators.
2) Explores dependability techniques from different scopes related to aging concerns.
3) Considers safety-critical standards in review.
4) Introduces functional resilience, timing resilience, and lifetime resilience for the first time
5) Proposes a new classification of timing resilience according to countermeasures conducted in the literature.
6) Explores lifetime resilience considering existing models.
7) Highlights current challenges and future research ideas in DNN accelerators from the aging perspective.

The overall outline of the paper, as illustrated in Fig 1, is organized as follows. We present preliminaries for this work in Section II. Section III overviews safety standards and resilience importance in aging presence. In Section IV, we study research conducted to enhance functional resilience. Section V presents state-of-the-art research on timing resilience improvement. Techniques for lifetime extension are surveyed in Section VI. Finally, Section VII concludes the paper. The proposed review is expected to improve the existing surveys, especially considering aging factors through a hierarchical and complete background study.

B. METHODOLOGY

The goal of this article is to answer the following questions about DNN accelerators in aging presence:

1) Which dependability counterparts are essential for aging?
2) Which safety standards are about the DNN accelerator?
3) What are the effects of aging on the DNN accelerator?
4) What are the main techniques for dealing with the PSFs?
5) What are the major methods to mitigate TE effects?
6) What are the main approaches for extending the lifetime?
7) What are the differences in dealing with aging-induced permanent faults in DNN deployment layers?
8) What are the scopes of dealing with aging-induced TEs?
9) What is the efficacy of optimization on the resilience of the aging-aware DNN accelerator?

This survey studies empirical research works issued as papers, written in English, and peer-reviewed mainly during the last few years. Database search and article identification is conducted to collect related articles from popular electronic databases and libraries like Springer, IEEE, ScienceDirect, ACM, etc. Table 1 shows the keywords used to extract doc-

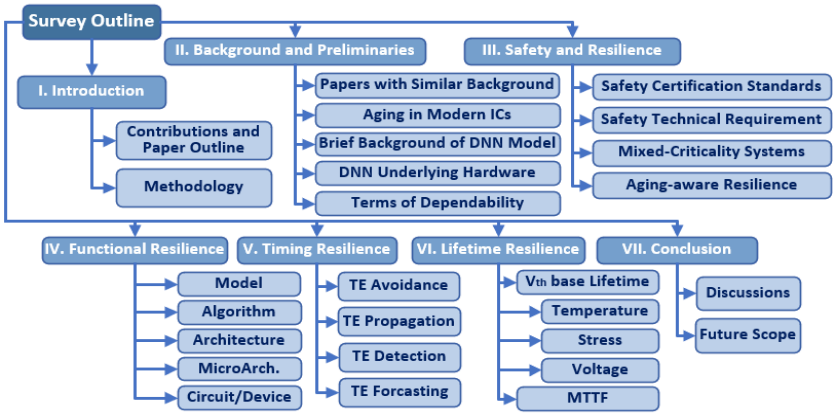

FIGURE 1. Paper outline.

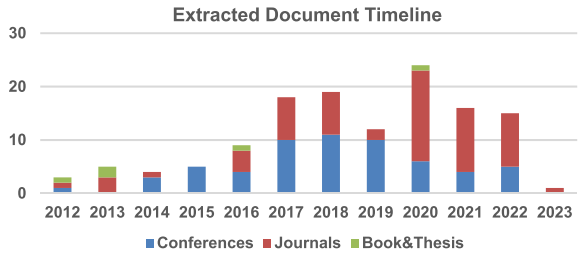

FIGURE 2. Selected paper timetable.

uments classified by the research questions. The collected documents' timetable and count are shown in Fig 2 including journals, conferences, and others, e.g., books, standards, and online documents. The papers mainly published from 2012 to 2023 are considered to analyze the recent trends. There has been a major increase in publications since 2016 due to the increasing adaptation of DNNs in different AI edge applications. A total of about 150 documents are extracted according to research questions and are utilized to provide this review. It should be noted that some of the referenced articles are from before 2012, which include two parts. The first part concerns defining basic concepts and tools like dependability, safety, timing error sensor, etc. The second part is related to papers previously proposed on SoCs (system on chip) as underlying hardware of NN. We present them to draw a comprehensive picture of techniques, tools, and research opportunities with the ability to be updated for advanced DNN accelerators.

II. BACKGROUND AND PRELIMINARIES

A. PAPERS WITH SIMILAR BACKGROUND

This subsection overviews the recent related survey papers with scopes ranging from the DNN accelerator reliability and resilience to hardware design, optimization, and safety concerns. In the past, exploring neural network (NN) resilience against dependability threats has been of some research interest. In [10], the author studied the methods used to

TABLE 1. Keywords used for the extraction of relevant articles.

Research Question	Keyword Searched
1	DNN dependability, fault tolerance, resilience, reliability, permanent faults, aging, timing error, dependability means, and measure
2	DNN safety-critical system standard, safety certificate, mission-critical, safety-critical application, resilience
3	Aging-aware DNN accelerator, resilience, reliability, efficiency, sensitivity analysis,
4	DNN accelerator, permanent fault, fault tolerance, stuck-at error, TDDB, EM, soft error, mapping, pruning, reconfiguration, training,
5	DNN accelerator timing error, NBTI, HCI, NTC, voltage, training, prediction, detection, ABFT, frequency speculation, STA, DTA
6	DNN accelerator lifetime, graceful degradation, aging model, thermal management, stress, voltage adaptation, DRM, DTM,
7	DNN accelerator, hierarchical resilience, layer-wise, cross-layer
8	DNN accelerator optimization, efficiency, implementation, precision adjustment, DNN compression, quantization, approximation,
9	IMC, floor planning, DNN accelerator, thermal management, sensor assignment

evaluate and optimize DNN accelerators' reliability. It covered resilience threats related to transient, permanent, and TEs induced by process variation, memory refresh rate scaling, voltage scaling, and thermal effects. The study of [11] proposes a survey on fault-tolerant deep learning with a hierarchical approach studying proposed methods from the model layer, architecture layer, circuit layer, and cross-layer views. Moreover, [12] reviews the reliability issues of DNN accelerators considering soft errors. It first studies the inherent fault tolerance of DNN models and then classifies proposed methods for improving the reliability considering fault type and passive and active fault tolerance. It analyzed different DNN accelerators and compared the commonly used accelerators to find a more robust one against transient faults. Another similar work was conducted by [13] on fault-tolerant NNs focusing on well-established passive techniques to improve reliability using design-time techniques for feedforward NNs. It reviews fault types and provides a taxonomy of the techniques and measures to improve the intrinsic fault tolerance of the NN model, and finally, it reviews some works on active fault tolerance in NNs. Resilience challenges and

TABLE 2. Scopes of similar papers.

Reference	DNN Accelerator	Aging Effects	Resilience	Safety Criticality	Stuck-at Fault	Timing Error	Lifetime	Approximation	Layer-wise	Dependability Means	Graceful Degradation	Thermal Effects	Stress Effects	V_{dd} Effects
[10]	✓		✓		✓	✓		✓				✓		✓
[11]	✓		✓		✓	✓		✓	✓					✓
[12]	✓		✓											
[13]	✓		✓		✓					✓	✓			
[14]	✓		✓			✓						✓		✓
[15]	✓	✓	✓		✓	✓		✓						
[16]				✓										
[17]	✓													
[18]	✓							✓						
[19]	✓							✓						
[20]	✓							✓						
Our Survey	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

opportunities for the operation of DNN accelerators in NTC (near-threshold computing) are studied in [14]. In the case of the TPU systolic array, the paper discusses the intensity of TEs and their effect on inference accuracy at NTC. It evaluates factors such as delay difference within arithmetic units, utilization pattern, hardware homogeneity, and workload characteristics and uncovers local and global techniques to deal with the NTC-induced TEs. Another recent review indicates that DNNs have intrinsic reliability threats and provided an overview of reliability threats and corresponding mitigation techniques [15].

The article [16] reviews research contributions to address random hardware failures, systematic failures, and execution independence in GPU devices. It discusses several challenges of safety certification in complex, parallel, and compute-intensive software functions with different safety-criticality levels on shared GPU devices. Recent advances in DNN accelerator designs are presented in [17]. It explored various proposed architectures in terms of dataflow optimization, internal communication network, deployment of emerging technologies, and computing units for advanced DNN applications. On the other hand, a comprehensive overview of existing optimization techniques to deploy DNN accelerator on FPGA is provided in [18]. Moreover, [19] reviews prior efforts to deploy DNN models on the end devices efficiently. The design ideas include the types of DNN models, hardware and software requirements for the development, resource constraints imposed by the computing devices, and optimization techniques. It reviews the following four dimensions: (1) The DNN model, (2) Hardware requirement, (3) Resources utilization, and (4) Application perspective. Finally, [20] explains the different DNN optimization techniques. It reviewed recent research aiming at the implementation of DNN models on FPGAs.

The features and scopes of prior studies are compared with the contributions of our paper in Table 2. Additionally, Table 3 gives a summary of these articles. The subsequent section defines the terms and concepts used in this paper.

B. AGING IN MODERN INTEGRATED CIRCUITS

Aging refers to the degradation due to continued electrical stress on CMOS devices that may eventually lead to PSF or TE occurrence and operational lifetime reduction. Aging is affected by several factors, like temperature, voltage, and stress. Shrinking CMOS feature size increases power density, peak temperature, and electrical fields inside modern integrated circuits, leading to more severe aging [21]. Overall, there are mainly four different aging mechanisms:

1) BIAS TEMPERATURE INSTABILITY (BTI)

BTI relates to the transistor's threshold voltage (V_{th}) increase and the carrier mobility (μ) decrease [22]. This causes to increase in circuit delay, which eventually leads to timing faults. Depending on the transistor type, there are two types of BTI (e.g., PBTI in nMOS transistors and NBTI for pMOS transistors). Two main models proposed for BTI include reaction-diffusion (RD) and trapping/de-trapping (TD) [23]. Both models describe BTI based on a two-phase process, including stress and recovery. Applied voltage in the stress phase causes the degradation, while in the next phase, the accumulated degradation partially recovers.

2) HOT CARRIER INJECTION (HCI)

Similar to BTI, HCI is manifested as increasing V_{th} and decreasing carrier mobility. The trapping process occurs when charge carriers in the channel gain sufficient velocity to be injected into the dielectric. HCI degradation depends on voltage, temperature, and transition density. HCI has no recovery phase [24].

3) TIME-DEPENDENT DIELECTRIC BREAKDOWN (TDDB)

Unlike BTI and HCI with gradual degradation, TDDB occurs as a sudden stuck-at-fault and the breakdown of a single transistor. TDDB is caused by the buildup of gate oxide defects due to a strong electric field, eventually producing a conducting path between the gate and the channel [24].

4) ELECTROMIGRATION (EM)

EM is the dominant aging mechanism in metal layers. It is characterized by an increase in wire resistance over time due to the drifting of atoms in the same direction as charge carriers. This creates atomic lattice distortions, reduces conductivity, and potentially forms atomic voids. As a result, conductors become permanently damaged, eventually, cause to open circuit failure [22].

Considering the past research done in Samsung and TSMC, NBTI is the dominant aging factor in recent CMOS technologies [25], [26]. Other research in Intel claims that NBTI is the biggest contributing mechanism but not dominant [26]. Another work has claimed that different mechanisms are dominated by voltage and temperature operating points [27]. According to the bathtub curve (Fig. 3), semiconductor product lifetime includes three primary phases: 1) Early life failure rate (or infant mortality) is characterized by a rel-

TABLE 3. Summaries of related review papers.

Ref.	Review Title	Summary
[10]	A survey on modeling and improving the reliability of DNN algorithms and accelerators	This article presents a survey on optimizing techniques for the reliability of DNN accelerators and architectures. This study covered soft/hard errors arising due to process variation, voltage scaling, TEs, DRAM errors, and thermal effects.
[11]	Special Session - Fault-Tolerant Deep Learning: A Hierarchical Perspective	It surveys fault-tolerant deep learning design approaches with a hierarchical perspective and considered them from themodel layer, architecture layer, circuit layer, and cross-layer, respectively.
[12]	Soft errors in DNN accelerators A comprehensive review	This article reviews the reliability of the DNN accelerators considering soft error. The study begins by reviewing the inherent fault tolerance. The DNN accelerators are classified based on several criteria (fault type, passive or active FT). Each is individually analyzed, and the commonly used accelerators are compared to answer the question, which accelerator is more reliable against transient faults?
[13]	Fault and Error Tolerance in Neural Networks A Review	This paper studies fault tolerance in neural networks focusing on well-established passive techniques to improve by design for feed-forward neural networks. It reviews fault types and measures used to evaluate performance and provides a taxonomy of the techniques to improve the intrinsic properties of neural models. Finally, it reviews representative works on active fault tolerance in neural networks.
[14]	Challenges and Opportunities in Near-Threshold DNN Accelerators around TEs	This paper reviews DNN architecture to determine challenges and opportunities for operation in NTC. By simulation in TPU systolic array, the paper discovered the intensity of TEs and its effect on inference accuracy at NTC. It evaluates factors such as data delay relationship, delay disparity within arithmetic units, utilization pattern, hardware homogeneity, and workload characteristics and uncovers local and global techniques to deal with the TEs in NTC.
[15]	Robust Machine Learning Systems: Challenges, Current Trends, Perspectives, and the Road Ahead	This article provides an overview of the vulnerabilities of modern ML systems. It surveys state-of-the-art defense mechanisms against reliability threats, describes different challenges and solutions, and identifies potential promising avenues to research.
[16]	GPU Devices for Safety-Critical Systems A Survey	The deployment of complex, parallel, and compute-intensive software functions with different safety-criticality levels on GPU devices with shared hardware resources introduces several challenges in safety certification. This survey classifies and provides a review of research contributions that address random hardware failures, systematic failures, and execution independence in GPU devices.
[17]	A Survey of Accelerator Architectures for Deep Neural Networks	This paper studies recent advances in accelerator designs for deep neural networks. It explores various architectures that support DNN executions in terms of computing units, dataflow optimization, targeted network topologies, architectures on emerging technologies, and accelerators for emerging applications.
[18]	Optimizing Neural Networks for Efficient FPGA Implementation A Survey	This paper reviews the existing optimization techniques and analyzes them to provide a comprehensive overview of FPGA-based DNN accelerators.
[19]	Design possibilities and challenges of DNN models: a review on the perspective of end devices	This paper reviews efforts for the design of DNN models to implement at the edge devices such as smart cameras. The design ideas include the types of DNN models, hardware and software requirements, resource constraints imposed by the computing devices, and optimization techniques. It reviewed the following four dimensions: (1) DNN model perspective, (2) Hardware perspective, (3) Resources optimization perspective, and (4) Application perspective
[20]	Accelerating Deep Neural Networks implementation, A survey	First, this paper explains different DNN optimization techniques presented in recent research works. Next, it reviewed research works aiming to accelerate the implementation of DNN models on FPGAs.

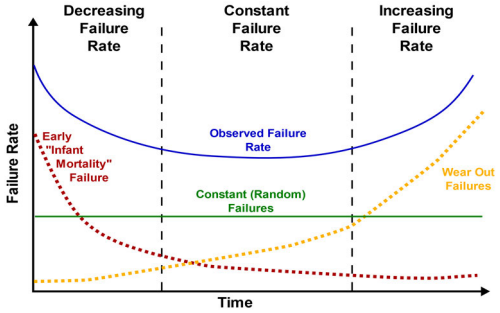

FIGURE 3. The bathtub curve shows the failure rate evolution vs. time.

TABLE 4. Different semiconductor market constraints (modified from [24]).

Specification	Consumer	Automotive	Aerospace and Military	Medical
Temperature Range	0 to 38° C	-40 to 150° C	-55 to 125° C	0 to 70° C
Expected Lifetime	3-5 years	15 years	30 years	7-10 years

atively higher initial failure rate, which decreases rapidly. 2) Normal life consists of a relatively constant failure rate, which remains stable over the useful lifetime of the device. 3) The wear-out phase represents the point at which intrinsic wear-out mechanisms begin to dominate, and the failure rate increases exponentially. The product lifetime is typically defined as the time from initial production until wear-out onset. Table 4 summarizes the lifetime and temperature specifications required in some key markets [24].

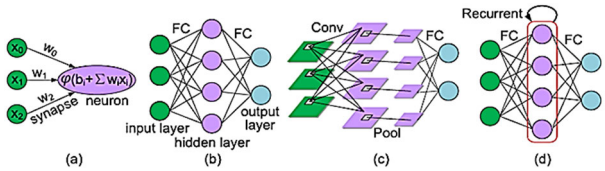

FIGURE 4. Neural network structures. (a) Single neuron. (b) MLP. (c) CNN. (d) RNN (modified from [163]).

C. BRIEF BACKGROUND OF DNN MODELS

In this section, we briefly explain the preliminaries of DNN and review major models. A neural network is composed of artificial neurons; each performs a weighted sum and a nonlinear transform on the result to produce an output that is multi-casted to the next layer. We name the connection as a synapse, the connection influence as weight (W), the input signal as activation (X), and the nonlinear transform as activation function (ϕ). So, the operation of each neuron is as follows:

$$y_i = \varphi\left(\sum_j x_j w_{ij} + b_i\right) \tag{1}$$

where b_i is a bias, as shown in Fig. 4(a). Usually, DNN has many hidden layers, so-called deep. The most popular types, according to the network structures, are as follows [28], [29]:

1) Multilayer perceptron (MLP) - Each node is composed of a nonlinear function of a weighted sum of all the

previous layer outputs (fully connected) [30], as shown in Fig. 4(b).
2) Convolutional neural network (CNN) - convolution, pooling, fully connected, and non-linear function layers are sequentially stacked together, as shown in Fig. 4(c). CNNs are powerful for feature extraction.
3) Recurrent neural network (RNN) -Each node is composed of a nonlinear function of the weighted sum of the previous layer outputs and the previous state of that neuron. RNNs are capable of learning sequential temporal dynamics in applications like speech recognition and language processing, which is difficult for MLPs and CNNs, as illustrated in Fig. 4(d).

DNNs are deployed in two distinct phases: *training* and *inference.* At first, the network tries to adjust the parameters (e.g., weights and biases) using a labeled dataset, including forward and backward propagation, to update the weight gradient. Next, an already trained NN makes predictions for new unseen data involving the forward pass. Training determines the model's parameters, while inference uses them to make predictions. In contrast to the training procedure, which is typically executed on distributed systems, the inference is usually executed on a single device [29] (e.g., a mobile, edge, and IoT device) where latency [55], as well as energy constraints, may become more serious. So, we require huge inference computation in a large DNN, which leads to higher latency and energy consumption. Hence, against training, inference has rarely been implemented directly, and several optimizations are used to meet real platform limitations.

DNN comprises multiple types of layers, including convolutional, pooling, normalization, and fully-connected sublayers [30]. In each layer, there are several channels (i.e., feature maps) extracting different local features of the input data. The convolutional layer performs the main portion of network computation. It performs a dot product between two matrices, namely the input feature map and weight set known as the kernel. Pooling is used to reduce the activation number of the next layer and thus reduce the memory and computation required. Based on the fact that neural networks usually have a normal distribution, the normalization layer keeps output values in the same input range (e.g., Batch Normalization [31]). It simplifies and speeds up the training process to reach higher learning rates and helps to interfere with lower cost and more resilience [32]. The activation functions applied to the output of neurons improve the usability of NNs by solving complex problems that cannot be solved with linear algebra. Usually, $ReLU\,(x) = max\,(x, 0)$ is used as the activation function ϕ in recent networks.

D. DNN ACCELERATOR UNDERLYING HARDWARE

Since the beginning of the last decade, the deployment of DNNs has not been limited to heavy computing systems, but there is much interest in deploying DNNs in resource and energy-limited edge devices. The underlying hardware ranges from *temporal* to *spatial* architectures with the main element of multiply-and-accumulate (MAC). In the following, we review the appropriate platforms for DNN deployment.

Typically, hardware acceleration can be implemented by CPU [33], GPU [34], FPGA [35], or ASIC [36] to improve performance and energy efficiency. CPU and GPU are based on temporal architecture, while spatial architecture has been used in ASIC and FPGA-based platforms. Although the similarities in the parallel computation of processing units, temporal and spatial architecture have differences in elements like control units, memory structures, and dataflows. The temporal platform uses processing units, namely ALUs, with centralized control. The ALUs have no dedicated local memory and no direct communication with each other. However, processing elements (PEs) can have their local memory and control logic in spatial platforms. PEs can directly communicate with each other to produce a processing chain. Moreover, CPUs use the single-instruction multiple-data (SIMD) for parallel computing, while GPUs use the single-instruction multiple-thread (SIMT).

In platforms with spatial architecture, the DNN implementation bottleneck is memory bandwidth. Therefore, local and global buffers are embedded in and among PEs to reduce data access from the DRAM. Accordingly, to execute convolution through MAC operations, PE arrays use direct-message passing to increase data reuse for decreasing the memory bandwidth. Different techniques for memory access reduction have been presented using local memory and data reuse by storing the data in a hierarchy to be exploited in DNN accelerators [37]: Input stationary (e.g., TPU [28]), weight stationary (e.g., SCNN [38]), output stationary (e.g., Origami [39]), and row stationary (e.g., Eyeriss [36]). In weight-stationary dataflow, the weights are stored in the internal register of PEs to be reused while inputs and the partial sums move among PEs. The input-stationary accelerator buffers the input activations in the internal registers, and convolution is done by passing the weight values to the PEs. In the output-stationary dataflow, partial sums are accumulated and kept in PEs until the final sums get ready to minimize the transfer of partial sums. Finally, row-stationary accelerators jointly maximize the reuse of the input activations, filter weights, and partial sums.

E. TERMS AND CONCEPTS OF DEPENDABILITY

This subsection overviews basic terms and concepts related to dependability theory. Dependability is an umbrella term including several system attributes and countermeasures. In this context, a system comprises different hardware and software components, which may have subsystems themselves. Moreover, the system delivers certain services to other systems in the surrounding environment. A system is dependable if it guarantees to deliver its intended level of service to its users during a period [40].

