

A Systematic PRISMA Survey on Fault-Tolerant DNN Accelerator Architectures for Safety-Critical Systems

Farah Natiq Qassabbashi, Shawkat Sabah Khairullah and Shefa A. Dawwd *

Department of Computer Engineering, College of Engineering, University of Mosul, Mosul 41002, Iraq; farah.qassabbashi@uomosul.edu.iq (F.N.Q.); shawkat.sabah@uomosul.edu.iq (S.S.K.)

* Correspondence: shefa.dawwd@uomosul.edu.iq

Abstract

Deep Neural Networks (DNNs) are increasingly being used in the design of industrial safety-critical autonomous applications such as autonomous vehicles, industrial robotics, and medical instrumentation and control systems. Ensuring reliable and robust operation of the DNN-based safety-critical systems is challenging because of the complex structure of DNN hardware accelerators utilized for inference that are susceptible to the effects of multi-faults, common-cause fault models, data uncertainties, and unpredictable erroneous behavior. Additionally, transient, permanent, and timing faults affect the accelerator design of processing elements, memory arrays, and datapaths, propagate through DNN computations, and potentially can cause catastrophic failures at the system level. The objective of this survey paper is to systematically evaluate the state-of-the-art fault-tolerant DNN accelerator architectures with particular emphasis on their applicability to safety-critical autonomous systems in industry. The survey investigates architectural perspective, fault modeling, and platform-level trade-offs, runtime resilience, validation practices, and certification readiness, following a PRISMA methodology with evidence-driven synthesis and unbiased study selection. Database searches across IEEE Xplore, Scopus, and Web of Science identified 200 records, of which 82 studies were included based on predefined inclusion and exclusion criteria emphasizing industrial safety-critical relevance, fault modeling at the hardware level, and the implementation at the architectural level. The results indicate that there was a clear shift from traditional redundancy-based approaches to cross-layer and adaptive approaches that provide better trade-offs between performance, reliability, and hardware overhead. The current studies presented are based on simplified fault models, incomplete validation procedures, and limited consideration of system-level and certification needs, which often do not consider critical failure modes such as Silent Data Corruption (SDC). This has resulted in a significant gap between research-level solutions and industrial deployment requirements. This survey underscores the need for scalable, integrated, and certification-aware design approaches to help connect fault modeling, architectural resilience, validation, and safety assurance to develop reliable and deployable DNN accelerator systems for next-generation industrial safety-critical autonomous applications.

Keywords: fault-tolerant; accelerators; hardware; fault models; safety-critical systems; cross-layer; resilience; certification

Academic Editor: Yannis Manolopoulos

Received: 18 May 2026

Revised: 23 June 2026

Accepted: 29 June 2026

Published: 2 July 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

The rapid expansion of Artificial Intelligence (AI) has significantly transformed modern safety-critical systems in Industry 4.0 and in advanced industrial settings [1]. Such systems are becoming more and more data-driven in decision-making to execute safety-