As seen in Fig. 5, availability, reliability, safety, integrity, and resilience are attributes leading to the dependability of a system [7]. The availability of a system relates to the fraction

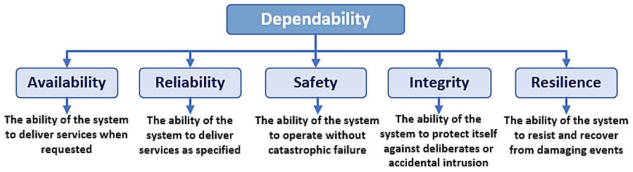

FIGURE 5. Attributes of the dependable system (redesigned of [164]).

of time during which it is ready to propose its correct service. On the other hand, reliability presents the probability that the system continues to perform its correct service over a period. Safety can be considered as an extension of reliability, namely reliability concerning failures that may create safety hazards. Safety can be seen as an attribute of the general concept of dependability. The term safety is commonly defined as a condition without danger or harm to humans or the environment.

Integrity has been provided if the system is not altered improperly. Lastly, resilience can be considered as a further desirable attribute of dependability, especially in complex systems that are subjected to change [41]. The resilience of a system is the persistence of dependability despite seen, foreseen, or unforeseen faults [42]. Here, in the context of the aging-aware DNN accelerator, we use resilience as the probability that the accelerator does not have a failure outcome despite the degradation of the underlying hardware.

Dependability threats are specified as faults, errors, and failures that put the system at risk, as well as countermeasures that can neutralize these threats [7]. Faults can be categorized as permanent, transient, or intermittent based on their temporal behavior. Transient and intermittent faults are both active over a limited time, but they have different causes. While transient faults are externally induced and occur at random positions, intermittent faults occur in burst due to internal shortcomings or instabilities and are usually recurring with some periodicity at the same location [43]. Four basic means against dependability threats are as follows:

1) Fault prevention reduces the possibility of fault occurrence through the design, deployment, and fabrication quality control. For hardware, this includes design reviews, component screening, and testing. For software, this includes structural programming, modularization, and formal verification.
2) Fault remova techniques reduce fault count or mitigate the severity. It is for the development phase (verification, diagnosis, and correction) or the operational life (corrective and preventive maintenance) of a system.
3) Fault forecastin gives a quantitative estimation of the present and future occurrence of faults and their consequence effects.
4) Fault toleranc is the ability to prevent failure and perform the intended functions in fault presence [44].

Fault tolerance can be classified into *passive*, *active*, and *hybrid*. In the passive case, the system exploits the implicit redundancies embedded in the system, which can mask the fault effects and ensure error-free outputs [45]. In this approach, we do not require diagnostics, reaction, location, or reconfiguration. In the active case, explicit and dynamic redundancy is used to detect and manage the fault effects by mechanisms like detection, localization, adaptation, retraining, or self-healing, which may lead to complexity [46]. For example, in the case of the DNN accelerator, this approach can be used for fault recovery by remapping, error detection, or reconfiguration [47]. Overall, not all failure scenarios can be considered at design time to provide passive or active arrangements. However, in a hybrid tolerance, passive and active approaches are used complementally to detect online and replace faulty modules [48]. Moreover, we have two kinds of redundancy in fault-tolerant systems: *space* and *time* [46]. In the first, additional elements, functions, or data are used, which lead to hardware, software, and information redundancy, respectively, depending on the type of redundancy added to the system. In time redundancy, the computation is repeated in fault presence.

TABLE 5. Comparison of safety levels [165].

Critical System Domain	Safety levels (high to low)				
Avionics DO-178C DAL [a]	A	B	C	D	E
Automotive ISO 26262 ASIL [b]	-	D	C/B	A	QM
General IEC-61508 SIL [c]	4	3	2	1	-
Railway EN 50128 SIL [c]	4	3	2	1	-

a Design assurance level, b Automotive safety integrity level, c Safety integrity level

III. SAFETY AND RESILIENCE

Generally, there are other types of critical systems rather than safety-critical, like mission-critical, mixed-critical, etc. The key difference between a safety-critical system and a mission-critical system is that a safety-critical system is a system that, if it fails, results in serious environmental damage or human injury, while the mission-critical system fails some goal-directed activities. Examples of mission-critical systems are a navigational system for a spacecraft and software controlling a baggage handling system of an airport.

A. SAFETY CERTIFICATION STANDARDS

As mentioned before, DNN accelerators are being employed in different domains, e.g., healthcare, defense, aerospace, cybersecurity, finance, drones, avionics, transportation, traffic control, railway, and autonomous driving. These disciplines contain safety-critical applications with an urgent need for dependability. In the dependability study, a system is considered safe if it is free of catastrophic failure consequences that may lead to harm the human or the environment [7]. In ISO 26262, the potential source of harm is hazard [49] which can cause unreasonable risk and thus prevent safety.

Regardless of the critical system domain, validation and certification of safety are done by international safety standards comprising guidelines. The safety levels of some major domains are demonstrated in Table 5. For example, the standard RTCA/DO-178C [50] guides several objectives to be fulfilled for safety-critical airborne software. It assigns a safety factor, namely design assurance level (DAL), to

software, ranging from level A (high critical) to E (low critical), depending on the evaluation process and the criticality determination. Higher safety levels require more strict analyses of both hardware and software to ensure the required assurance levels. In addition to RTCA/DO-178C, a framework for safety analysis can be found in the standard ISO 26262 proposed for the automotive disciplines [2]. It defines safety as the absence of unreasonable risk, where risk is a factor that combines the severity and the probability of harm, i.e., physical injury or damage to the health of persons. Safety-critical systems must follow strict certification processes according to domain-specific standards to reduce the risk of causing such casualties.

The generic safety standard of IEC 61508 [37] is accepted as a reference by several domains, such as robotics (ISO 10218 [51]) and railway (EN 5012X [52]). However, among all these safety standards, there is considerable variability in terminology, definitions, and requirements. For example, in IEC 61508, the safety integrity level (SIL) is defined with a discrete level range from SIL1 (the lowest) to SIL4 (the highest), while in ISO 26262, the level is defined by the automotive safety integrity Level (ASIL) with a range from A the lowest (ASIL-A) to D the highest (ASIL-D).

B. SAFETY TECHNICAL REQUIREMENTS

According to IEC 61508 and ISO 26262 standards, to develop a DNN accelerator-based safety-critical system, the following fundamental safety technical requirements, already declared for multi-core platforms, must be addressed [53]:

1) Reliability of a component is defined as the ability to perform as required, without failure, for a time interval, under given conditions. It is quantified by FIT (Failure in Time) or failure count per 10^9 hours of operation.
2) Diagnostic Coverage (DC) means the fraction of dangerous failures detected by automatic online diagnostic tests. It is expressed as coverage percentage classified in ranges: low ($60\% \leq DC < 90\%$), medium ($90\% \leq DC < 99\%$), and high ($DC \geq 99\%$).
3) Temporal independence suggests that one element shall not cause another element to function incorrectly by taking too much execution time of the processor or blocking another element by locking a shared resource. The reference standards recommend techniques like WCET (worst-case execution time) analysis, temporal diagnostics, and predictability (e.g., design time scheduling).
4) Spatial independence indicates that data used by one element shall not be changed by another. For instance, a widely used technique to support spatial independence is exclusive access to shared memory [54].

C. AGING-AWARE RESILIENCE

Resilience is an important dependability feature for a safety-critical system operating in an open context, i.e., for which a definitive characteristic of the desired output in response to all possible inputs is impossible due to the application and the environment's complexity. We face this open-context dependability challenge in upcoming technologies like autonomous vehicles [55]. DNNs are enabling such technologies due to their ability to deal with open-context situations. However, to guarantee safety, we must assess and enhance the resilience against faults of the underlying hardware platform.

DNN is considered to have a degree of inherent resilience against faults of the underlying hardware. This is due to several factors: First, DNN includes an extra number of neurons (more than needed) to perform a specific task [59], so it can continue its overall operation despite a few faults. Second, DNN can tolerate some kind of faults in computation because of its similarity to the human brain [56]. Third, the learning enables DNN to find solutions to problems that are not trained for [57], so the learning capability allows DNN to have a high degree of resilience to computational errors and noise in input data. Moreover, there are other reasons, including widely used activation functions, pooling utilization, and output generation based on ranking. However, inherent resilience does not guarantee fault tolerance against aging which may cause substantial accuracy loss or lifetime reduction. Eventually, we require more attention to the resilience of DNN accelerators, especially in the safety-critical discipline [58], [59].

To explore resilience, the vulnerability of the DNN algorithms and architecture must be evaluated. There are two major approaches for evaluating resilience: 1) Experimental and 2) Statistical or theoretical. In the first one, physical faults are inducted in real hardware devices [33], [34], emulated on hybrid platforms [60], virtually injected in model simulation [34], [61], or eventually, errors are behaviorally simulated at the algorithm level [62], [63]. According to the literature, the majority of research on resilience evaluation in DNN accelerators has been experimental. Overall, experimental methods use different features like the type and abundance of errors, the data type and precision, the DNN type and structure, and the location of error occurrence. However, while this approach is useful for an accurate resilience evaluation, it has heavy computation and provides only a limited vision of design choices to improve resilience during the development phase. The second approach proposes a theory-guided approach that has the advantage of interpretability and avoids heavy fault injection. In [64], the authors proposed a method to evaluate the forward error propagation of neurons with PSFs. They determined design factors that affect the forward error propagation as activation function and number of neurons per layer. Another analytic technique for DNN resilience prediction is proposed in approximate computing [65]. Here, a technique to determine weight updates in DNN training, namely backpropagation of error gradients, was proposed to estimate the output vulnerability to internal errors of neurons.

Our focus in the proposed survey is to classify the prior research works on their contributions into three categories: The first class is *functional resilience,* including software

or hardware techniques to avoid the occurrence, remove or manage the propagation of PSFs. Functional resilience can be achieved, for example by passive fault tolerate architecture, active redundancies, or online error detection and correction mechanisms. The second class is *timing resilience*, which is the ability at design-time or runtime to evolve or dynamically adapt to timing variations induced by aging. *Graceful degradation* by frequency scaling, i.e., the ability to continue DNN accelerator operation with only minor performance decreases in the presence of TEs, is related to this as well. Finally, the last class is *lifetime resilience* which is related to design-time or runtime methods of conserving or even extending the lifetime of the DNN accelerator, e.g., guard-banding.

Herein, we summarize studies performed to enhance the resilience of DNN accelerators with a layer-wise approach in different resilience classes. In the following, insights from the literature for each resilience class are presented.

IV. AGING-AWARE FUNCTIONAL RESILIENCE

In this section, we discuss prior studies of functional resilience improvement against permanent faults in DNN accelerators considering aging mechanisms, e.g., TDDB and EM. To mitigate or even remove permanent fault effects, a lot of work has been conducted in different layers with various means, including prevention, removal, forecasting, and tolerance.

In this review, we investigate proposed studies with a layer-wise approach that matches the general DNN deployment stack, including high-level models to low-level circuits. Fig. 6 illustrates the taxonomy of existing functional resilience-improving techniques in layers of the DNN deployment stack. Herein, we explore works done in 5 layers, including 1) model, 2) algorithm, 3) architecture, 4) microarchitecture, and 5) circuit and device. According to this figure, the DNN model's inherent resilience, training approach, sensitivity analysis and limiting of activators and weights, and dataset effects are explored in the first layer. In the algorithm layer, resilience enhancement based on data and software redundancies for different parts of the network and with software or hardware deployment, and variable precision quantization are considered. Resilience enhancement using DNN high-level structural factors like NN layers, PE and ME mapping, replication and redundancy, pruning, selective hardening, fault-aware mapping, deactivation, and reconfiguration are presented in the architecture layer. In the microarchitecture layer, low-level DNN factors such as datatypes, numeric systems, computing element duplication and approximation, and selective bit hardening are considered. Finally, circuit and device layer techniques focus on low-level design features such as partial reconfiguration, memory cell design, synthesis, and sensor design for fault detection. Cross-layer affords usually combines techniques of different layers in a unified framework to achieve effective protection with low overhead [66]. In the following, we explore some of the major research conducted in each layer in more detail.

A. MODEL-LEVEL RESILIENCE

This subsection summarizes the research done at the top level of the system stack. High-level techniques (i.e., the model level and algorithm level) can have prohibitively lower costs, especially scaling up the complexity of DNNs while providing acceptable resilience because they have indirect mitigating effects on more fault types of the underlying hardware. There exists a big body of work exploring model-level factors to facilitate resilience with lower overhead. The inherent resilience of DNN is the key feature used by all high-level techniques. Accordingly, only a fraction of neurons are vulnerable and have serious effects on output accuracy. Most of the proposed works try to mitigate and compensate for vulnerability through sensitivity analysis and training.

The major method for model-level resilience improvement is sensitivity-based training and retraining of the DNN model. Thereby, the resilience of feedforward NN considering the sensitivity of neurons was explored in [67]. It deploys a cross-layer technique to design networks with the required resilience satisfaction. This method starts with a well-trained network and follows two main approaches, i) removing unimportant nodes in the hidden layers using the sensitivity threshold, and ii) retraining the pruned network and adding some redundant nodes to share the load of the most critical ones in the network. Here, the sensitivity of links and nodes is evaluated by output accuracy in mean square error (MSE). After that, this work is extended by modifying the training method and using explicit redundancy. This work restricts weight values throughout the training to have lower magnitudes because of the resilience reduction by high-magnitude weights. However, these methods require an exhaustive simulation to measure neuron criticality, which is infeasible for large-scale DNNs. In the work of [68], a training approach to adjust neuron sensitivities is introduced using an analytical approach instead of fault injection in training. The authors claim that homogeneous (uniform) resilience in the DNN can help to avoid special protection of critical parts. Also, it introduces an explicit weight rescaling method to balance the vulnerability of channels in a layer conducting the iterative rescaling and finetuning. It proposed minor changes in layer placement of DNN architecture to avoid layers with few neuron outputs (low resilience ones). Finally, neurons with more than average criticality scores are pruned, and neurons with less than average criticality scores are expanded.

Overall, two major approaches can be identified for resilience-aware training: fault-aware training and changing the learning rules. Fault-aware training tries to obtain a resilient model by adding noise to input activations and weights or direct error injection to PEs during the training. The study of [69] explores DNN training to improve functional resilience in permanent faults presence. This method uses a fault map generated from a post-synthesis simulation at design time [70] or a post-fabrication test for a specific chip [66]. The produced fault map is used during training to allow DNN adaption according to the fault presence at

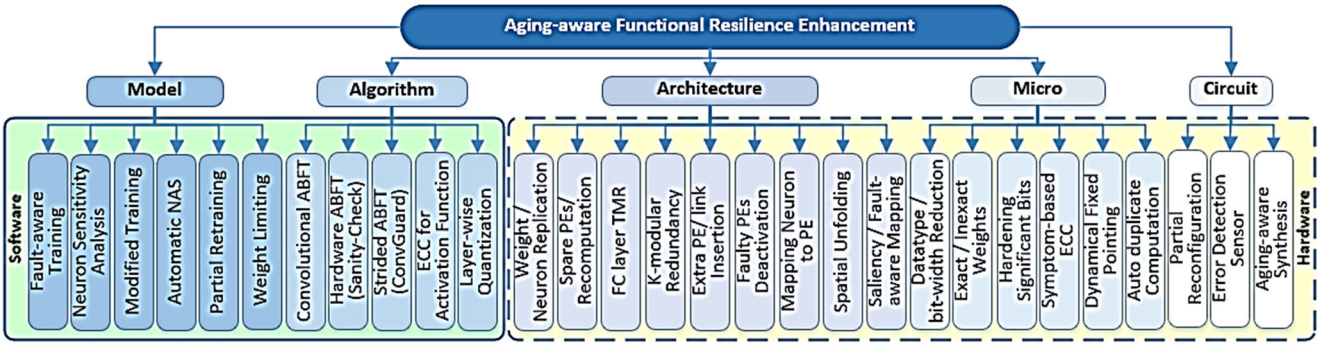

FIGURE 6. Taxonomy of techniques for functional resilience enhancement.

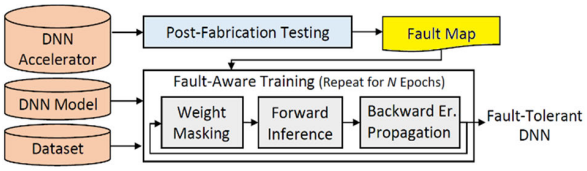

FIGURE 7. The flow of fault-aware training (modified from [66]).

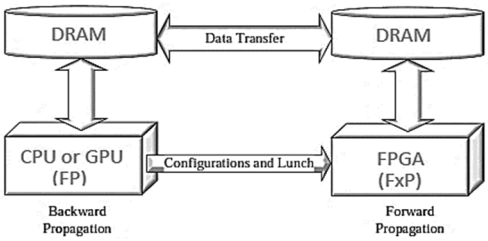

FIGURE 8. The modified training of which performs FWP using FxP on the accelerator and BWP using FP on the GPU (modified from [76]).

the hardware. Fig. 7 shows the flow diagram of fault-aware training after fabrication. The drawback of the proposed approach is the complexity of the training loop by fault injections or realistic fault simulations. Other works establish random weight or activation fault models to inject fault during training and mitigate the effects of erroneous inputs and weights [71], [72], [73]. In this way, the model can learn to tolerate the type of injected fault to mitigate the sensitivities of neurons and prevent accuracy loss. According to [74], DNN accuracy is more sensitive to weights with large magnitudes, and there are opportunities to improve functional resilience by providing different protection mechanisms for weights and activators, such as limiting weight magnitudes in training [75].

Changing the learning rules can also be done by regularization of conventional algorithms, adapting backpropagation, or modifying learning algorithms to find weights values that are more equally distributed and avoid saliency. A technique of modified DNN training method considering the resilience issues is proposed by [76]. Herein, backward propagation of training is done on the error-free GPU or CPU while the forward inference is performed on the error-prone FPGA-based accelerator. This technique accounts for different sources of error during DNN training to compensate for the predicted accuracy loss, as shown in Fig. 8. The training is finished when the accuracy loss goes below the threshold, and it implies that the trained DNN can be safely used on the error presence. A challenge of the proposed technique is that it has a high overhead requiring conversion between fixed-point and floating-point in each iteration of the training.

The state-of-the-art research [77] proposes an approach to improve functional safety (FuSa) in DNN accelerators for mission-critical applications using the concept of generative adversarial networks (GANs). The proposed method produces a set of functional test patterns based on GAN that are independent of the DNN model and the accelerator features. It injects memory faults that may violate FuSa and tries to detect unsafe application behavior using the generated test patterns to improve the training process and functional resilience.

Also, if the error rate exceeds a threshold, some works conduct retraining to adapt the DNN weights for increasing functional resilience and mitigating the impact of errors. Moreover, retraining can only be done in the limited sections of DNN, like fault-prone FC layers, to reduce the overhead of complete retraining [78].

Apart from manual model design and modification, state-of-the-art work [79] proposes FTT-NAS framework (fault tolerant training - network architecture search) to automatically find and tune a resilient neural network model. The framework first finds a primitive resilient model and then applies fault-tolerant training to improve its resilience and precision. NAS includes a controller to sample different architecture rollouts from the search space and a shared weights-based evaluator to evaluate the performance of different rollouts on datasets using fault-tolerant objectives.

B. ALGORITHM-LEVEL RESILIENCE

Although DNNs have a certain degree of the inherent resilience of algorithms, improving functional resilience by exploiting the algorithmic features is an active field

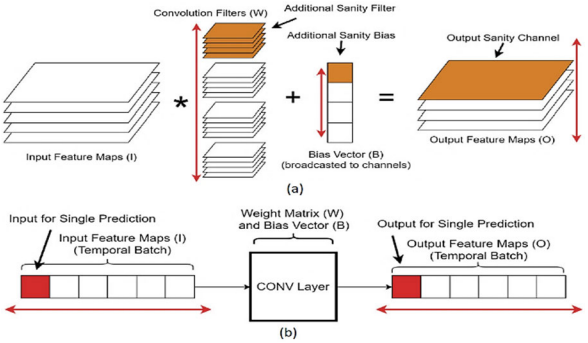

FIGURE 9. The spatial (a) and temporal (b) checksum in the convolutional re-illustrated from [82]).

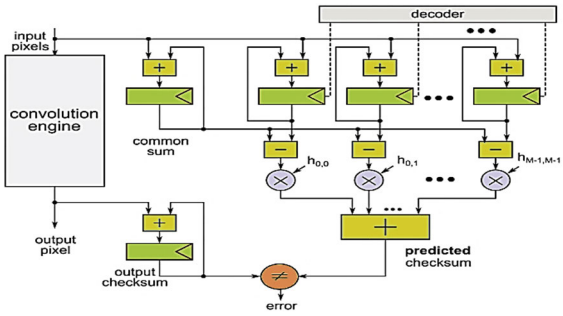

FIGURE 10. Generalized ConvGuard architecture that supports arbitrary convolutions (modified from [83]).

of research. In the following, we try to summarize techniques that have been taken. The main approaches for algorithm-level resilience improvement are layer-wise quantization and algorithm-based fault tolerance (ABFT).

Layer-wise quantization tightens the safety margins and matches the utilized range in different DNN layers to reduce the vulnerability. In state-of-the-art work of [80] a new regularization method namely outlier is proposed in the training phase to further tighten the numeric range and shape the distributions of parameters. Herein, DNNs of higher resilience merely through algorithmic modifications can be achieved by the proper design of the quantization schemes for each layer using the guidance offered by appropriate training technique.

Noting that a large fraction of DNN computation is spent on convolutions, [81] employs ABFT to improve functional resilience in this layer. In the case of matrix multiplication as the basic block of convolution, this research adds the sum of rows and columns to the original input matrices. In the output, the sum of both extended multiplication and accumulation of output matrix elements are compared to detect error occurrence. In other words, the checksum of the results was calculated by two approaches on both row and column to detect and recover errors. This work proposes a workflow to integrate techniques with limited overhead and conducts ABFT for runtime error detection and correction, supporting different convolution implementations and fault configurations.

The authors of [82] proposed the Sanity-Check, which is an error detection mechanism for the convolutional layer using linear checksums. They introduced a dedicated hardware unit integrated with modern DNN accelerators to mitigate the performance overhead of the software methods with a little area and power overhead. Sanity-Check offers an alternative for low-cost error detection in safety-critical DNN applications based on spatial and temporal checksums. In the first, the output checksum is calculated in the spatial dimension. In the second, the summation is performed through the point-wise accumulation of the inputs over the temporal batch and compared with the expected value, as illustrated in Fig. 9.

To reduce the overhead, a lightweight ABFT, namely ConvGuard has been proposed by state-of-the-art research [83], which only predicts the convolution checksum on the border pixels of the input feature map. ConvGuard computes the checksum of the output pixels and compares it to its predicted checksum value in parallel to the operation of the convolution engine. Also, the stride convolutions are used on selected sub-image / sub-filter pairs. After computing the convolution, the appropriate common sums per channel are subtracted from the coefficient accumulators, as shown in Fig. 10. The common sums and the filter coefficients related to the same channel are selected by the mask logic. Similar to reducing the overhead of retraining and as the last layer of DNNs directly affects the output, [84] proposes a method based on the ECC instead of conventional SoftMax to tolerate permanent faults. Specifically, this research uses an variable-length decode-free ECC coupled with a collaborative logistic classifier for asymmetric binary classification to detect and correct errors in the output.

C. ARCHITECTURE-LEVEL RESILIENCE

In safety-critical discipline, it is insufficient to protect the DNN accelerator with only model, algorithm, or inherent resilience because of the progressive aging fault configuration space that cannot be considered in design time. Also, model layer approaches typically require time-consuming training to cover the runtime faults. Moreover, algorithmic techniques usually rely on DNN input data which may not always be accessible during the deployment step. Architectural techniques for resilience enhancement are investigated to mitigate aging effects with fewer limitations compared to high-level algorithms. There are three high-level architectural approaches for improving resilience: redundancy, pruning, and mapping.

1) RESILIENCE ENHANCEMENT BY REDUNDANCY

Redundancy is the conventional mechanism of resilience improvement. A forefront mechanism called augmentation is proposed by [85] which can improve resilience by replicating hidden neurons and their associated connections. Also, [86] proposes a technique to detect more sensitive elements and duplicate inputs, biases, weights, or even neurons, according to the evaluation criteria. Moreover, state-of-the-art research [87] focuses on selective sensitive neuron

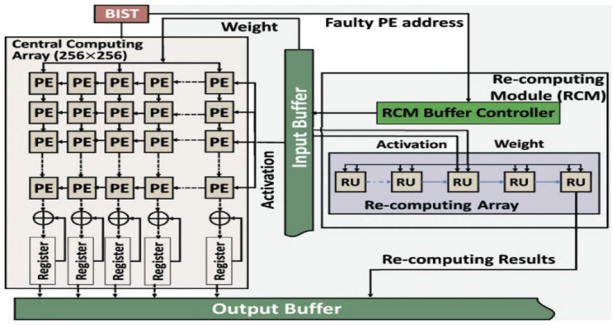

FIGURE 11. The fault-tolerant systolic array (modified from [91]).

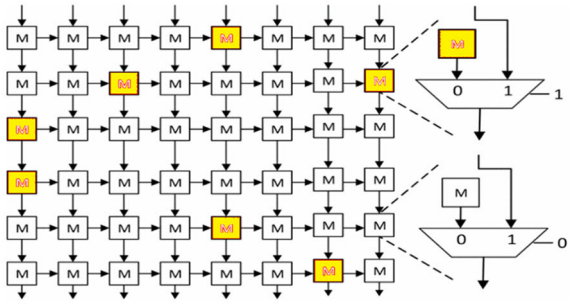

FIGURE 12. Fault-aware pruning (modified from [95]).

duplication to achieve higher robustness with lower overhead. Here, selective redundancy is only applied to critical neurons.

The study of [88] proposes an approach of k-modular redundancy which is created by k-fold replicating of hidden neurons and dividing the weights to k compared with the voting mechanism of the conventional methods. The authors evaluated the method for three applications on the NN accelerator: Black-Scholes, RSA, and Sobel. It concluded that for small-size NN, the proposed method is more effective. Moreover, [89] proposes FT-HNN (fault-tolerant Hopfield neural network) on the FPGA by inserting additional links to the baseline and providing hidden accumulation states to vote among them for fault tolerance. The proposed FT-HNN requires less hardware overhead compared to the baseline TMR because of sharing partial results in accumulations.

Moreover, [90] introduces a method of adding spare Pes, which can be reconfigured to replace any neuron. It can be used for both fault detection and correction, but it is limited to spatial architecture and can be used to recover from limited faulty PEs. In the state-of-the-art study [91], the authors proposed FSA (fault-tolerant systolic array) by adding re-compute units to the base accelerator array to conserve accuracy loss in the presence of PSFs. The key factor of FSA is a unified re-computing module (RCM) that dynamically recalculates computations of faulty PEs with minimal latency and power consumption (Fig. 11).

According to previous layers, there is an observation that the activation function and adders of the output layer have the most effect on the output accuracy, thereby the resilience of the NN accelerator can be increased by hardening these elements [92]. Based on this, [93] selectively applies TMR to the hidden layer and the last FC layer to provide a better balance between resilience and resource utilization than the full TMR. Also, the study of [94] proposes SHIELDeNN, an online resilience assessment approach, to determine the most vulnerable elements of DNN. Based on the vulnerability assessment, a hardening strategy based on selective TMR is employed to provide an optimal design for a resilient DNN accelerator with resource constraints.

2) RESILIENCE ENHANCEMENT BY PRUNING

A permanent fault-aware pruning technique to improve the functional resilience of systolic array DNN accelerators is presented by [95]. The proposed technique drops the mapped computations to a faulty PE. Fig. 12 shows the hardware modification needed to enable this technique. Also, [58] presents a technique to add a zero-bypass data path for PEs which will be enabled in PSF presence in PEs. Zero-bypassing can cause a loss of accuracy, although the impact on DNN accuracy is more predictable than random erroneous values. After pruning, the authors used retraining to address the accuracy and adapt to different fault configurations. The state-of-the-art study of [96] presents an application-driven approach to bin the DNN accelerator chips and reduce yield loss by correlating the faults in the PEs with the desired accuracy of the target AI application. The authors explored the inherent resilience of trained models and a strategy of selective faulty PEs deactivation to mitigate yield loss. They derived an analytical relationship among fault location, fault rate, and the AI task's accuracy to decide if the accelerator chip could pass the final yield test. Finally, yield-loss-aware fault isolation and test flow are provided for the PEs.

3) RESILIENCE ENHANCEMENT BY MAPPING

As seen, the state-of-the-art research works address functional resilience by adding homogeneous redundant PEs to the baseline array and bypassing faulty PEs. However, this methodology induces accuracy loss, extra hardware costs, and performance overhead. Another approach to increase functional resilience is to change the mapping of neurons to PEs. SalvageDNN [97] proposed a mapping methodology to reduce the effects of pruning on functional resilience without the need for retraining. It employs a fault-aware mapping of DNN elements on the hardware accelerator by leveraging the saliency of the DNN parameters and the underlying hardware fault map. Moreover, it introduced modifications in the design of the systolic array DNN accelerator to further improve resilience. Fig. 1 demonstrates an example of the mapping adaptation by the fault map change corresponding to a systolic array including 16 PEs. Black and red crosses correspond to fabrication and wear-out faults, respectively. Also, [98] considers the neuron's sensitivity in the DNN

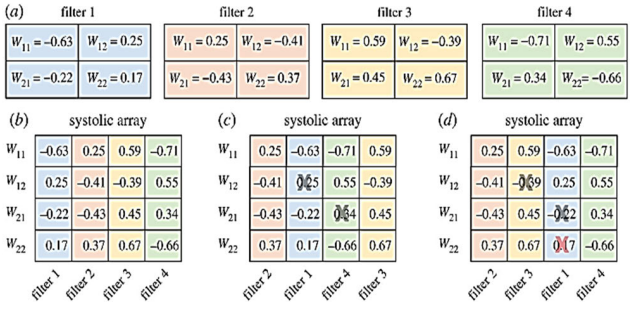

FIGURE 13. Mapping adaptation example (modified from [97]).

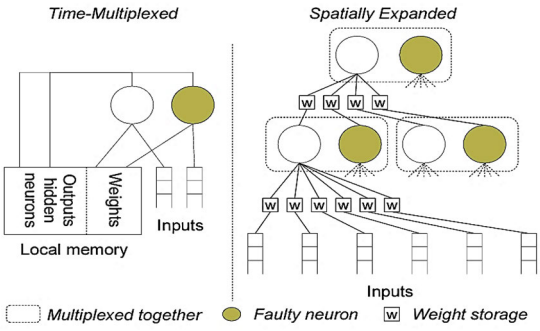

FIGURE 14. Implementation of an accelerator using (a) time multiplexing or (b) spatial-unfolding approach (modified from [92]).

model and proposes a heterogeneous array of PEs in two levels of resilience. In this case, more sensitive neurons are allocated to highly resilient PEs of the computing array while lower sensitive neurons are allocated to the rest of the PEs, which ensures higher resilience with lower overhead.

Compared with time-multiplexed architecture, the research of [92] proposed a design with spatially unfolding for a DNN accelerator to achieve higher functional resilience (Fig. 14). Spatial-unfolding mitigates energy consumed by data movement, placing PEs closer to MEs. Despite the time-multiplexed architecture with a high impact of faults on accuracy and resilience reduction, the impact of faults on spatially-unfolded designs is more restricted. Herein, the control unit is employed only at the input/output stage, and its vulnerability can be easily mitigated by hardening. Eventually, in the proposed design, if the size of DNN exceeds the maximum size of the spatially unfolded accelerator, time multiplexing can be used to reuse the computing resources.

D. MICROARCHITECTURE-LEVEL RESILIENCE

In this subsection, we explore several research works conducted in the microarchitecture layer to improve functional resilience based on data precision adjustment, selective bit hardening, symptom-based ECC, and computing unit duplication. To increase functional resilience, some works store weights with different formats, e.g., floating-point and fixed-point [69], or hold both exact and inexact weights [99] to ensure training convergence. The state-of-the-art study of [100] explores the resilience mitigation approaches of neuromorphic DNN underlying hardware. First, it considers the DRAM memory of commercial systolic array DNN accelerators in the presence of faults. Then, it quantifies the impact of fault presence on output accuracy considering operators' LSBs to MSBs. Finally, it proposes a mapping approach to improve endurance in emerging neuromorphic hardware.

Moreover, [101] proposes techniques for fault effects mitigation in DNN accelerators. First, the authors exploited layer-wise data precision adjustment with a numeric range just sufficient for the values of the layer. As an example, they adjust fixed-point numbers precision to reduce the FIT (failure in time) rate of the data path. Next, they hardened the most significant bits to reduce SDCs (silent data corruption). These hardening schemes mitigate the FIT rate of the data path by two orders of magnitude with only a 20% area penalty. They also suggested adding normalization layers in the DNN model and placing circuit-level error-detection sensors after them to avoid benign faults identification.

Although on-chip buffers allow data reuse to reduce data movement energy, these buffers have high sensitivity, which makes it difficult to protect them through ECC. The study of [102] proposes a symptom-based error detection mechanism to protect buffers. First, it learns the thresholds considering the typical range of values in each layer with fault-free execution and adding a guard band. Then, operational values are compared with these thresholds to identify errors during actual execution. Here, the latency can be ignored by detecting errors at the time of data usage as the input of the next layer. However, this method is effective only for networks and datatypes in which the internal values lie in a reasonable range and show strong symptoms. Also, [103] proposes a cross-layer approach employing the dynamic fixed point (DFP) and device variability-aware (DVA) training methodologies to improve the resilience of ReRAM based DNN accelerator. The authors used the DFP for representing data with dynamic decimal point position considering the data range and minimizing the unused most significant bits (MSBs). They employed a DVA training method with stochastic noise addition to the parameters during training to improve the resilience of the network against faults and variations or improve the robustness to input data noise.

Moreover, [104] exploits the advantages of relaxing some PEs due to inherent NN resilience instead of using TMR to protect them. This technique employs selective TMR only for cells of the ripple-carry adder, which affects the higher bit position of the output (Fig. 15). It protects cells related to the higher output bits more seriously, and the protection strategy depends on the corresponding resilience level of the applications. The experimental results illustrated less chip area and a shorter critical path compared to full TMR protection under a similar protection level.

Another approach to increase functional resilience is to provide different protection for weights and activators. The research of [105] introduces a significance-driven mapping of weight bits to memory cells with different resilience.

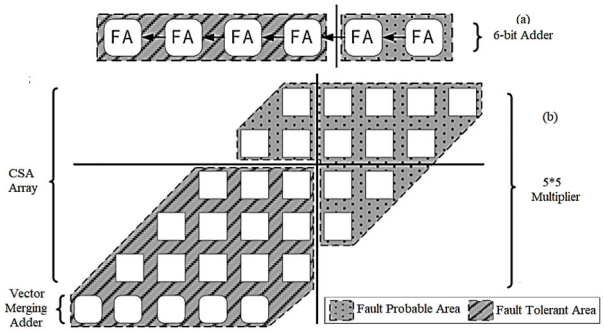

FIGURE 15. (a) A relaxed 6-bit adder with AUC = 2 (FA = full adder, CSA = carry-save adder) (b) A relaxed 5 × 5 multiplier (re-illustrated from [104]).

Similarly, [106] presents the partial mapping of different neurons to inexact or unreliable PEs to increase efficiency. Finally, another state-of-the-art research [107] proposes an automated tool to evaluate the sensitivity of running DNN inference to be used in microarchitecture-level selective TMR. The tool triplicates the most sensitive computations to increase the resilience of the DNN accelerator without full TMR. Designers use this tool to provide the trade-off between resilience and hardware costs.

E. CIRCUIT-LEVEL RESILIENCE

This subsection discusses techniques conducted in the circuit layer. Previously, many techniques were presented for circuit-level functional resilience improvement using hardware reconfiguration. Herein, partial reconfiguration is the major approach to repair components and mitigate permanent faults. The research of [108] proposes a small flexible controller to drive the reconfiguration process in PSF presence. This technique uses alternative pre-synthesized configurations for permanent fault reduction, bitstream relocation reducing the required bitstream, and fault type detection in case of success in previous repairs, etc. In [109], the authors presented a method to extend the lifetime of SRAM-based FPGAs in space missions. They focus on recovering permanent faults. This technique uses a fix-sized fault detection module to detect permanent faults and propose a permanent fault recovery mechanism. By partial reconfiguration, they developed a lifetime estimation model to find an optimal partition for designing the module-based fault recovery with the maximum lifetime. Moreover, dynamic partial reconfiguration (DPR) has been employed as a solution to deal with permanent faults in space-borne using off-the-shelf FPGA devices. DPR mechanisms detect the permanent fault in a module and perfor the reconfiguration process. The amount of silicon resources reserved for DPR is the main issue and is different for the same module in the designs employed so far.

Finally, Table 6 presents the summary and key factors of proposed techniques in the literature to enhance functional resilience at different levels of the DNN deployment stack, which was indicated with different colors similar to Fig. 6.

V. AGING-AWARE TIMING RESILIENCE

In digital circuits, TEs occur due to several phenomena like process, voltage, and temperature (PVT) variations or aging [110]. We should mention that aging and the consequent TE rate are increasing in the scaling of emergent process technologies. TEs can lead to performance reduction, energy penalties, and also a big loss of accuracy. In this section, we discuss prior research related to the timing resilience of DNN accelerator against TE that occurs due to aging mechanisms, i.e., NBTI and HCI. Fundamentally, timing resilience in digital circuits is based on the observation that timing critical paths are rarely activated, and the worst-case clock cycle produced by statistical timing analysis is used infrequently. Thereby, the circuit has an inherent timing resilience, and if the errors can be handled, the system can continue to perform in the case of TE while improving the lifetime and performance or even reducing the power consumption. To mitigate TE and improve resilience, many studies have been conducted in different layers and life spans of the system, including overheads in terms of latency and power consumption. According to the four basic countermeasures of dependability specified in Section II-B, we can classify proposed techniques according to the approach they take facing TEs:

1) TE avoidance (TEA) by design time methods such as transistor sizing, fault-aware synthesis, or static timing analysis for conservative guard bands to eliminate the possibility of TE at the expense of efficiency loss.
2) TE propagation (TEP) scheme ignores the TEs and lets to move forward relying on inherent resilience or mechanisms arranged in design time based on TE tolerance or graceful degradation. Inherent resilience is not quite effective alone if the accelerator is blasted with intermittent TEs. TE mitigation techniques are based on fault-aware training, masking, skipping, normalization, quantization, etc. The common denominator of both TEA and TEP is to characterize the timing resilience at design time.
3) TE detection (TED) tries to detect errors in runtime using Razor-like flip-flops [111] or checksum coding [112]. Afterward, removes errors or mitigates the effects by runtime (active) methods like ECCs [9], [99], dropping [113], or mapping [74], respectively.
4) TE forecasting (TEF) prevents TE occurrence by relying on runtime TEs prediction and monitoring like the Canary circuit [69] or dynamic timing analysis [110]. We have different runtime adaptation mechanisms considering TEF, such as model retraining [95], hardware reconfiguration [47], and DVFS adaption [114].

The Razor is a circuit-level mechanism for timing speculation based on the dynamic detection of critical path errors. It uses a double sampling flip-flop to detect TEs in a pipelined design [111]. A timing mis-speculation causes a delay alarm which is generated by comparing the speculative execution output with worst-case assumptions. In such cases, recovery

TABLE 6. Functional resilience evaluation and improvement techniques and their key characteristics.

Technique	Summary	Layer	Platform	Mean/ Scope	Design / Runtime	Cost, Dependency	Works Done
Fault-aware Training	In this approach, first, a fault map is derived using post-synthesis simulation at design-time or post-fabrication testing for a specific chip, then it can be used during training to allow DNN adaption according to the faults in the hardware.	Model	All	Tolerance	Design-time	Training overhead	[69], [72] [103] [99]
Modified Training Scheme / Weight Restriction	In this approach, training is done on the error-free GPU or CPU. The inference is performed on the error-prone accelerator. Also, it employs the iterative elimination of unimportant neurons and network retraining to reach target accuracy.	Model	All	Tolerance	Design-time	Training overhead	[76]
Automatic NAS	This method adopted the manual neural architecture search method to automatically find resilient networks and apply a fault-tolerant training approach to obtain both reliable and high-precision models.	Model	FPGA/ ASIC	Tolerance	Design-time	Design overhead	[79]
Retraining	This technique retrains the model to increase functional resilience by compensating for accuracy loss due to PSFs.	Model	All	Tolerance	Runtime	Runtime overhead	[78]
Checksum ABFT for Convolution	In this technique, the sum of rows/ columns is added to the input matrices, and in the output, the sum of both extended multiplication and accumulation of output matrix elements are compared to detect and correct error occurrence.	Algorithm	All	Removal	Runtime	Process Memory overhead	[81]
Hardware ABFT (Sanity-Check)	It proposes a dedicated hardware unit that integrates with modern deep learning accelerators to use linear algorithmic checksums without the performance overhead of the software-based implementation. Sanity-Check proposed spatial and temporal checksums.	Algorithm	All	Removal	Runtime	Area, power overhead	[82]
Stride ABFT	It predicts the checksum of convolution output implicitly by accumulating only the pixels at the border of the dropped input features. The stride convolutions are used on selected sub-image / sub-filter pairs once for a convolution.	Algorithm	All	Removal	Runtime	Run, power overhead	[83]
Activation Function ECC	A method ECC in the last layer instead of conventional SoftMax to tolerate permanent stuck-at fault. This method uses an optimized variable-length decode-free ECC coupled with a collaborative logistic classifier to detect and correct errors.	Algorithm	All	Removal	Runtime	Run, power overhead	[84]
Layer-wise Quantization	It presented the possibility of inherent resilient features of DNNs by the proper layer-wise design of the quantization mechanism and shaping the parameter distributions according to training methods.	Algorithm	All	Tolerance	Runtime	Design complexity	[80]
Uniform Resilience Distribution	This technique distributes the resilience uniformly avoiding the extra protection of critical portions. It proposes two schemes: (1) adapting the weights and (2) changing layer placement. Neurons with high criticality scores are pruned, and neurons with low criticality scores are increased.	Model / Architecture	All	Tolerance	Design-time	Training overhead	[68]
Hidden Node Redundancy /Pruning	In this method, first, unimportant neurons are pruned, then redundant neurons are added in sensitive parts of MLPs to improve resilience.	Architecture	All	Tolerance	Design-time	Sensitivity overhead	[86]
K-modular Redundancy	K-modular redundancy is created by hidden neuron replication in a non-redundant NN by k-fold and dividing the weights by k.	Architecture	All	Tolerance	Design-time	Area/ Power overhead	[88]
Selective Modular Redundancy	Online resilience evaluation, to determine the most vulnerable elements of DNN for a hardening strategy using selective modular redundancy to improve resilience with resource constraint.	Architecture	FPGA	Tolerance	Design-time	Hardware overhead	[92] [94]
Spare PE Addition	A method of adding spare PEs which can be reconfigured to replace any neuron in the model. It can be used for both fault detection and correction, but it is limited to spatial neural network architecture.	Architecture	All	Tolerance	Runtime	Hardware overhead	[90]
Recompute	The main idea is to add extra computing units embedded beside the main DNN accelerator on extra PEs to recompute the operations of the faulty PEs.	Architecture	All	Tolerance	Runtime	Area, power overhead	[59] [91]
Hidden Accumulation	Insert additional links to obtain hidden accumulation states differently to vote them for fault tolerance. The proposed model requires less hardware overhead compared to the baseline TMR implementation for sharing partial results in accumulations.	Architecture	All	Tolerance	Design-time	Area, power overhead	[89]
Significance-driven Mapping	A significance-driven mapping of weight bits to memory cells with different resilience.	Architecture	All	Tolerance	Design-time	Sensitivity overhead	[105]
Fault-aware Mapping	It employs a fault-aware mapping of different parts of a given DNN on the hardware accelerator by leveraging the saliency of the DNN parameters and the fault map of the underlying PE.	Architecture	All	Tolerance	Runtime	Power overhead	[97]
Fault-aware Pruning	A fault-aware pruning technique for improving functional resilience in systolic array DNN accelerators. The proposed technique prunes/drop the computations that have to be mapped onto a faulty PE.	Architecture	All	Tolerance	Runtime	Fault map overhead	[97] [95]
Spatial Unfolding	Spatial-unfolding mitigates data-movement energy by placing neurons (PEs) closer to the synapses (MEs). In time-multiplexed, any fault in hardware has a high impact on the output accuracy and can cause accelerator resilience loss but the impact of faults in spatially-unfolded are more restricted.	Architecture	All	Tolerance	Design-time	Hardware overhead	[92]
Store Weights with Different Formats	Some works store weights in different formats or both inexact and exact weights to ensure training convergence.	Micro	All	Forecasting	Design-time	Process/ Memory	[69] [99]
PE Deactivation	It explored circuit aging and endurance in emerging hardware platforms to present an architecture-level approach to mitigate them. It proposed an approach based on the Faulty PE deactivation and mapping using fault and sensitivity information gathered from circuit and microarchitecture levels.	Architecture	Systolic / SIMD	Tolerance	Runtime	Area/ power overhead	[100]
Fxp and FP/ Inexact and Exact Weights	Store weights with different formats, i.e., Fix-point and Floating-point, or both inexact and exact weights to ensure training convergence while increasing functional resilience.	Micro	All	Tolerance	Runtime	Overhead	[99] [69]
Symptom-based Error Detection	First, it learns the typical range of values in each layer with fault-free execution and adds a 10% guard band. Then, the values are compared with these thresholds to identify errors during actual execution.	Micro	All	Tolerance	Runtime	Overhead	[101]
Selective TMR	It employs selective TMR for ripple-carry adder cell protection in DNN accelerator PEs based on the affected output bit position. This technique protects cells related to the higher significant bits seriously and the protection strategy depends on the corresponding resilience level of the applications.	Micro	All	Tolerance	Design-time	Process/ Memory overhead	[104]
Dynamical Fixed Pointing	This method used a dynamic fixed-point for data representation to dynamically change the decimal point position considering the data range and minimizing the unused most significant bits (MSBs).	Micro	All	Tolerance	Design-time	Process/ Memory	[103]
Data type, Bit-width Optimization	It investigated a compromise between data type, bit-width reduction, and resilience evaluating the impact of permanent faults.	Micro	All	Tolerance	Design-time	Design-time overhead	[102]
Triplicating Sensitive Computations	An automated tool that evaluates the sensitivity of computations within DNN inference to output accuracy to be used in selective TMR. The tool triplicates the most sensitive computations to increase the functional resilience of the DNN accelerator without full TMR.	Micro	All	Removal	Design-time	Computing	[107]
Partial Reconfigure	Partial reconfiguration (PR) has been employed as a solution to deal with permanent faults. PR mechanisms detect the permanent fault in a module and perform the reconfiguration process.	Circuit	FPGA	Removal	Runtime	Hardware overhead	[108] [109]

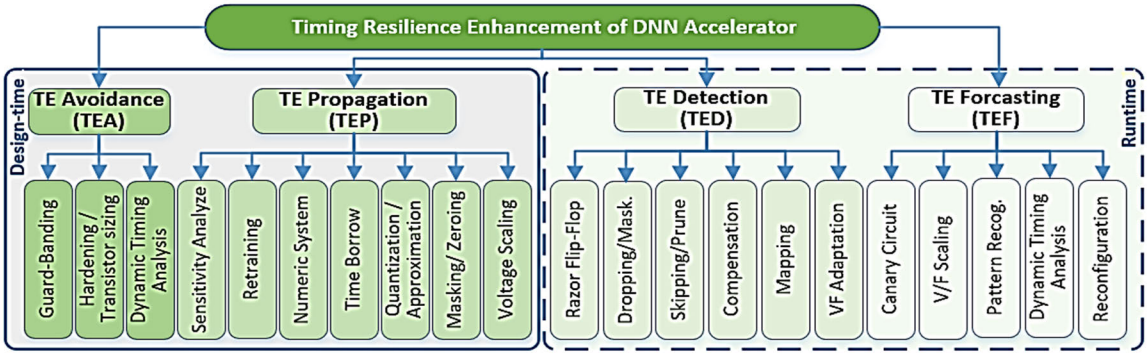

FIGURE 16. Techniques for timing resilience enhancement.

mechanisms are engaged to achieve a correct state. Herein, computational correctness is not achieved through worst-case guard bands but rather through in situ error detection and recovery. On the other hand, the common method for TEs prediction and forecasting the failure point at the circuit level relies on the Canary. Canary circuits are typically implemented by delay chains as an approximation of the critical path delay of the main circuit. They are employed to track the critical path delay across PVT corners and aging.

Fig. 16 illustrates the taxonomy and techniques proposed for TE handling in DNN accelerators. As seen in this figure, gaur-banding, transistor sizing, and dynamic timing analysis (DTA) are the three main methods of avoiding TEs in CMOS circuits, including DNN accelerators. The second class of TEP techniques includes sensitivity analysis, retraining, numeric system change, time borrowing, quantization and approximation, masking, and voltage scaling. Moreover, in 3rd category, Razor-based designing, TE dropping, TE pruning, TE skipping, TE compensation, TE-based mapping, and TE-based voltage scaling are placed. Finally, TEF contains methods such as Canary-based circuits, predictive voltage scaling, erroneous pattern detection, predictive timing speculation, and predictive reconfiguration. In the following, we explore major techniques in more detail.

A. TIMING ERROR AVOIDANCE

The study of [115] presents a design time method to alleviate NBTI-induced TEs in CMOS 6T-SRAM cells. The proposed method is based on nMOS access transistors sizing of SRAM cells to mitigate the NBTI effects in pMOS pull-up transistors and improve cell robustness. Accordingly, access transistors are sized to improve static hold noise margin under NBTI, other transistors of the 6T-SRAM cell could be properly sized for improving read stability or write-ability.

B. TIMING ERROR PROPAGATION

In this subsection, we review research done considering TEP in DNN accelerators. Herein, the first step is to evaluate TE probability and effects. Next, different techniques e.g., training, arithmetic system change, approximation, and operational parameters adaptation can be employed to mitigate TE effects and improve resilience. The study of [116] is one of the first works that has been done to analyze aging-induced TEs on performance degradation of DNN accelerator. In this study, the authors prepared a framework using the ASIC design flow to investigate degradation on MAC arrays based on library characterization by commercial synthesis tools.

According to this research, the classification accuracy of DNN accelerators running at peak throughput may drop to 84% after a year by degradation progress, versus throughput relaxation can compensate for the loss of accuracy considerably. Moreover, [70] explores two questions related to TE propagation: (a) What variations affect the accuracy of DNN output? and (b) Are there opportunities to optimize the DNN deployment to improve TE resilience The results of this study on MLP and CNN show that TEs can significantly affect inference accuracy. This paper presents a two phases cross-layer approach that first extracts timing faults from 20 operating conditions by hardware simulation, and next injects extracted faults back into the software simulation to answer the second question.

As mentioned before, if fault reduces the DNN output accuracy significantly, it will be better to update weights by retraining to avoid the critical situation. Thereby, a methodology for TE-aware training is proposed by [117] for ASIC DNN deployment, as shown in Fig. 17. Herein, the RTL description is synthesized into a gate-level netlist using synthesis tools and technology libraries to prepare placement and routing. Then, STA (static timing analysis) is conducted to provide the timing information. Next, the timing-aware gate-level simulation could discover the behavior of the design under a given working clock frequency. After that, joint forward TEs propagation and MSE monitoring are done on the datasets of real applications. For the selected dataset and working frequency, the current MSE is compared with the minimum. In the case of being smaller, the minimum MSE will be updated, and current accuracy and weights will be recorded. At last, the backward propagation algorithm will be used to generate modified weights to be fed back during retraining. Retraining repeats until the maximum epoch is reached.

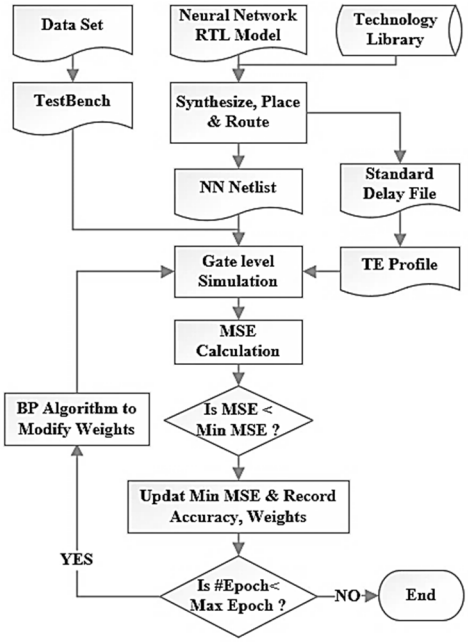

FIGURE 17. The overall flow of TE-aware training methodology (redesigned from [117]).

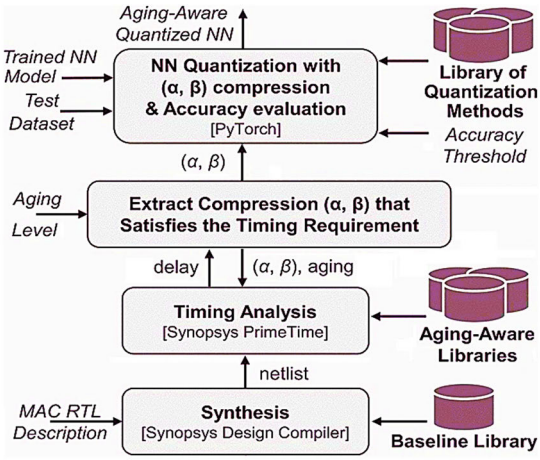

FIGURE 18. Aging-aware quantization in NPUs (Redesigned from [120]).

The conventional arithmetic system, i.e., parallel binary, does not fail gracefully in the face of aging or PVT variations when arriving beyond the conservative timing margins determined by STA. Since TEs are mainly occurring due to long carry propagation chains, it has less impact on online arithmetic than conventional ones. The study of [118] explores online arithmetic proposed for digit serial operation. It synthesizes digit parallel online operators to provide graceful degradation. This work shows that substantial performance benefits can be obtained by online arithmetic. Through the analytical exploration and experimental FPGA deployment, it claims that significant TE reduction and performance improvement can be achieved by online arithmetic in front of aging or variations.

Recently, approximate computing has been used to address aging and improve performance in error-tolerant applications [119]. The state-of-the-art study of [120] is one of the recent works conducted to mitigate TE effects using approximation. This study proposes a resilience-aware quantization based on an adaptive approximation to remove extra guard bands and mitigate aging effects in DNN accelerators. The proposed technique tries to compensate for the delay increase by managed quantization and provides a graceful accuracy degradation over the lifetime. The authors claimed that for the lifetime of 10 years, the average accuracy loss is merely 3% while achieving 23% higher performance due to the elimination of the guard band. Fig. 18 summarizes the proposed aging-aware quantization flow starting from the device level up (down in figure) to the model level where the inference accuracy is impacted. Here, α and β are the compression bit number for activations and weights.

Lastly, [114] presents a technique for TE handling in SRAM-based DNN accelerators using operating voltage scaling. The authors investigated fault propagation effects on the accuracy of a fully connected NN using different data types. They found that the weights are more sensitive to TEs rather than activations. Moreover, faults in the first layer have a larger effect than the last layer because the fault affects accumulation until it reaches the output. They proposed a technique to control the voltage for every memory operation with no latency penalty. It sets distinct voltage-level for SRAM cells storing weights of different layers, activations, outputs, and control bits. For example, a low voltage is set for input activations with more resilience.

C. TIMING ERROR DETECTION

In this subsection, we explore TEP-based research works conducted to enhance resilience. These works are mainly based on Razor-like flip-flops and use different knobs such as masking, dropping, mapping, and frequency scaling. It should be noted that some techniques are mainly proposed to mitigate TEs induced by voltage scaling of NTC. However, aging also has the same effects on delay and the proposed methods can be used to mitigate aging-induced TE effects too. The study of [33] proposes a method of minimizing the worst-case guard band by circuit-level timing violation detection. Based on the inherent error resilience of DNN, the authors used a simple technique instead of error correction through the replay. They proposed an accelerator processor with a five-stage pipeline equipped with Razor flip-flops to monitor the timing violation in two timing-critical stages: 1) the WMEM load and 2) the MAC execute units. Moreover, the authors reduced intermittent bit flips potential by two implicit techniques rather than explicit methods of error correction: 1) using the sign-magnitude number system and 2) use of time borrowing (TB) in data-path among pipeline stages. Considering small magnitude weights with random signs, the use of the sign-magnitude numbering system causes

switching activity reduction, eventually leading to a lower TE probability.

Word masking [121] and TE-Drop [113] aim to recover TEs on SRAM memory and PEs, respectively. We classify these two methods as zeroing since they reset the operation result to zero after error detection. A cross-layer optimization framework in pipelined DNN processor is presented by [121] which employs five means of training, microarchitecture exploration, quantization, ineffectual operation skipping, and SRAM voltage scaling in different layers of DNN implementation. The last stage of optimization introduces a TE mitigation technique, which allows for aggressive SRAM supply voltage reduction. The proposed technique first tries to detect TE location by Razor flip-flop. Next, it uses simple schemes to mitigate the fault effects by masking in two granularities, including word masking, in which all of the bits of a faulty word are put to zero, and bit-masking, in which only the faulty bit is replaced with the sign bit. Word masking removes the corresponding connection from the DNN by zeroing faulty PE's weight. In bit masking, the magnitude of a faulty weight is reduced by rounding a faulty bit position towards the sign bit. Fig. 19 illustrates modifications applied to the DNN processor microarchitecture in red.

A circuit-level technique of TE-drop for a systolic array accelerator is presented by [113]. This technique employs Razor flip-flops to detect TEs and consequently steals a cycle from the next PE to mitigate the error effects by dropping erroneous computations. TE-drop decreases the latency and power overhead compared with error correction. It uses a multiplexer to select between the previous layer's error-free partial sum in case of TE presence, or the usual partial sum computed by the MAC unit itself in case of TE absence, based on the error signal received from Razor shadow flip-flop. The TE-drop mitigates TE effects with less than 1% loss of accuracy and without performance loss. The only limitation will be when it is used for the last row of PEs which may affect performance and accuracy. In this case, the impact of TE is high because the values of partial sums are large at the bottom of the array. The final accuracy reduces significantly with increasing fault rate and skipping the update of some PEs.

To improve the timing resilience considering the sensitivity of neurons, [74] presents another cross-layer TED-based approach by skipping and mapping for a TPU-like systolic array accelerator. First, the sensitivity of a neuron is approximated by the product of the gradient and the fault rate. The authors calculated the gradient by differentiating and averaging over the target variable for the whole dataset. They claim that weights with small values have higher sensitivity, whereas those with larger values have less sensitivity against TEs. Then, the Razor flip-flop is used between the multiplier and the adder as an extra pipeline stage to measure the fault rate. In the case of TE, the multiplication output is ignored and the unmodified partial sum passes to the next stage. Here, the throughput is not affected, whereas the latency increases by a clock cycle. Finally, they proposed a mapping strategy to mitigate TE. The ideal case is to assign weights with higher sensitivity to more robust MAC units and vice versa, which requires large interconnect overhead and non-trivial changes to the hardware. The proposed mapping technique assigns filters with higher sensitivity to the MAC column having lower mean TE rates. Using the sensitivity of neurons provides better results than the TE-drop for the same accuracy loss.

The state-of-the-art study of [122] proposes a countermeasure to handle TEs utilizing shadow flip-flops for error detection and lightweight mechanisms for predictive error correction. The forward error compensation circuit corrects the error-inflicted partial sum by estimating the difference between the correct and incorrect results. The difference is added back to the final result at the next cycle without stalling the pipeline. Lastly, the forefront research of [123] proposes a technique to improve the efficiency of DNN accelerators with spatial architecture based on overclocking (timing speculation) and inherent error resilience. The authors presented an algorithmic-based lightweight TE detection mechanism to protect convolution layers, enabling aggressive timing speculation. They developed an algorithmic error detection mechanism using algebraic properties of the computation. Fig. 20 demonstrates the system-level overview of the proposed architecture, including the TE detection mechanism. After error detection, a recovery mechanism using dynamic frequency scaling adjusts the clock speed to prevent TEs.

D. TIMING ERROR FORECASTING

Here, we discuss the state-of-the-art works in timing resilience improvement based on error forecasting using Canary, erroneous pattern detection, and delay estimation. In the case of error prediction approximation, frequency, and voltage adjustment are employed to remove forecasted errors. In this regard, [119] presents a methodology to monitor the critical path delay at runtime according to TEF and turn the aging-induced timing violation into approximate computing error without the conservative guard bands or increasing the supply voltage level. The method was evaluated in two levels: RTL level and system level. The experimental results show a significant improvement in terms of MSE. The proposed approach successfully converts the aging-induced TEs into much less harmful approximate computing errors, and thus it improves quality up to perceptually acceptable levels.

A technique of TE mitigation in NTC (applicable for aging) is proposed by forefront research [124]. This research investigates the delay distribution of PEs by supplying all possible input combinations. The authors claimed that all multiplication operations activate paths with small delays except a few input patterns, which leads to TE. So, TE prediction is easy as they occur due to the same input sequences. The erroneous input patterns are highly likely to create TEs in the next layers, too, as long as remaining in the input. This technique predicts TEs based on the erroneous input pattern detection by adding a TE control unit (TECU) in each

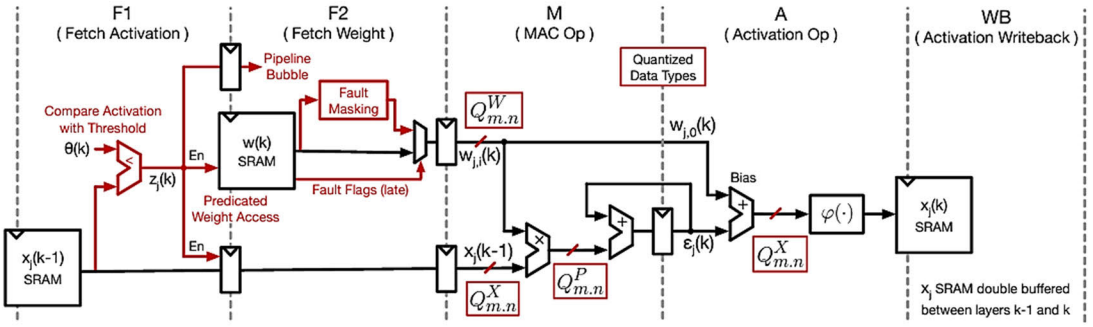

FIGURE 19. The microarchitecture of a single data-path lane (modified from [121]).

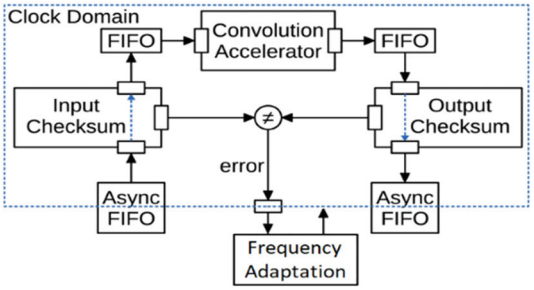

FIGURE 20. Block diagram of GreenTPU (re-illustrated from [124]).

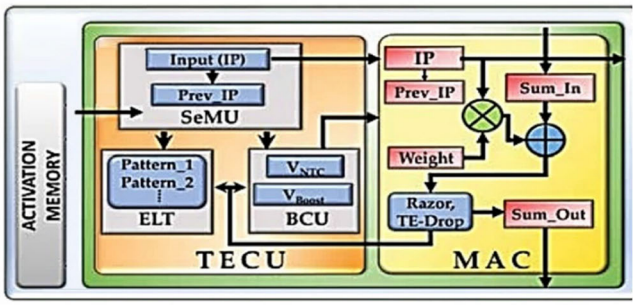

FIGURE 21. A system-level overview of timing speculation [123].

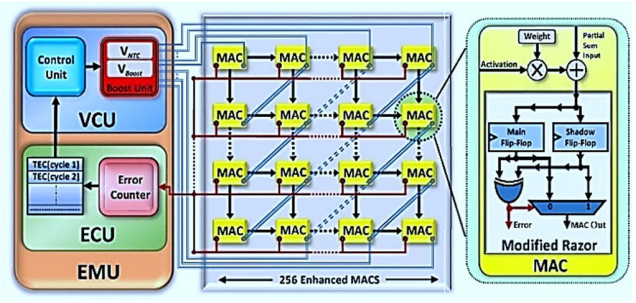

FIGURE 22. Voltage control, error management unit, and enhanced PEs of PREDITOR (modified from [126]).

row input, between the activation memory and the accelerator array to predict TE in PE before it occurs, as shown in Fig. 21. Razor-like flip-flop is used to detect TEs and record relevant input patterns. Eventually, the voltage of PEs increases in the face of recorded patterns to prevent further TEs.

In [125], the authors proposed an approach based on TEF for fast and accurate timing evaluation (FATE) of DNN accelerators, e.g., Google TPU. FATE has two novelties: (i) DelayNet, a DNN-based timing model for timing estimation of MAC units for different inputs; and (ii) a statistical sampling method to reduce the number of timing estimations. They reached 8× to 58× speed-up using FATE for timing simulations, while less than 2% error in accuracy. DelayNet is trained using main DNN inputs profiling and TEs detected by the Razor mechanism. Also, [69] presents a technique based on TEF, named MATIC (adaptive memory training with in-situ canaries), that mitigates TEs occur in weight buffer due to aggressive voltage scaling (applicable for aging). MATIC employs training and voltage adaptation to mitigate TEs. First, it creates an SRAM fault map by performing the read-after-write and read-after-read operations on different SRAM addresses to find the preferred state of bit cells. It uses fault-aware training using the fault map to allow CNN to compensate for the TEs. Next, the in-situ canary circuit is used in weight SRAM to predict TE and dynamically control voltage.

The state-of-the-art study of [126] proposes another TEF-based technique, namely PREDITOR as a low-power TPU operating in the NTC (can tolerate aging effects too). PREDITOR uses mathematical analysis to mitigate the TEs by boosting the voltage of selective PEs at specific intervals to enhance the performance, thereby ensuring a high inference accuracy at low voltage. PREDITOR includes an error collection unit (ECU), a 16-bit error counter and a content-addressable memory, and a voltage control unit (VCU) as shown in Fig. 22. Herein, PEs of the systolic array is enhanced with a modified Razor flip-flop. VCU is composed of the control unit (CU) and the boost unit (BU) itself. CU computes the range of operational clock frequency based on analyzing the TE control in ECU and voltage boosting. BU boosts the voltage of PEs for the clock cycle interval predicted by the CU. ECU collects and stores the TE count from each cycle. The voltage boosting aids in mitigating both detectable and undetectable TEs. They proposed a cross-layer method to evaluate the proposed design across DNN applications.

Finally, for the FPGA deployment of the DNN accelerator, continuous runtime monitoring of on-chip delays at the circuit level allows us to detect and adapt for TEs induced by aging. A TEF-based method to monitor transitions and estimate the remaining timing margin is presented by [127]. The proposed method identifies the individual degraded paths with the worst-case delay using a lightweight mechanism of DDFFL (difference detector with first-fail latch). Combined with fine-grained repair alternatives like partial reconfiguration, the authors introduced a runtime and in-system mechanism for incremental repair of links with timing degradation.

Table 7 presents the summary and features of proposed techniques for timing resilience enhancement according to the mentioned categories marked with colors similar to Fig. 16.

VI. AGING-AWARE LIFETIME RESILIENCE

In this section, the last aspect of resilience related to conserving or even extending the lifetime of the DNN accelerator is discussed. We explore prior works in two periods of accelerator life, including before wear-out and wear-out. Referring to the bathtub curve mentioned in Subsection II-B (Fig. 3), the first period includes early life and normal life phases. Herein, lifetime resilience can be conserved by monitoring and managing effective factors according to the lifetime-expressing models. In the wear-out phase, the accelerator is aged, and different permanent or timing faults occur in PEs or MEs due to aging progress. All of the techniques related to functional and timing resilience enhancement, which we have already discussed in Sections V and VI, could be employed to extend the wear-out phase. In the following, we review techniques presented to extend the first period. Fig. 23 demonstrates a taxonomy of techniques for lifetime resilience improvement categorized by appropriate time to apply and affecting factors. According to the figure, there are two main approaches to express lifetime: 1) threshold voltage-based and 2) MTTF-based. In the following, we survey research done in these two disciplines, including voltage scaling, dual V_{th} assignment, body biasing, power gating, workload monitoring, mapping, dynamic thermal, and reliability management.

As mentioned before, NBTI is the dominant aging phenomenon among aging mechanisms, i.e., NBTI, HCI, TDDB, and EM. Thus, we mainly study previous research related to lifetime resilience considering NBTI effects. Although, in general, proposed methods for NBTI-aware lifetime extension can mitigate the effects of other aging mechanisms too.

A. VTH-BASED LIFETIME EXTENSION

As we have already stated, NBTI leads to the threshold voltage (V_{th}) shift in PMOS transistors which increases the critical path delays and eventually causes timing violations or TE. NBTI can be understood in terms of two different phenomena: stress and recovery. NBTI occurs when the PMOS transistors are under stress with negative voltage bias ($V_{GS} = -V_{dd}$). The V_{th} change is partially recovered after the removal of the negative bias. As it undergoes repeated stress and recovery, the V_{th} of the device is gradually increased. The long-term NBTI *degradation* model based on reaction-diffusion (RD) theory estimates ΔV_{th} as follows [128]:

$$\Delta V_{th} = A \cdot f\,(V_{dd}, V_{th}, T, R)\,.t^{n} \tag{2}$$

A denotes the aging factor, which reflects the actual stress/recovery pattern along with all technological and operational parameters. f is a function of voltage (V_{dd}), V_{th}, temperature (T), and R as an indication of all device parameters. According to this model, the delay degradation due to the V_{th} shift can be calculated by the alpha power model [129]:

$$d = \frac{J}{(V_{dd} - V_{th0})^{\alpha}} \times d_0 \tag{3}$$

According to this model, the lifetime is defined as a period in which ΔV_{th} reaches the critical fraction P_{crt} of the initial value ($\Delta V_{th}/V_{th} \leq P_{crt}$). In practice, P_{crt} is typically defined as 10% [130]. The amount of degradation will be diverse for a chip that runs different workloads and experiences changing operating conditions. For example, high utilization of MAC units in a DNN accelerator [131] causes faster aging of constituent transistors because of continuous stress without enough time for recovery. Moreover, higher operating temperature worsens the degradation as the majority of aging mechanisms exponentially depend on [132]. In the following, we survey the works conducted to mitigate the gradual progression of degradation by managing contributing factors, i.e., temperature, stress, and supply voltage.

1) THERMAL MANAGEMENT

In previous studies, thermal issues have been explored, while thermal management is still an active research area. In advanced CMOS technologies, higher chip temperature accelerates transistor aging. High temperature incurs resilience challenges since all aging mechanisms, e.g., NBTI, are exponentially dependent on temperature. On the other hand, power management techniques can help to control temperature by reducing power density. Although, power dissipation reduction is not always effective alone and may conflict with thermal management when they turn off underutilized parts and concentrate activities in a smaller area, which may increase power density. Here, we classify techniques of thermal management into six categories; liquid cooling, floor planning, dynamic thermal management, thermal-aware mapping, temperature monitoring, and microarchitecture modification.

a: LIQUID COOLING

Liquid cooling technique is an alternative to traditional air cooling. Many high-performance MPSoCs use air cooling due to simplicity and cheapness. However, as the thermal issue became more important, other cooling techniques have been investigated. Water is an emerging coolant for its high heat capacity. The study of [133] introduces an indirect liquid cooling technique in stacked integrated circuits using

TABLE 7. Timing resilience evaluation and improvement techniques and their key characteristics.

Technique	Summary	Scope	Layer	Platform Hardware/ Software	Element	Measure	Design-Time/ Runtime	Cost, Dependency	Works Done
Aging-aware Vulnerability Evaluation	It evaluates the aging effect on performance degradation of DNN accelerators. A framework for TE analysis using ASIC design flow was presented. First, TEs are extracted. Next, extracted TEs, are injected into the forward computation of DNN to explore the accuracy.	TEA/ TEP	Circuit	ASIC	PE	ASIC Design Flow	Design-time	Design-time overhead	[70] [116]
Training	In this technique, DNN model training is performed in the presence of TEs. For this, we should prepare a fault map by post-synthesis design or post-fabricated chip to inject during the DNN model training.	TEA	Circuit / Model	ASIC/ FPGA	ME	In-situ Canary	Design-time	Design modification /	[69][117] [95]
Time borrowing	MSE is used to limit the worst-case magnitude of a logic error. Instead of error correction through replay, this technique uses a cheaper error mitigation technique by time borrowing (TB) in the data path among pipeline stages to reduce the TE effects.	TEP/ TED	Circuit	CPU / GPU	PE / ME	Razor flip-flops	Design-time	Design modification	[33]
Arithmetic System Change	In this method, the impact of timing violation on online arithmetic primitives is quantified and showed that substantial performance benefits can be obtained in comparison to conventional binary arithmetic. Since TEs are caused by long carry chains, these result in errors in the least significant digits with online arithmetic, causing less impact than conventional implementations.	TEP	Architecture	All	PE	Online Arithmetic	Design-time	Daley	[118]
Voltage Scaling	This technique controls the voltage for every memory operation with no latency penalty. It sets different voltage levels for SRAM cells storing weights of different layers, activations, outputs, and control bits.	TEP	Architecture /Circuit	All	ME	DVS	Design-time	Design-time optimization	[114]
Retraining	This method partially retrains the model to update their weights when TEs significantly affect the output results used to circumvent critical TEs.	TEP	Model	ASIC	PE	ASIC Design Flow	Design-time	Retraining overhead	[65] [117] [69][117]
Masking / Skipping	In this technique, erroneous neurons or neurons with high sensitivity are pruned or skipped based on active hardware fault detection coupled with domain-aware error mitigation.	TEP/ TED	Model / Architecture	ASIC/ FPGA	ME	Razor flip-flops	Design-time	Hardware modification	[69][117] [121]
Dropping	TE-drop uses Razor flip-flops to detect TEs and steals the subsequent clock cycle from its downstream MAC to correctly finish its update to the partial sum while skipping the update of the downstream MAC. This method mitigates the latency and power overhead induced by TED.	TED	Circuit	ASIC/ FPGA	PE	Razor flip-flops	Runtime	Hardware modification	[113]
Sensitivity based Mapping	This technique rapidly evaluates filter/channel-level weight sensitivities of large-scale DNNs. With measured TE probability of each MAC unit considering the sensitivity variation among filter weights can be leveraged to design a DNN accelerator, such that the computations with more sensitive weights are assigned to robust MAC units, while weights with less sensitive are assigned to less robust MAC units.	TED	Circuit / Algorithm	ASIC/ FPGA	PE	Razor flip-flops	Runtime	Sensitivity analysis / Hardware modification	[74]
Fault-Aware Pruning	Two novel strategies, fault-aware pruning (FAP) and fault-aware pruning + retraining (FAP+T) enable the TPU to operate at fault rates of up to 50%, with negligible drop in classification accuracy (as low as 0:1%) and no run-time performance overhead.	TED	Architecture / Model	ASIC/ FPGA	PE	---	Runtime	Hardware modification	[95]
Lightweight Error Correction	This technique determines the incorrect partial sum of the MAC operation comparing the correct and error-inflicted computation. The difference is added back to the final result at a later cycle without stalling the execution pipeline.	TED	Circuit / Algorithm	FPGA/ ASIC	PE	Razor flip-flops	Runtime	HW modification, Power overhead	[122]
ABFT	This method augments the accelerator with a lightweight error detection mechanism to protect against TEs in convolution layers, enabling aggressive timing speculation or aging effect tolerance. The error detection mechanism developed at the algorithm level, utilizing algebraic properties of the computation	TED	Circuit / Algorithm	FPGA	PE	ABFT	Runtime	Computation redundancy	[123]
Adaptive / Selective Voltage Scaling	To ensure a high inference accuracy in TE presence, it identifies the patterns of the error-causing activator sequences and prevents further TEs by intermittently boosting the voltage of the specific MAC units at specific intervals. Mathematical analysis or Sequence Monitor Unit (SMU) is used to identify the possibility of TE. The input activation data, coming to each row, is intercepted by SMU.	TEF	Cross-Layer	ASIC/ FPGA	PE	SMU / Razor	Runtime	HW modification, Power overhead	[124] [126]
Frequency Speculation	In this technique, a simple (NN) model is prepared in the design time to predict the timing requirement of DNN according to samples of input data patterns. Then it uses timing prediction to prevent TE by adjusting the frequency of DNN.	TEF	Model / Circuit	ASIC	PE	ASIC Design Flow	Runtime	Prediction overhead	[123] [125]
Adaptive Partial Reconfigure	A method to continuously monitor path delays in an operational FPGA design and to improve slow paths by incremental partial reconfiguration. Since online delay measuring is more accurate than design time estimation, this approach allows to balance delays which can be used to improve performance or reduce power consumption. In addition, it counteracts aging effects and prolongs the system's useful lifetime	TEF	Circuit / Architecture	FPGA	PE	Razor flip-flops	Runtime	Runtime reconfiguration overhead	[127] [47]

microchannels between layers. Contrary to the indirect technique, [134] introduces a direct liquid cooling approach for the 3D integrated chip. It uses a water tank, inlet, and outlet. Water flows through silicon vias (TSV) between the dies.

b: FLOOR PLANNING

Floor planning research includes both CAD and microarchitectural works and covers thermal-aware floor planning techniques for 2D and 3D chips. Already, floor planning is used for performance improvement and energy reduction by reducing wire length. But it can also mitigate the temperature considering increasing power density in emergent circuits. Thereby, a tool, namely HotFloorplan, for temperature-aware floor planning is developed [135]. It proposed a floor planning algorithm using peak steady-state temperature, chip area, wire delays, and simulated annealing.

c: DYNAMIC THERMAL MANAGEMENT (DTM)

One representative research area of thermal management is DTM which was followed by some researchers trying to decrease the temperature by low-power techniques, e.g., DVFS. Such techniques reduce performance and may not alleviate the growing severity of thermal stress at the rated performance. Considering proposed DTM techniques for high-performance processors [136], there is an initial delay before DTM activation when the temperature of the chip reaches the predefined trigger. After the DTM is engaged, the processor checks the temperature periodically. When the

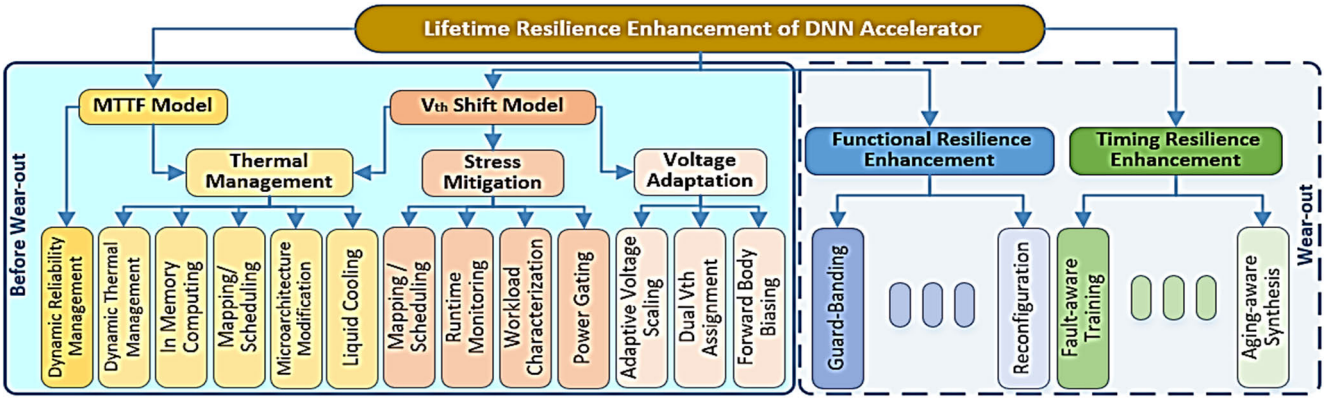

FIGURE 23. Lifetime resilience improving taxonomy.

temperature drops below the DTM trigger, the DTM is disengaged, and the processor runs normally again. Deactivation may also have some delay. The proposed DTM mechanism is based on voltage/frequency scaling and rate throttling. The study of [137] proposes an online learning algorithm to evaluate DTM on MPSoCs, formulating a loss function. The loss function includes four factors: hotspots, thermal cycles, spatial gradients, and performance.

d: THERMAL-AWARE MAPPING/SCHEDULING

This subsection discusses prior works related to temperature management by task assignment and scheduling. In the conventional priority queues, tasks are sorted according to their importance. The proposed technique in [138] adds the thermal features to the priority queues; their policy gives a lower priority to the hot tasks (tasks may exceed the predefined temperature threshold due to high power consumption). The control system monitors tasks at runtime through the performance counter values to detect hot tasks. If all tasks are hot, a thermal emergency can be avoided with hardware-supported clock gating as a failsafe mechanism. The scheduling technique proposed by [139] is based on temporal thermal correlations of tasks, while thermal-aware task scheduling usually uses spatial correlations among processing elements to balance the workloads. Moreover, this technique focuses on choosing the appropriate threads to keep the processor temperature below the threshold. For example, with two hot and cool tasks waiting in the run queue, the hot-cool sequence causes less temperature increase compared to the cool-hot ones.

The study of [140] introduces a technique to address thermal issues of hybrid NNs execution on a DNN accelerator with 3D+2.5D stacking. In this design, the PE array can be flexibly partitioned into two portions to accelerate ConvNet and sparse FC computations. This method decreases memory access by data reuse and peak access reduction. The complementary memory access patterns of different networks are mixed to reduce both steady-state and peak temperature without performance loss (Fig. 24). The key idea is to avoid

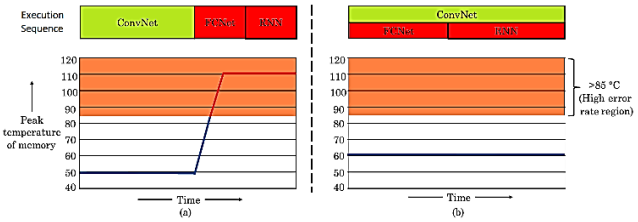

FIGURE 24. (a) Sequential execution of ConvNet and FCNet/RNN leads to high memory temperature (b) parallel execution presents a uniform temperature profile (modified of [140] [10]).

burst memory access using spatial division mapping (SDM). SDM executes ConvNet and sparse FC in parallel and reduces the peak memory bandwidth demand based on the bandwidth difference of CONV and sparse FC. For ConvNet, it uses a batch size of one, whereas, for RNN and FCNet, a larger batch size is used. The introduced compiler allocates PE array and SPM buffer between ConvNet and RNN/FCNet to minimize overall inference latency and peak bandwidth demand.

High-bandwidth memory (HBM) improves memory bandwidth based on vertically stacked memory architecture and through-silicon via (TSV) fast interconnect. But the stacked architecture increases power density leading to thermal issues when running modern memory-hungry DNNs. Prior research on the DTM of 3-D DRAM ignored the physical structure of HBM and caused heavy DTM performance overhead. The state-of-the-art study [141] proposes an application-aware task mapping technique to map DNN to PEs based on exploiting the HBM channel layout and temperature gradient across DRAM dies. The proposed method considers variations in the memory access of DNN layers and tries to minimize stalling due to thermal hotspots in the HBM stack. It also uses application-aware DVFS and DRAM low-power states to further improve performance. NeuroMap includes two main elements: 1) Mapper and 2) DTM Engine. The Mapper dynamically conducts an efficient task-to-core mapping, migration, swapping, and workload-aware DVFS, depending on the type of task and memory footprint of the current

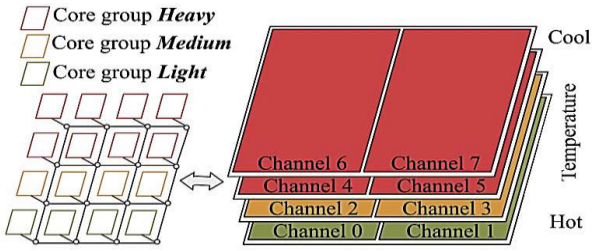

FIGURE 25. Static core to channel mapping to minimize vertical stacking of thermal hotspots (redesigned from [141]).

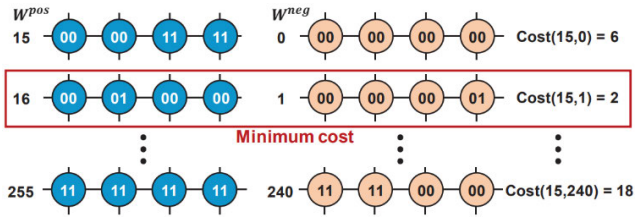

FIGURE 26. Thermal-aware weight decomposition (modified of [148]).

executing layer and the data reuse policy. The DTM Engine periodically monitors the temperatures of the memory channels and detects the proper power state for each rank to be applied by the memory controller. NeuroMap considers a layer as a task and classifies the DNN layers into three categories: 1) heavy; 2) medium; and 3) light, based on their memory access rates. The task mapping assigns cores to tasks where a light layer is mapped to a core in the bottommost row of the 4 × 4 grid, a heavy layer is mapped to a core in the topmost row, and a medium layer is mapped to a core in the middle rows. Fig. 25 illustrates core-to-channel mapping introduced by NeuroMap. The workload knowledge helps to limit the memory traffic to HBM channels in a way that mitigates the DTM penalty by minimizing the heating of the bottommost channels.

e: TEMPERATURE MONITORING

Temperature estimation and sensor allocation techniques are important since DTM uses the results as feedback to control the temperature or allow higher performance, as appropriate. Typically, there are two types of digital and analog thermal sensors which they mainly employed to detect localized hotspots or to read on-die temperatures [142]. In a sensor-based approach, we have to deploy many sensors to measure temperature accurately. An efficient allocation of the limited number of sensors is conducted by [143] to reduce the cost. In addition to hardware sensors, there have been studies on thermal estimation models with performance counter and software techniques to minimize hardware overhead [144]. Herein, model-based temperature estimation is presented, which may be more robust than a sensor that is vulnerable to noise or process variation.

f: MICROARCHITECTURE MODIFICATION

Here, we review micro-architectural methods of thermal management for MEs and PEs. A thermal management technique for simultaneous multi-threading processors is introduced by [145] which adjusts the instruction fetch policy of conventional architecture. The proposed thread selection mechanism chooses the coolest thread by looking at the profiled register file access frequency when the integer register file or the floating-point register file is overheated. The other research [146] proposes a register file bank-switching technique, given that a smaller register file leads to little performance loss. The proposed technique divides the register file into two banks, primary and secondary. Banks are activated periodically, thus halving the power density of the register file and eventually leading to better thermal management. Another work for a specific functional unit is the O2C (occasional two-cycle operation) proposed by [147] which applies O2C to the adder and the multiplier. When thermal sensors detect the processor overheating, VDDL (the lower VDD) is supplied to the execution pipeline stage instead of VDDH (the nominal VDD). Consequently, it leads to an increase in the execution stage latency to two cycles (originally one cycle) due to voltage reduction.

In the ReRAM-based DNN accelerator, thermal effects reduce the lifetime and inference accuracy. To address this, [148] proposes state-of-the-art temperature optimization methods, including three offline temperature optimization and one online error compensation. Offline steps for temperature mitigation include thermal-aware weight decomposition, thermal-aware column reordering, and fine-grained weight adjustment. Fig. 26 illustrates examples of thermal-aware weight decomposition, which selects a presentation of weights with minimum cost. The proposed online method compensates for errors that occur due to temperature variation based on the mirror circuit adaptation, which illustrates reliable performance regardless of temperature change.

DNN accelerator includes thousands of PEs requiring high energy consumption. To decrease energy consumption and consequently reduce the temperature, approximate computing is employed, while complex DNNs can be sensitive to approximation. To satisfy tight temperature constraints, thermal-aware design approximation of a DNN accelerator based on trading-off approximation with temperature effects. Another state-of-the-art study [149] demonstrates how approximation reduces the temperature of the accurate circuit from 139°C down to 79°C. The authors claimed that it enables DNN accelerators to fulfill thermal constraints, improve performance and decrease energy consumption with an accuracy loss of about 0.44% on average. Moreover, they explored reliability improvement from the aging perspective due to the reduced voltage and temperature of the approximate design. According to the presented experimental results, the approximate design exhibits 40% less aging degradation compared to the baseline (Fig. 27). The evaluation is first done in the circuit-level full-chip deployment and is expanded

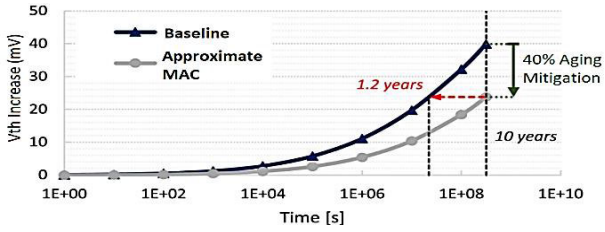

FIGURE 27. V_{th} increase due to BTI and HCI (re-illustrated of [149]).

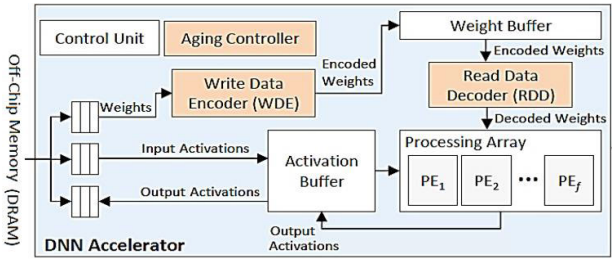

FIGURE 28. The proposed architecture (redesigned from [154]).

up to the system level to evaluate the accuracy and latency of inference. They achieved significantly higher throughput by approximate MAC arrays compared to the baseline. They reduce the voltage of the approximate MAC arrays from 0.7V to 0.6V, leveraging throughput gain to mitigate power consumption.

Another state-of-the-art research [150] proposes a configurable approximate multiplier with three modes of operation, i.e., exact, positive error, and negative error. Also, it proposes a filter-oriented approximation approach to map the weights to modes of multipliers. The mapping algorithm balances the positive error and the negative error of approximation to maximize the energy reduction while minimizing the overall approximation error. Finally, to tackle DVFS for power reduction, the forefront work of [151] introduces a new control knob based on the size of input batches fed to the DNN inference in the GPU. The authors first analyzed the effects of batch size on power and performance. Then, they designed and deployed a fast and lightweight runtime system called Batch DVFS for dynamic batching by adaptive changing batch size to trade-off throughput with power consumption. They used binary search to find the proper batch size in a short period. Batch DVFS Combines dynamic batching with DVFS to control power in wider ranges while conserving throughput in the presence of power caps.

2) STRESS MITIGATION

Based on the reaction-diffusion model, aging in terms of V_{th} shift is dependent on workload-induced stress. Along with other effective parameters, i.e., time, temperature, and voltage, [152] takes the workload features into account. The workload features include signal probability (SP) and transition density. The SP is the ratio of the number of periods with logic one at a gate input to total clock periods, while the probability of 0's determines the NBTI-induced degradation in PMOS devices. Thus, SP and 1-SP are important to cover both NBTI and PBTI in aging estimation models. These factors are given to the model to estimate the gate delay degradation. Here, we review prior works that try to alleviate aging in accelerator PEs or MEs by stress reduction. There are four main approaches for stress mitigation, including power gating, workload characterization, runtime monitoring, and mapping/scheduling.

Neuromorphic computing using non-volatile memory (NVM) can improve performance and reduce energy consumption based on spike-based computations and bio-inspired learning algorithms. Although, high voltages of NVMs operations can accelerate aging in PEs leading to the lifetime reduction of neuromorphic hardware. The state-of-the-art study of [153] analyzes the long-term impact of DNN execution on a neuromorphic accelerator, considering different aging mechanisms, e. g. NBTI. To mitigate aging, it proposes the resilience-performance trade-off based on periodic relaxation of neuromorphic circuits, i.e., a stop-and-go pattern and de-stressing of all PEs at fixed intervals.

Due to the NBTI effect on SRAM weight memory of DNN accelerators, another state-of-the-art work [154], introduces the DNN-Life framework for aging mitigation by stress reduction in DNN accelerators. The authors, first, evaluated different DNN quantization effects on the distribution of the bits of weight values at the algorithm level. Next, using the insights of this evaluation, they introduced a microarchitecture to employ low-cost memory write (and read) transducers to achieve an optimal SP at runtime, thereby balancing the aging of the weight memory cells. Transducers periodically change the coding of weight values by bit-flipping randomization to store them in the SRAM buffer with balanced SP. Weights are decoded back to the original value for feeding the PEs. As a result, the DNN-Life framework enables efficient aging mitigation of weight memory of DNN hardware at minimal energy overhead during the inference process. The main drawback of this work is the overhead of data encoding and decoding. Fig. 28 demonstrates the introduced architecture and microarchitecture of DNN life, respectively. The highlighted boxes are used to mitigate NBTI.

Also, because of the vulnerability and continuous degrading of SRAM storage due to NBTI wear-out effects, the state-of-the-art research of [155] is concentrated on aging mitigation of on-chip SRAM activation buffers of DNN accelerator mitigating the stress. It is aimed at minimizing both NBTI and HCI aging effects. The authors quantified aging degradation induced by performing different DNN inferences considering factors of duty cycle (or SP), transition density, and access patterns in activation memory cells. Using the insights gained from this study, this work proposes a micro-architectural technique, namely Gated-CNN, to ensure a uniform degradation of memory cells. The key idea of Gated-CNN is to joint use of bank address rotation and bank power-gating mechanisms to balance transition density, access patterns,

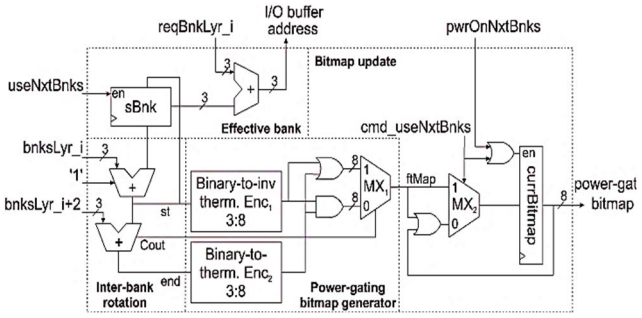

FIGURE 29. Gated-CNN main modules (modified from [155]).

and switch-off cycles for uniform duty cycle distributions across all the memory banks. Gated-CNN includes four main sequential modules of effective bank module, inter-bank rotation module, power-gating bitmap generator module, and bitmap update module. The first module calculates the bank id where a requested activation is found in the I/O buffer. The 2nd module identifies the starting and ending banks needed by the next inference. The 3rd module generates a bitmap that shows banks that should be powered on/off in the next inference step. The last module updates the bitmap of power-gating at runtime to mask the bank wake-up latency penalty. The main advantage of Gated-CNN is leveraging the workload features, e.g., the memory size of activation layers, temporal and spatial localities, and I/O buffers that alternatively exchange input and output roles. The main drawback of Gated-CNN is that power gating can cause thermal cycling problems. Fig. 29 demonstrates the Gated-CNN modules and architecture.

3) VOLTAGE ADAPTATION

Herein, we review proposed techniques of lifetime resilience improvement based on operating voltage adaptation. In this regard, [156] introduces a technique to hide the effects of aging and slow degradation down based on *adaptive supply voltage* (ASV) and *adaptive body biasing* (ABB). According to the literature, ASV and ABB techniques can be used to mitigate aging effects [157]. In ASV, the chip's V_{dd} is slightly changed around the nominal value. Under ASV+, the gates become faster and spend more dynamic and static power and vice versa. In ABB, a voltage is applied between the chip's substrate and the source (or drain) of the transistors, which requires addition of on-chip signal lines for the bias voltage. It either decreases the V_{th}(FBB or forward BB) or increases it (RBB or reverse BB) depending on the voltage polarity. Under FBB, the gates become faster and consume more leakage power and vice versa. The proposed framework, namely Facelift, hides aging through aging-driven application scheduling. It mitigates aging by applying voltage adaptation at key times. It employs a non-linear optimization algorithm to balance the effects of voltage adaptations on the aging rate and the critical path delays. Moreover, Facelift can gainfully configure the chip for a short service life while improving performance. Accordingly, by hiding and slowing down aging, designers can design chips for 7-year life while running at a 15% higher frequency or design for 5 to 7 months of service life and still use it for seven years.

Moreover, [158] proposes *scheduled voltage scaling*, a technique to gradually increase the operating voltage of the chip to compensate for the NBTI degradation instead of setting a fixed operating voltage guard band to rectify aging. In scheduled voltage scaling, the voltage increases gradually at runtime and has the potential to increase IC lifetime by about 45%. In the state-of-the-art study of [159], a statistical optimization framework is presented to improve the lifetime resilience of digital circuits in the presence of aging degradation based on the gate-level delay degradation model. Here, to estimate a criticality metric, a set of statistically optimized critical gates is selected. The *dual-threshold voltage assignment* technique is used for the identified critical gates enabling the target chip to improve lifetime resilience and reduce timing yield loss.

B. MTTF-BASED LIFETIME EXTENSION

Lifetime can also be expressed by the measure of mean time to failure (MTTF). Here, NBTI can be modeled as follows:

$$MTTF_{NBTI} = \left\{ \left[\ln\left(\frac{A}{1+2e^{B/kT}}\right) - \ln\left(\frac{A}{1+2e^{B/kT}} - C\right) \right] \times \frac{T}{e^{-D/kT}} \right\}^{1/\beta} \quad (4)$$

where A, B, C, D, and β represent appropriate parameters, and k represents Boltzmann's constant. This lifetime reliability model was built on experiments done at IBM [160]. Herein, the temperature is the main parameter affecting the lifetime in the long term. The critical point in this model is that the MTTF becomes smaller as the temperature increases. Another important implication of this model is that the lifetime resilience of CMP (chip multiprocessor) depends on the running workload because of the temperature dependency. An MTTF-based mapping approach is proposed by [156] to mitigate aging effects. Assuming the same clock frequency, which is determined by the slowest core for all PEs of CMP, the proposed technique maps relatively cool tasks to the slower cores, where the pipeline slack margin is tighter compared to the faster cores and vice versa. Consequently, aging distributes across the cores uniformly. Here, the average temperature of each task is measured using per-core sensors.

1) DYNAMIC RELIABILITY MANAGEMENT

The idea of dynamic reliability management (DRM) is to monitor system elements continuously and make periodic decisions for control knobs to shift operations to a state where the lifetime is as close as possible to a desired predefined value. DRM has two components, the MTTF online estimator and the DRM controller. DRM usually employs task mapping and DVFS techniques to improve the lifetime resilience of the desired target. In [161] a dynamic reliability management (DRM) technique was presented based on the MTTF

TABLE 8. Lifetime resilience evaluation and improvement techniques and their key characteristics.

Measure	Improvement Technique	Summary	Layer	Platform Hardware/ Software	Scope	Design-Time/ Runtime	Cost / Depend-ency	Works Done
MTTF	MTTF Mapping	The proposed technique maps relatively cool tasks to the slower cores, where the pipeline slack margin is tighter compared to the faster cores, and vice versa.	Architec-ture	PE/ME	Forecast	Runtime	Runtime overhead	[156]
MTTF	DRM	A management algorithm for hybrid DRM combines thread migration and DVFS.	Architec-ture	PE/ME	Removal	Runtime	Runtime overhead	[162]
Temperature	Liquid Cooling	A direct or indirect liquid cooling technique for temperature management.	Device	2D/3D Chip	Prevention	Design-time	Design and Fab cost	[133] [134]
Temperature	Floor Planning	A floor planning algorithm based on peak steady-state temperature, chip area, and wire delay using classical methods like simulated annealing.	Device	ASIC /FPGA	Prevention	Design-time	Design overhead	[135]
Temperature	DTM	A mechanism for thermal management based on voltage/frequency scaling, rate throttling, online learning, and balancing by control theory.	Cross-layer	All	Removal/ Forecast	Runtime	Perfor-mance overhead	[136] [137]
Temperature	Mapping/ Scheduling	Adds the thermal-aware features to the conventional priority queues, e.g., gives a lower priority to the hot tasks. The PE array can be flexibly partitioned into several portions to accelerate different DNNs.	Architec-ture	All PE	Tolerance	Runtime	Runtime overhead	[138] [139] [140] [141]
Temperature	Memory Access Reduction	This method decreases memory access by data reuse and peak memory accesses reduction using different batch size.	Architec-ture	All ME	Tolerance	Design-time	Design overhead	[140]
Temperature	Estimation / Prediction	A model-based approach for temperature estimation and predictive management using the performance counter.	Architec-ture	All PE	Forecast	Runtime	Runtime overhead	[144]
Temperature	T-Sensor Allocation	efficient allocation of the limited number of thermal sensors to reduce the cost.	Circuit / Device	All PE	Removal/ Forecast	Design-time	Design overhead	[143]
Temperature	Voltage Scaling	When thermal sensors detect overheating, VDDL is supplied to the execution pipeline stage instead of VDDH increasing the execution stage latency to two cycles.	Micro	ASIC /FPGA	Removal	Runtime	Runtime overhead	[147]
Temperature	weight decom-pose /column reordering	A thermal-aware optimization approach for ReRAM-based DNN accelerator namely TOPAR.	Model-Architec-ture	All PE	Tolerance	Design-time	Design overhead	[148]
Temperature	Error compen-sates by mirror circuit	Thermal-aware error compensation by adopting a current mirror circuit, which shows reliable performance on analog multiplication regardless of temperature variation.	Circuit	All PE	Removal/ Forecast	Runtime	Extra Area overhead	[148]
Temperature	Approximation	An approximate multiplier (dynamic configurable) for the thermal-aware design of the DNN accelerator based on additional trading-off approximation with temperature effects.	Micro	All PE	Tolerance	Design-time	Design overhead	[149] [150]
Temperature	DVFS / Dynam-ic Batch Size	Combines dynamic batching with DVFS to control power in wider ranges while conserving throughput.	Architec-ture/ Circuit	All PE	Removal/ Tolerance	Runtime	Runtime overhead	[151]
Stress	Periodic Relax-ation	It explored the resilience-performance trade-off by periodic relaxation of neuromor-phic circuits, i.e., a stop-and-go pattern and de-stressing all PEs at fixed intervals	Micro	PE	Forecast	Runtime	Stalling overhead	[153]
Stress	Quantization / Random Bit Generation	It analyzed the effect of quantization on the distribution of the bits of weights and proposed a micro-architecture that employs low-cost memory-write transducers to achieve an optimal duty cycle at runtime in the weight memory cells.	Micro	SRAM Memory	Forecast	Design-time	Area and power overhead	[154]
Stress	Storage Rota-tion / Power Gating	To mitigate the aging of SRAM memory in the CNN accelerator, a microarchitectural technique for uniform aging combines bank address rotation with power gating.	Architec-ture / Micro	SRAM Memory	Forecast	Runtime	Runtime extra area	[155]
Voltage	Forward Body Biasing	It hides the effects of aging and slows degradation down based on Adaptive Supply Voltage (ASV) and Adaptive Body Bias (ABB).	Architec-ture	PE/ME	Removal	Runtime	Area Design overhead	[156]
Voltage	Scheduled V_{dd} Scaling	Gradual increase V_{dd} to compensate for the NBTI degradation instead of setting a fixed operating voltage guard band to rectify aging based on the aging sensor.	Architec-ture	PE/ME	Forecast	Runtime	Area overhead	[158]
Voltage	Dual Vth As-signment	The dual-threshold voltage assignment technique is used for the identified critical gates	Circuit	PE/ME	Prevention	Design-time	Area Design	[159]

enhancement. In this technique, the performance is dynamically adjusted using DVFS and architectural adaptation based on different resilience target points of PEs. For example, considering the target FIT (Failure-In-Time), if there is a margin in terms of MTTF, applications can be run faster. Otherwise, performance should be degraded to meet the target MTTF design point. Finally, the study of [162] proposes a hybrid DRM algorithm that combines thread migration and DVFS. The goal is to increase the lifetime resilience of the overall system to the desired target with minimal performance degradation. The DRM idea is to propose a control algorithm that continuously monitors the temperatures of processing elements of the CMP to extend the lifetime based on the MTTF model.

Table 8 demonstrates the summary and key features of proposed techniques to enhance lifetime resilience based on

TABLE 9. Some answered questions in this survey.

Index	Question Description	Survey Answer
1	Which dependability counterparts are essential for aging mitigation in DNN accelerators?	Fault prevention, fault removal, fault forecasting, and especially fault tolerance are the 4 countermeasures used in different articles to face aging paradigms of DNN accelerators.
2	Which safety standards are related to the DNN accelerator?	RTCA/DO-178C [50] for safety-critical airborne software, ISO 26262 for the automotive disciplines [2], and the generic safety standard of IEC 61508 [37] are 3 major standards of safety related to DNN accelerator.
3	What are the effects of aging on the DNN accelerator?	According to different mechanisms of aging, e.g., BTI, HCI, TDDB, and EM. Aging eventually causes TE and PSF occurrence and eventually leads to accelerator performance degradation or even failure. These effects threaten the functionality, timing, or even the lifetime of the DNN accelerator.
4	What are the main techniques for dealing with the PSFs in the DNN accelerator?	The major techniques proposed for improving the functional resilience of DNN accelerators are fault-aware training, skipping, pruning, ABFT, quantization, normalization, retraining, weight and sensitivity balancing, datatype and precision selection, selective and modular redundancies, recomputing by extra PE or recomputing unit, fault-aware mapping, and partial reconfiguration.
5	What are the main techniques for dealing with the TEs in the DNN accelerator?	The major techniques introduced for improving the timing resilience of DNN accelerators are training, retraining, time borrowing, arithmetic system selection, DVFS, masking, skipping, and pruning, sensitivity-based mapping, ABFT, frequency speculation, and partial reconfiguration.
6	What are the main approaches for extending the lifetime in the DNN accelerator?	The major approaches which can be used for improving timing resilience can be categorized into 2 parts: before the faults are seen and after the fault occurrence. At first, we try to manage effective parameters like temperature, voltage, and stress by general techniques like mapping, scheduling, and DVFS to postpone the occurrence of the faults. In the second, we can use all of the techniques presented for improving functional resilience and timing resilience.
7	What are the differences in dealing with aging-induced permanent faults of DNN deployment layers?	In this review, we investigate proposed studies with a top-dawn layer-wise approach that matches the general DNN deployment stack including model, algorithm, architecture, microarchitecture, and circuit and device. High-level techniques can have prohibitively lower costs and higher scalability to DNN complexity while providing acceptable resilience because they have indirect mitigating effects on more fault types of the underlying hardware. Low-level techniques are more accurate and efficient but have more overhead, mitigate specified fault types, and have lower scalability and moveability.
8	What are the scopes of dealing with aging-induced timing errors in the DNN accelerator?	4 basic scopes dealing with the aging of the DNN accelerators: 1) avoid it based on design-time consideration and robustness, 2) ignore and let propagate based on inherent resilience and design-time mechanisms arranged to mitigate the effects. 3) detect errors and try to remove or mitigate effects by runtime methods, 4) monitor the circuit to predict degradation and prevent error occurrence in runtime.
9	What is the efficacy of optimization on the resilience of the aging-aware DNN accelerator?	Although very high performance, cost, and energy efficiency have been achieved by optimizations, e.g., removing redundant weights, compressing data, and even sacrificing timing margin, such trends push the deep learning systems to the error threshold can be disastrous for the tasks they performed due to aging-induced faults or degradations. Accordingly, different healing techniques and mechanisms are required to compensate for optimizations.

TABLE 10. Some new research opportunities.

Index	New Research Idea	Idea Description
1	Aging-aware DNN Accelerator with Mixed Criticality	The mix-criticality task model was proposed by [166] to address the different criticality requirements in a mixed-criticality multicore system. In the DNN accelerator, it seems that we require research to present scheduling and WCET assessment techniques, especially considering the aging timing effects.
2	Aging-aware Temperature Management of DNN Accelerator	As we have mentioned, temperature is the major factor in aging. We have seen that a few works have been done to mitigate the temperature of the DNN accelerator such as using approximation. Some of the studies that we reviewed had been done for CMP or MPSoCs which can be extended for the DNN accelerator.
3	DTM for Aging-aware DNN Accelerator	DTM-based techniques try to mitigate the temperature by low-power techniques, e.g., DVFS and mapping. According to our search, nothing has been done about DTM usage for the DNN accelerator.
4	DRM for Aging-aware DNN Accelerator	DRM monitors system elements continuously and makes periodic decisions for changing control knobs shifting operations to a state where the lifetime extends. The references, we have mentioned are related to CMPs. According to our search, nothing has been done about DTM usage for the DNN accelerator.
5	Aging Prediction / Stress Estimation in DNN Accelerator	As we have seen, workload-induced stress is another factor of aging. We reviewed 3 major research done to mitigate the stress of DNN accelerators. In the past, stress estimation technique has been presented for processors [167]. According to our search, nothing has been done for the DNN accelerators.
6	Dynamic Timing Analysis in DNN Accelerator	We studied several articles related to frequency speculation for timing resilience improvement. The proposed works are based on TEs detection sensors or additional NN for delay prediction. Previous, research had been done on dynamic timing analysis (DTA) in processors [110].
7	Numbering System Effect on DNN Accelerator Aging	According to the literature, timing resilience depends on arithmetic and numbering systems [118]. There are research opportunities to improve the resilience of DNN accelerators using online arithmetic or redundant number systems.
8	Adaptive Approximation for Aging-aware DNN Accelerator	As we have seen, approximation has been used for power and temperature reduction. There are research opportunities to compensate for aging degradation using controlled approximations.
9	Explore the efficacy of IMC on the lifetime resilience of the DNN accelerator	According to the technique of floor planning, it is better to locate PEs or MEs with high-temperature dissipation as far apart as possible. On the other hand, IMC can reduce the data movements which can lead to temperature reduction too. Data movement between PE and off-chip MEs consumes two orders of magnitude more energy than a floating-point operation [17]. It seems that it is required to conduct new research to create a tradeoff.

different aging factors and introduced models which are categorized by different colors similar to Fig. 23.

VII. DISCUSSION AND FUTURE SCOPE

DNNs have affected many aspects of our lives while it is utilized in different safety-critical disciplines with a high demand for dependability. In this regard, DNN accelerators have shown a high degree of efficiency with great potential for acceleration beyond what is possible on conventional processors, making them the primary framework for DNN deployment on edge devices. Despite the inherent resilience of DNN applications, the proposed systems are highly vulnerable to dependability threats requiring special attention, particularly with the technology shrinking to Nano Era, which worsens the dependability concerns due to factors like aging. Some safety-critical application disciplines, such as autonomous cars and healthcare, require very strict dependability specifications. DNN accelerators will find adoption in these disciplines only if their dependability has been thoroughly proven. As such, the studied techniques can be used as the ultimate method of the efficacy of dependable solutions.

In this survey, we reviewed the major techniques for evaluating and improving the resilience of DNN algorithms and accelerators as a measure of dependability regarding different paradigms of aging. We studied the impact of various design decisions on accelerator resilience and

summarized the techniques of aging mitigation. The survey started with a brief review of related articles studying DNN accelerator resilience and efficiency issues. Next, we studied safety-critical standards, certification requirements, and mixed-critical system specifications. After that, we proposed preliminaries of this survey, including aging paradigms, DNN models, underlying platforms, and resilience concerns. Thus, we introduced a taxonomy of enhancing techniques of DNN accelerator resilience composed of functional, timing, and lifetime resilience according to considering different aging threats, e.g., PSFs, TEs, and lifetime. In this survey, we shed light on each category and show the amount of attention they gain separately from the aging perspective. Herein, we considered the deployed layers and platforms, scopes and approaches, time to apply, and possible overheads. For each category, the literature studies were analyzed, and representative works were presented in a uniform figure and a summary table.

Finally, this survey ended with a conclusion, answers to research questions, and a summary of some notable challenges and directions for future research. Table 9 shows answers to the main questions of this study. Also, Table 10 summarizes several topics for future research.

REFERENCES

[1] W. M. Gondal, J. M. Köhler, R. Grzeszick, G. A. Fink, and M. Hirsch, "Weakly-supervised localization of diabetic retinopathy lesions in retinal fundus images," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 2069–2073.

[2] S. Alcaide, L. Kosmidis, C. Hernandez, and J. Abella, "High-integrity GPU designs for critical real-time automotive systems," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 824–829.

[3] J. Athavale, A. Baldovin, R. Graefe, M. Paulitsch, and R. Rosales, "AI and reliability trends in safety-critical autonomous systems on ground and air," in *Proc. 50th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. Workshops (DSN-W)*, Jun. 2020, pp. 74–77.

[4] C. Adams, A. Spain, J. Parker, M. Hevert, J. Roach, and D. Cotten, "Towards an integrated GPU accelerated SoC as a flight computer for small satellites," in *Proc. IEEE Aerosp. Conf.*, Mar. 2019, pp. 1–7.

[5] J. Athavale, A. Baldovin, and M. Paulitsch, "Trends and functional safety certification strategies for advanced railway automation systems," in *Proc. IEEE Int. Rel. Phys. Symp. (IRPS)*, Apr. 2020, pp. 1–7.

[6] H. Ahmadian, R. Obermaisser, and J. Perez, *Distributed Real-Time Architecture for Mixed-Criticality Systems*. Boca Raton, FL, USA: CRC Press, 2018.

[7] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secure Comput.*, vol. 1, no. 1, pp. 11–33, Mar. 2004, doi: 10.1109/TDSC.2004.2.

[8] A. Arunachalam, S. Kundu, A. Raha, S. Banerjee, and K. Basu, "Fault resilience of DNN accelerators for compressed sensor inputs," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2022, pp. 329–332.

[9] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, Nov. 2018, Art. no. e00938, doi: 10.1016/j.heliyon.2018.e00938.

[10] S. Mittal, "A survey on modeling and improving reliability of DNN algorithms and accelerators," *J. Syst. Archit.*, vol. 104, Mar. 2020, Art. no. 101689, doi: 10.1016/j.sysarc.2019.101689.

[11] C. Liu, Z. Gao, S. Liu, X. Ning, H. Li, and X. Li, "Special session: Fault-tolerant deep learning: A hierarchical perspective," in *Proc. IEEE 40th VLSI Test Symp. (VTS)*, Apr. 2022, pp. 1–12.

[12] 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," *Microelectron. Rel.*, vol. 115, Dec. 2020, Art. no. 113969, doi: 10.1016/j.microrel.2020.113969.

[13] C. Torres-Huitzil and B. Girau, "Fault and error tolerance in neural networks: A review," *IEEE Access*, vol. 5, pp. 17322–17341, 2017, doi: 10.1109/ACCESS.2017.2742698.

[14] P. Pandey, N. D. Gundi, P. Basu, T. Shabanian, M. C. Patrick, K. Chakraborty, and S. Roy, "Challenges and opportunities in near-threshold DNN accelerators around timing errors," *J. Low Power Electron. Appl.*, vol. 10, no. 4, pp. 1–19, Dec. 2020, doi: 10.3390/jlpea10040033.

[15] M. Shafique, M. Naseer, T. Theocharides, C. Kyrkou, O. Mutlu, L. Orosa, and J. Choi, "Robust machine learning systems: Challenges, current trends, perspectives, and the road ahead," *IEEE Des. Test.*, vol. 37, no. 2, pp. 30–57, Apr. 2020.

[16] J. Perez-Cerrolaza, J. Abella, L. Kosmidis, A. J. Calderon, F. Cazorla, and J. L. Flores, "GPU devices for safety-critical systems: A survey," *ACM Comput. Surv.*, vol. 55, no. 7, pp. 1–37, Jul. 2023, doi: 10.1145/3549526.

[17] Y. Chen, Y. Xie, L. Song, F. Chen, and T. Tang, "A survey of accelerator architectures for deep neural networks," *Engineering*, vol. 6, no. 3, pp. 264–274, Mar. 2020, doi: 10.1016/j.eng.2020.01.007.

[18] R. Ayachi, Y. Said, and A. B. Abdelali, "Optimizing neural networks for efficient FPGA implementation: A survey," *Arch. Comput. Methods Eng.*, vol. 28, no. 7, pp. 4537–4547, Dec. 2021, doi: 10.1007/s11831-021-09530-9.

[19] H. Hussain, P. S. Tamizharasan, and C. S. Rahul, "Design possibilities and challenges of DNN models: A review on the perspective of end devices," *Artif. Intell. Rev.*, vol. 55, no. 7, pp. 5109–5167, Oct. 2022, doi: 10.1007/s10462-022-10138-z.

[20] M. Dhouibi, A. K. B. Salem, A. Saidi, and S. B. Saoud, "Accelerating deep neural networks implementation: A survey," *IET Comput. Digit. Techn.*, vol. 15, no. 2, pp. 79–96, Mar. 2021, doi: 10.1049/cdt2.12016.

[21] J. Henkel, L. Bauer, N. Dutt, P. Gupta, S. Nassif, M. Shafique, M. Tahoori, and N. Wehn, "Reliable on-chip systems in the nano-era: Lessons learnt and future trends," in *Proc. 50th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, May 2013, pp. 1–10.

[22] X. Guo and M. R. Stan, "Circuit techniques for BTI and EM accelerated and active recovery," in *Circadian Rhythms for Future Resilient Electronic Systems*. Cham, Switzerland: Springer, 2020, pp. 79–120.

[23] S. Mahapatra, N. Goel, and S. Mukhopadhyay, *Introduction: Bias Temperature Instability (BTI) in N and P Channel MOSFETs*. New Delhi, India: Springer, 2016, pp. 1–42.

[24] I. Hill, P. Chanawala, R. Singh, S. A. Sheikholeslam, and A. Ivanov, "CMOS reliability from past to future: A survey of requirements, trends, and prediction methods," *IEEE Trans. Device Mater. Rel.*, vol. 22, no. 1, pp. 1–18, Mar. 2022, doi: 10.1109/TDMR.2021.3131345.

[25] D. S. Huang, J. H. Lee, Y. S. Tsai, Y. F. Wang, Y. S. Huang, C. K. Lin, R. Lu, and J. He, "Comprehensive device and product level reliability studies on advanced CMOS technologies featuring 7 nm high-k metal gate FinFET transistors," in *Proc. IEEE Int. Rel. Phys. Symp. (IRPS)*, Mar. 2018, pp. 6F.7-1–6F.7-5.

[26] C. Liu, H.-C. Sagong, H. Kim, S. Choo, H. Lee, Y. Kim, H. Kim, B. Jo, M. Jin, J. Kim, S. Ha, S. Pae, and J. Park, "Systematical study of 14 nm FinFET reliability: From device level stress to product HTOL," in *Proc. IEEE Int. Rel. Phys. Symp.*, Apr. 2015, pp. 2F.3.1–2F.3.5.

[27] J. B. Bernstein, A. Bensoussan, and E. Bender, "Reliability prediction with MTOL," *Microelectron. Rel.*, vol. 68, pp. 91–97, Jan. 2017.

[28] N. P. Jouppi et al., "In-datacenter performance analysis of a tensor processing unit," in *Proc. ACM/IEEE 44th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2017, pp. 1–12.

[29] S. Venkataramani et al., "Efficient AI system design with cross-layer approximate computing," *Proc. IEEE*, vol. 108, no. 12, pp. 2232–2250, Dec. 2020.

[30] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[31] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[32] A. P. Arechiga and A. J. Michaels, "The robustness of modern deep learning architectures against single event upset errors," in *Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC)*, Sep. 2018, pp. 1–6, doi: 10.1109/HPEC.2018.8547532.

[33] P. N. Whatmough, S. K. Lee, D. Brooks, and G.-Y. Wei, "DNN engine: A 28-nm timing-error tolerant sparse deep neural network processor for IoT applications," *IEEE J. Solid-State Circuits*, vol. 53, no. 9, pp. 2722–2731, Sep. 2018, doi: 10.1109/JSSC.2018.2841824.

[34] F. F. D. Santos, P. F. Pimenta, C. Lunardi, L. Draghetti, L. Carro, D. Kaeli, and P. Rech, "Analyzing and increasing the reliability of convolutional neural networks on GPUs," *IEEE Trans. Rel.*, vol. 68, no. 2, pp. 663–677, Jun. 2019.

[35] H. Sharma, J. Park, D. Mahajan, E. Amaro, J. K. Kim, C. Shao, A. Mishra, and H. Esmaeilzadeh, "From high-level deep neural models to FPGAs," in *Proc. 49th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2016, pp. 1–12.

[36] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017, doi: 10.1109/JSSC.2016.2616357.

[37] D. Ghimire, D. Kil, and S.-H. Kim, "A survey on efficient convolutional neural networks and hardware acceleration," *Electronics*, vol. 11, no. 6, p. 945, Mar. 2022.

[38] A. Parashar, M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, J. Emer, S. W. Keckler, and W. J. Dally, "SCNN: An accelerator for compressed-sparse convolutional neural networks," *ACM SIGARCH Comput. Archit. News*, vol. 45, no. 2, pp. 27–40, May 2017.

[39] L. Cavigelli and L. Benini, "Origami: A 803-GOp/s/W convolutional network accelerator," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 11, pp. 2461–2475, Nov. 2017.

[40] J.-C. Laprie, "Dependable computing and fault-tolerance," in *FTCS Dig. Papers*, Jun. 1985, vol. 10, no. 2, p. 124.

[41] J.-C. Laprie, "From dependability to resilience," in *Proc. 38th IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jun. 2008, pp. G8–G9.

[42] T. Anderson, *Resilient Computing Systems*, vol. 1. Hoboken, NJ, USA: Wiley, 1985.

[43] C. Constantinescu, "Intermittent faults and effects on reliability of integrated circuits," in *Proc. Annu. Rel. Maintainability Symp.*, Jan. 2008, pp. 370–374.

[44] B. Johnson, "Fault-tolerant microprocessor-based systems," *IEEE Micro*, vol. MM-4, no. 6, pp. 6–21, Dec. 1984.

[45] M. Stanisavljević, A. Schmid, and Y. Leblebici, "Fault-tolerant architectures and approaches," in *Reliability of Nanoscale Circuits and Systems*. New York, NY, USA: Springer, 2011, pp. 35–47.

[46] E. Dubrova, "Hardware redundancy," in *Fault-Tolerant Design*. New York, NY, USA: Springer, 2013, pp. 55–86.

[47] H. Giesen, B. Gojman, R. Rubin, J. Kim, and A. Dehon, "Continuous online self-monitoring introspection circuitry for timing repair by incremental partial-reconfiguration (COSMIC TRIP)," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 11, no. 1, pp. 1–23, Mar. 2018.

[48] V. P. Nelson, "Fault-tolerant computing: Fundamental concepts," *Computer*, vol. 23, no. 7, pp. 19–25, Jul. 1990.

[49] R. Debouk, "Overview of the second edition of ISO 26262: Functional safety-road vehicles," *J. Syst. Saf.*, vol. 55, no. 1, pp. 13–21, 2019.

[50] *Software Considerations in Airborne Systems and Equipment Certification*, document RTCA DO-178C, Radio Technical Commission for Aeronautics, Washington, DC, USA, 2011.

[51] *Robots and Robotic Devices—Safety Requirements for Industrial Robots—Part 1: Robots*, International Organization for Standardization, Standard ISO 8373:2012, 2012.

[52] E. N. Cenelec, *Railway Applications—Communication, Signalling and Processing Systems-Software for Railway Control and Protection Systems*, Standard 50128, 2012.

[53] J. P. Cerrolaza, R. Obermaisser, J. Abella, F. J. Cazorla, K. Grüttner, I. Agirre, H. Ahmadian, and I. Allende, "Multi-core devices for safety-critical systems: A survey," *ACM Comput. Surv.*, vol. 53, no. 4, pp. 1–38, Jul. 2021.

[54] A. Larrucea, J. Perez, I. Agirre, V. Brocal, and R. Obermaisser, "A modular safety case for an IEC-61508 compliant generic hypervisor," in *Proc. Euromicro Conf. Digit. Syst. Design*, Aug. 2015, pp. 571–574.

[55] R. Amarnath, P. Munk, E. Thaden, A. Nordmann, and S. Burton, "Dependability challenges in the model-driven engineering of automotive systems," in *Proc. IEEE Int. Symp. Softw. Rel. Eng. Workshops (ISSREW)*, Oct. 2016, pp. 1–4.

[56] W. Maass, "Noise as a resource for computation and learning in networks of spiking neurons," *Proc. IEEE*, vol. 102, no. 5, pp. 860–880, May 2014.

[57] F. Su, P. Yuan, Y. Wang, and C. Zhang, "The superior fault tolerance of artificial neural network training with a fault/noise injection-based genetic algorithm," *Protein Cell*, vol. 7, no. 10, pp. 735–748, Oct. 2016.

[58] J. J. Zhang, K. Basu, and S. Garg, "Fault-tolerant systolic array based accelerators for deep neural network execution," *IEEE Des. Test.*, vol. 36, no. 5, pp. 44–53, Oct. 2019.

[59] D. Xu, C. Chu, Q. Wang, C. Liu, Y. Wang, L. Zhang, H. Liang, and K.-T. Cheng, "A hybrid computing architecture for fault-tolerant deep learning accelerators," in *Proc. IEEE 38th Int. Conf. Comput. Design (ICCD)*, Oct. 2020, pp. 478–485.

[60] C. De Sio, S. Azimi, and L. Sterpone, "FireNN: Neural networks reliability evaluation on hybrid platforms," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 2, pp. 549–563, Apr. 2022, doi: 10.1109/TETC.2022.3152668.

[61] B. Salami, O. S. Unsal, and A. C. Kestelman, "On the resilience of RTL NN accelerators: Fault characterization and mitigation," in *Proc. 30th Int. Symp. Comput. Archit. High Perform. Comput. (SBAC-PAD)*, Sep. 2018, pp. 322–329.

[62] 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 *Proc. 55th ACM/ESDA/IEEE Design Autom. Conf. (DAC)*, Jun. 2018, pp. 1–6.

[63] J. Marques, J. Andrade, and G. Falcao, "Unreliable memory operation on a convolutional neural network processor," in *Proc. IEEE Int. Workshop Signal Process. Syst. (SiPS)*, Oct. 2017, pp. 1–6.

[64] E. M. El Mhamdi and R. Guerraoui, "When neurons fail," 2017, *arXiv:1706.08884*.

[65] Y. Wang, J. Deng, Y. Fang, H. Li, and X. Li, "Resilience-aware frequency tuning for neural-network-based approximate computing chips," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 10, pp. 2736–2748, Oct. 2017, doi: 10.1109/TVLSI.2017.2682885.

[66] M. A. Hanif, L.-H. Hoang, and M. Shafique, "Cross-layer approaches for improving the dependability of deep learning systems," in *Proc. 23th Int. ACM Workshop Softw. Compil. Embedded Syst. (SCOPES)*, May 2020, pp. 78–81, doi: 10.1145/3378678.3391884.

[67] S. Piche, "Robustness of feedforward neural networks," in *Proc. IJCNN Int. Joint Conf. Neural Netw.*, Jun. 1992, pp. 783–788.

[68] C. Schorn, A. Guntoro, and G. Ascheid, "An efficient bit-flip resilience optimization method for deep neural networks," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 1507–1512.

[69] S. Kim, P. Howe, T. Moreau, A. Alaghi, L. Ceze, and V. Sathe, "MATIC: Learning around errors for efficient low-voltage neural network accelerators," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 1–6.

[70] X. Jiao, M. Luo, J.-H. Lin, and R. K. Gupta, "An assessment of vulnerability of hardware neural networks to dynamic voltage and temperature variations," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2017, pp. 945–950.

[71] O. Osoba and B. Kosko, "Noise-enhanced clustering and competitive learning algorithms," *Neural Netw.*, vol. 37, pp. 132–140, Jan. 2013.

[72] G. B. Hacene, F. Leduc-Primeau, A. B. Soussia, V. Gripon, and F. Gagnon, "Training modern deep neural networks for memory-fault robustness," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2019, pp. 1–5.

[73] Z. He, J. Lin, R. Ewetz, J.-S. Yuan, and D. Fan, "Noise injection adaption: End-to-end ReRAM crossbar non-ideal effect adaption for neural network mapping," in *Proc. 56th ACM/IEEE Design Autom. Conf. (DAC)*, Jun. 2019, pp. 1–6.

[74] W. Choi, D. Shin, J. Park, and S. Ghosh, "Sensitivity based error resilient techniques for energy efficient deep neural network accelerators," in *Proc. 56th ACM/IEEE Design Autom. Conf. (DAC)*, Jun. 2019, pp. 1–6.

[75] Y. Huang, "Advances in artificial neural networks—Methodological development and application," *Algorithms*, vol. 2, no. 3, pp. 973–1007, Aug. 2009.

[76] K. Xing, "Training for 'unstable' CNN accelerator: A case study on FPGA," 2018, *arXiv:1812.01689*.

[77] S. Kundu, S. Banerjee, A. Raha, F. Su, S. Natarajan, and K. Basu, "Trouble-shooting at GAN point: Improving functional safety in deep learning accelerators," *IEEE Trans. Comput.*, vol. 72, no. 8, pp. 2194–2208, Aug. 2023.

[78] K. Jia, Z. Liu, Q. Wei, F. Qiao, X. Liu, Y. Yang, H. Fan, and H. Yang, "Calibrating process variation at system level with in-situ low-precision transfer learning for analog neural network processors," in *Proc. 55th ACM/ESDA/IEEE Design Autom. Conf. (DAC)*, Jun. 2018, pp. 1–6.

[79] X. Ning, G. Ge, W. Li, Z. Zhu, Y. Zheng, X. Chen, Z. Gao, Y. Wang, and H. Yang, "FTT-NAS: Discovering fault-tolerant convolutional neural architecture," *ACM Trans. Design Autom. Electron. Syst.*, vol. 26, no. 6, pp. 1–24, Nov. 2021.

[80] E. Ozen and A. Orailoglu, "SNR: Squeezing numerical range defuses bit error vulnerability surface in deep neural networks," *ACM Trans. Embedded Comput. Syst.*, vol. 20, no. 5s, pp. 1–25, Oct. 2021.

[81] K. Zhao, S. Di, S. Li, X. Liang, Y. Zhai, J. Chen, K. Ouyang, F. Cappello, and Z. Chen, "FT-CNN: Algorithm-based fault tolerance for convolutional neural networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1677–1689, Jul. 2021.

[82] E. Ozen and A. Orailoglu, "Low-cost error detection in deep neural network accelerators with linear algorithmic checksums," *J. Electron. Test.*, vol. 36, no. 6, pp. 703–718, Dec. 2020.

[83] D. Filippas, N. Margomenos, N. Mitianoudis, C. Nicopoulos, and G. Dimitrakopoulos, "Low-cost online convolution checksum checker," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 30, no. 2, pp. 201–212, Feb. 2022.

[84] T. Liu, W. Wen, L. Jiang, Y. Wang, C. Yang, and G. Quan, "A fault-tolerant neural network architecture," in *Proc. 56th ACM/IEEE Design Autom. Conf. (DAC)*, Jun. 2019, pp. 1–6.

[85] M. D. Emmerson and R. I. Damper, "Determining and improving the fault tolerance of multilayer perceptrons in a pattern-recognition application," *IEEE Trans. Neural Netw.*, vol. 4, no. 5, pp. 788–793, Sep. 1993.

[86] F. M. Dias and A. Antunes, "Fault tolerance improvement through architecture change in artificial neural networks," in *Proc. Int. Symp. Intell. Comput. Appl.*, 2008, pp. 248–257.

[87] I. Baek, W. Chen, Z. Zhu, S. Samii, and R. R. Rajkumar, "FT-DeepNets: Fault-tolerant convolutional neural networks with kernel-based duplication," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2022, pp. 1878–1887.

[88] S. Eldridge and A. Joshi, "Exploiting hidden layer modular redundancy for fault-tolerance in neural network accelerators," in *Proc. Boston Area Archit. (BARC) Workshop*, 2015, pp. 1–2.

[89] J. A. Clemente, W. Mansour, R. Ayoubi, F. Serrano, H. Mecha, H. Ziade, W. El Falou, and R. Velazco, "Hardware implementation of a fault-tolerant Hopfield Neural Network on FPGAs," *Neurocomputing*, vol. 171, pp. 1606–1609, Jan. 2016.

[90] A. Ahmadi, M. H. Sargolzaie, S. M. Fakhraie, C. Lucas, and S. Vakili, "A low-cost fault-tolerant approach for hardware implementation of artificial neural networks," in *Proc. Int. Conf. Comput. Eng. Technol.*, vol. 2, Jan. 2009, pp. 93–97.

[91] Y. Zhao, K. Wang, and A. Louri, "FSA: An efficient fault-tolerant systolic array-based DNN accelerator architecture," in *Proc. IEEE 40th Int. Conf. Comput. Design (ICCD)*, Oct. 2022, pp. 545–552.

[92] O. Temam, "A defect-tolerant accelerator for emerging high-performance applications," in *Proc. 39th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2012, pp. 356–367.

[93] F. Libano, B. Wilson, J. Anderson, M. J. Wirthlin, C. Cazzaniga, C. Frost, and P. Rech, "Selective hardening for neural networks in FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 66, no. 1, pp. 216–222, Jan. 2019.

[94] N. Khoshavi, A. Roohi, C. Broyles, S. Sargolzaei, Y. Bi, and D. Z. Pan, "SHIELDeNN: Online accelerated framework for fault-tolerant deep neural network architectures," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, Jul. 2020, pp. 1–6.

[95] J. J. Zhang, T. Gu, K. Basu, and S. Garg, "Analyzing and mitigating the impact of permanent faults on a systolic array based neural network accelerator," in *Proc. IEEE 36th VLSI Test Symp. (VTS)*, Apr. 2018, pp. 1–6.

[96] M. Sadi and U. Guin, "Test and yield loss reduction of AI and deep learning accelerators," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 1, pp. 104–115, Jan. 2022.

[97] M. Abdullah Hanif and M. Shafique, "SalvageDNN: Salvaging deep neural network accelerators with permanent faults through saliency-driven fault-aware mapping," *Phil. Trans. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 378, no. 2164, Feb. 2020, Art. no. 20190164, doi: 10.1098/rsta.2019.0164.

[98] C. Schorn, A. Guntoro, and G. Ascheid, "Accurate neuron resilience prediction for a flexible reliability management in neural network accelerators," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2018, pp. 979–984.

[99] L. Zhao, Y. Zhang, and J. Yang, "AEP: An error-bearing neural network accelerator for energy efficiency and model protection," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2017, pp. 1047–1053.

[100] S. Kundu, K. Basu, M. Sadi, T. Titirsha, S. Song, A. Das, and U. Guin, "Special session: Reliability analysis for AI/ML hardware," in *Proc. IEEE 39th VLSI Test Symp. (VTS)*, Apr. 2021, pp. 1–10, doi: 10.1109/VTS50974.2021.9441050.

[101] 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 *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, Nov. 2017, pp. 1–12.

[102] A. Ruospo, A. Bosio, A. Ianne, and E. Sanchez, "Evaluating convolutional neural networks reliability depending on their data representation," in *Proc. 23rd Euromicro Conf. Digit. Syst. Design (DSD)*, Aug. 2020, pp. 672–679, doi: 10.1109/DSD51259.2020.00109.

[103] Y. Long, X. She, and S. Mukhopadhyay, "Design of reliable DNN accelerator with un-reliable ReRAM," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 1769–1774.

[104] H. R. Mahdiani, S. M. Fakhraie, and C. Lucas, "Relaxed fault-tolerant hardware implementation of neural networks in the presence of multiple transient errors," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 8, pp. 1215–1228, Aug. 2012.

[105] G. Srinivasan, P. Wijesinghe, S. S. Sarwar, A. Jaiswal, and K. Roy, "Significance driven hybrid 8T-6T SRAM for energy-efficient synaptic storage in artificial neural networks," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2016, pp. 151–156.

[106] Z. Du, K. Palem, A. Lingamneni, O. Temam, Y. Chen, and C. Wu, "Leveraging the error resilience of machine-learning applications for designing highly energy efficient accelerators," in *Proc. 19th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2014, pp. 201–206.

[107] T. G. Bertoa, G. Gambardella, N. J. Fraser, M. Blott, and J. McAllister, "Fault-tolerant neural network accelerators with selective TMR," *IEEE Des. Test. IEEE Des. Test. Comput.*, vol. 40, no. 2, pp. 67–74, Apr. 2023.

[108] L. Miculka and Z. Kotasek, "Generic partial dynamic reconfiguration controller for transient and permanent fault mitigation in fault tolerant systems implemented into FPGA," in *Proc. 17th Int. Symp. Design Diag. Electron. Circuits Syst.*, Apr. 2014, pp. 171–174.

[109] J. Zhang, Y. Guan, and C. Mao, "Optimal partial reconfiguration for permanent fault recovery on SRAM-based FPGAs in space mission," *Adv. Mech. Eng.*, vol. 5, Jan. 2013, Art. no. 783673.

[110] I. Moghaddasi, M. E. Salehi Nasab, and M. Kargahi, "Aging-aware instruction-level statistical dynamic timing analysis for embedded processors," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 2, pp. 433–442, Feb. 2020.

[111] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: A low-power pipeline based on circuit-level timing speculation," in *Proc. 36th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec. 2003, pp. 7–18.

[112] T. Marty, T. Yuki, and S. Derrien, "Algorithm level timing speculation for convolutional neural network accelerators," Univ. Rennes, Inria, CNRS, IRISA, France, 2018.

[113] J. Zhang, K. Rangineni, Z. Ghodsi, and S. Garg, "ThUnderVolt: Enabling aggressive voltage underscaling and timing error resilience for energy efficient deep learning accelerators," in *Proc. 55th ACM/ESDA/IEEE Design Autom. Conf. (DAC)*, Jun. 2018, pp. 1–6, doi: 10.1109/DAC.2018.8465918.

[114] N. Chandramoorthy, K. Swaminathan, M. Cochet, A. Paidimarri, S. Eldridge, R. V. Joshi, M. M. Ziegler, A. Buyuktosunoglu, and P. Bose, "Resilient low voltage accelerators for high energy efficiency," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2019, pp. 147–158.

[115] A. Chenouf, B. Djezzar, H. Bentarzi, and A. Benabdelmoumene, "Sizing of the CMOS 6T-SRAM cell for NBTI ageing mitigation," *IET Circuits, Devices Syst.*, vol. 14, no. 4, pp. 555–561, Jul. 2020.

[116] W. Liu and C.-H. Chang, "Analysis of circuit aging on accuracy degradation of deep neural network accelerator," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2019, pp. 1–5, doi: 10.1109/ISCAS.2019.8702226.

[117] J. Deng, Y. Fang, Z. Du, Y. Wang, H. Li, O. Temam, P. Ienne, D. Novo, X. Li, Y. Chen, and C. Wu, "Retraining-based timing error mitigation for hardware neural networks," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2015, pp. 593–596.

[118] K. Shi, D. Boland, E. Stott, S. Bayliss, and G. A. Constantinides, "Datapath synthesis for overclocking: Online arithmetic for latency-accuracy trade-offs," in *Proc. 51st ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2014, pp. 1–6.

[119] H. Kim, J. Kim, H. Amrouch, J. Henkel, A. Gerstlauer, K. Choi, and H. Park, "Aging compensation with dynamic computation approximation," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 4, pp. 1319–1332, Apr. 2020.

[120] S. Salamin, G. Zervakis, O. Spantidi, I. Anagnostopoulos, J. Henkel, and H. Amrouch, "Reliability-aware quantization for anti-aging NPUs," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Feb. 2021, pp. 1460–1465.

[121] B. Reagen, P. Whatmough, R. Adolf, S. Rama, H. Lee, S. K. Lee, J. M. Hernández-Lobato, G.-Y. Wei, and D. Brooks, "Minerva: Enabling low-power, highly-accurate deep neural network accelerators," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 267–278.

[122] W. Liu and C.-H. Chang, "A forward error compensation approach for fault resilient deep neural network accelerator design," in *Proc. 5th ACM Workshop Attacks Solutions Hardw. Secur.*, Nov. 2021, pp. 41–50, doi: 10.1145/3474376.3487281.

[123] T. Marty, T. Yuki, and S. Derrien, "Safe overclocking for CNN accelerators through algorithm-level error detection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 12, pp. 4777–4790, Dec. 2020, doi: 10.1109/TCAD.2020.2981056.

[124] P. Pandey, P. Basu, K. Chakraborty, and S. Roy, "GreenTPU: Improving timing error resilience of a near-threshold tensor processing unit," in *Proc. 56th ACM/IEEE Design Autom. Conf. (DAC)*, Jun. 2019, pp. 1–6.

[125] J. Zhang and S. Garg, "FATE: Fast and accurate timing error prediction framework for low power DNN accelerator design," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2018, pp. 1–8, doi: 10.1145/3240765.3240809.

[126] N. D. Gundi, P. Pandey, S. Roy, and K. Chakraborty, "Implementing a timing error-resilient and energy-efficient near-threshold hardware accelerator for deep neural network inference," *J. Low Power Electron. Appl.*, vol. 12, no. 2, p. 32, Jun. 2022, doi: 10.3390/jlpea12020032.

[127] H. Giesen, R. Rubin, B. Gojman, and A. DeHon, "Self-adaptive timing repair," *IEEE Des. Test. IEEE Des. Test. Comput.*, vol. 34, no. 6, pp. 54–62, Dec. 2017.

[128] M. Alam, "Reliability- and process-variation aware design of integrated circuits," *Microelectron. Rel.*, vol. 48, nos. 8–9, pp. 1114–1122, Aug. 2008.

[129] K. A. Bowman, B. L. Austin, J. C. Eble, X. Tang, and J. D. Meindl, "A physical alpha-power law MOSFET model," in *Proc. Int. Symp. Low Power Electron. Design*, Aug. 1999, pp. 218–222.

[130] J. W. McPherson and J. W. McPherson, "Time-to-failure modeling," *Reliability Physics and Engineering: Time-To-Failure Modeling*. Cham, Switzerland: Springer,2013, pp. 37–49.

[131] H. Amrouch, G. Zervakis, S. Salamin, H. Kattan, I. Anagnostopoulos, and J. Henkel, "NPU thermal management," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 11, pp. 3842–3855, Nov. 2020.

[132] W. Sootkaneung, S. Howimanporn, and S. Chookaew, "Thermal effect on performance, power, and BTI aging in FinFET-based designs," in *Proc. Euromicro Conf. Digit. Syst. Design (DSD)*, Aug. 2017, pp. 345–351.

[133] J.-M. Koo, S. Im, L. Jiang, and K. E. Goodson, "Integrated microchannel cooling for three-dimensional electronic circuit architectures," *J. Heat Transf.*, vol. 127, no. 1, pp. 49–58, Jan. 2005.

[134] T. Brunschwiler, B. Michel, H. Rothuizen, U. Kloter, B. Wunderle, H. Oppermann, and H. Reichl, "Forced convective interlayer cooling in vertically integrated packages," in *Proc. 11th Intersociety Conf. Thermal Thermomech. Phenomena Electron. Syst.*, May 2008, pp. 1114–1125.

[135] K. Sankaranarayanan, S. Velusamy, M. Stan, and K. Skadron, "A case for thermal-aware floorplanning at the microarchitectural level," *J. Instruct.-Level Parallelism*, vol. 7, no. 1, pp. 8–16, 2005.

[136] D. Brooks and M. Martonosi, "Dynamic thermal management for high-performance microprocessors," in *Proc. 7th Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Jan. 2001, pp. 171–182.

[137] A. K. Coskun, T. S. Rosing, and K. C. Gross, "Temperature management in multiprocessor SoCs using online learning," in *Proc. 45th ACM/IEEE Design Autom. Conf.*, Jun. 2008, pp. 890–893.

[138] A. Kumar, L. Shang, L.-S. Peh, and N. K. Jha, "System-level dynamic thermal management for high-performance microprocessors," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 1, pp. 96–108, Jan. 2008.

[139] J. Yang, X. Zhou, M. Chrobak, Y. Zhang, and L. Jin, "Dynamic thermal management through task scheduling," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, Apr. 2008, pp. 191–201.

[140] S. Yin, S. Tang, X. Lin, P. Ouyang, F. Tu, L. Liu, J. Zhao, C. Xu, S. Li, Y. Xie, and S. Wei, "Parana: A parallel neural architecture considering thermal problem of 3D stacked memory," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 1, pp. 146–160, Jan. 2019.

[141] S. Pandey and P. R. Panda, "NeuroMap: Efficient task mapping of deep neural networks for dynamic thermal management in high-bandwidth memory," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 11, pp. 3602–3613, Nov. 2022.

[142] A. Naveh, "Power and thermal management in the Intel Core Duo processor," *Intel Technol. J.*, vol. 10, no. 2, pp. 109–122, May 2006.

[143] S. O. Memik, R. Mukherjee, M. Ni, and J. Long, "Optimizing thermal sensor allocation for microprocessors," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 3, pp. 516–527, Mar. 2008.

[144] O. Khan and S. Kundu, "A framework for predictive dynamic temperature management of microprocessor systems," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2008, pp. 258–263.

[145] J. Donald and M. Martonosi, "Leveraging simultaneous multithreading for adaptive thermal control," in *Proc. 2nd Workshop Temp.-Aware Comput. Syst.*, 2005, pp. 1–10.

[146] K. Patel, W. Lee, and M. Pedram, "Active bank switching for temperature control of the register file in a microprocessor," in *Proc. 17th ACM Great Lakes Symp. (VLSI)*, Mar. 2007, pp. 231–234.

[147] S. Ghosh, J.-H. Choi, P. Ndai, and K. Roy, "O^2C: Occasional two-cycle operations for dynamic thermal management in high performance in-order microprocessors," in *Proc. 13th Int. Symp. Low Power Electron. Design (ISLPED)*, Aug. 2008, pp. 189–192.

[148] H. Shin, M. Kang, and L.-S. Kim, "A thermal-aware optimization framework for ReRAM-based deep neural network acceleration," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, Nov. 2020, pp. 1–9.

[149] G. Zervakis, I. Anagnostopoulos, S. Salamin, O. Spantidi, I. Roman-Ballesteros, J. Henkel, and H. Amrouch, "Thermal-aware design for approximate DNN accelerators," *IEEE Trans. Comput.*, vol. 71, no. 10, pp. 2687–2697, Oct. 2022, doi: 10.1109/TC.2022.3141054.

[150] O. Spantidi, G. Zervakis, I. Anagnostopoulos, H. Amrouch, and J. Henkel, "Positive/negative approximate multipliers for DNN accelerators," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, Nov. 2021, pp. 1–9.

[151] S. M. Nabavinejad, S. Reda, and M. Ebrahimi, "Coordinated batching and DVFS for DNN inference on GPU accelerators," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 10, pp. 2496–2508, Oct. 2022.

[152] D. Lorenz, M. Barke, and U. Schlichtmann, "Aging analysis at gate and macro cell level," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2010, pp. 77–84.

[153] S. Song and A. Das, "A case for lifetime reliability-aware neuromorphic computing," 2020, *arXiv:2007.02210*.

[154] M. A. Hanif and M. Shafique, "DNN-Life: An energy-efficient aging mitigation framework for improving the lifetime of on-chip weight memories in deep neural network hardware architectures," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Feb. 2021, pp. 729–734, doi: 10.23919/DATE51398.2021.9473943.

[155] N. L. Muñoz, A. Valero, R. G. Tejero, and D. Zoni, "Gated-CNN: Combating NBTI and HCI aging effects in on-chip activation memories of convolutional neural network accelerators," *J. Syst. Archit.*, vol. 128, Jul. 2022, Art. no. 102553, doi: 10.1016/j.sysarc.2022.102553.

[156] A. Tiwari and J. Torrellas, "Facelift: Hiding and slowing down aging in multicores," in *Proc. 41st IEEE/ACM Int. Symp. Microarchit.*, Nov. 2008, pp. 129–140.
[157] T. Chen and S. Naffziger, "Comparison of adaptive body bias (ABB) and adaptive supply voltage (ASV) for improving delay and leakage under the presence of process variation," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 5, pp. 888–899, Oct. 2003.
[158] L. Zhang and R. P. Dick, "Scheduled voltage scaling for increasing lifetime in the presence of NBTI," in *Proc. Asia South Pacific Design Autom. Conf.*, Jan. 2009, pp. 492–497.
[159] M. Raji, R. Mahmoudi, B. Ghavami, and S. Keshavarzi, "Lifetime reliability improvement of nano-scale digital circuits using dual threshold voltage assignment," *IEEE Access*, vol. 9, pp. 114120–114134, 2021.
[160] S. Zafar, B. H. Lee, J. Stathis, A. Callegari, and T. Ning, "A model for negative bias temperature instability (NBTI) in oxide and high-κ pFETs 13-×-C6D8C7F5F2," in *Proc. Dig. Tech. Papers. Symp. VLSI Technol.*, Jun. 2004, pp. 208–209.
[161] M. G. Moghaddam, A. Yamamoto, and C. Ababei, "Investigation of DVFS based dynamic reliability management for chip multiprocessors," in *Proc. Int. Conf. High Perform. Comput. Simulation (HPCS)*, Jul. 2015, pp. 563–568.
[162] M. G. Moghaddam, "Dynamic energy and reliability management in network-on-chip based chip multiprocessors," in *Proc. 8th Int. Green Sustain. Comput. Conf. (IGSC)*, Oct. 2017, pp. 1–4.
[163] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model compression and hardware acceleration for neural networks: A comprehensive survey," *Proc. IEEE*, vol. 108, no. 4, pp. 485–532, Apr. 2020.
[164] I. Sommerville, *Software Engineering*, 10th ed. Harlow, U.K.: Pearson, 2016.
[165] Wikipedia Foundation. *Automotive Safety Integrity Level*. Accessed: Jan. 15, 2023. [Online]. Available: https://en.wikipedia.org/wiki/Automotive_Safety_Integrity_Level
[166] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *Proc. 28th IEEE Int. Real-Time Syst. Symp. (RTSS)*, Dec. 2007, pp. 239–243.
[167] I. Moghaddasi, A. Fouman, M. E. Salehi, and M. Kargahi, "Instruction-level NBTI stress estimation and its application in runtime aging prediction for embedded processors," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 8, pp. 1427–1437, Aug. 2019.

IRAJ MOGHADDASI received the B.Sc. degree in computer engineering from Shahid Beheshti University, Tehran, Iran, in September 2000, the M.Sc. degree in computer engineering from the Iran University of Science and Technology, Tehran, in May 2003, and the Ph.D. degree from the School of Electrical and Computer Engineering, University of Tehran, Tehran, in September 2018. From 2019 to 2021, he was a Research Associate with the Iran Telecommunication Research Center (ITRC). He is currently a Postdoctoral Researcher with Chosun University. His research interests include computer architecture, reliable and high-performance computing, DL hardware accelerators, resilient embedded systems, and SRAM-based processing in memory.

SAEID GORGIN (Senior Member, IEEE) received the B.S. degree in computer engineering from Islamic Azad University, South Tehran Branch, Tehran, Iran, in 2001, the M.S. degree in computer engineering from Islamic Azad University Science and Research Branch, Tehran, in 2004, and the Ph.D. degree in computer system architecture from Shahid Beheshti University, Tehran, in 2010. He is currently an Associate Professor of computer engineering with the Department of Electrical Engineering and Information Technology, Iranian Research Organization for Science and Technology, Tehran. He is also a Visiting Scientist with the Laboratory of Computer Systems, Department of Computer Engineering, Chosun University, South Korea. His current research interests include computing systems, computer arithmetic, and VLSI design.

JEONG-A LEE (Senior Member, IEEE) received the B.S. degree (Hons.) in computer engineering from Seoul National University, Seoul, South Korea, in 1982, the M.S. degree in computer science from Indiana University Bloomington, Bloomington, IN, USA, in 1985, and the Ph.D. degree in computer science from the University of California at Los Angeles, Los Angeles, CA, USA, in 1990. From 1990 to 1995, she was an Assistant Professor with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX, USA. Since 1995, she has been with Chosun University, Gwangju, South Korea. From 2008 to 2009, she was the Program Director of the ECE Division, National Research Foundation of Korea. Her current research interests include high-performance computer architectures, memory architecture, approximate computing, and reliable computing. She is a member of the National Academy of Engineering, South Korea.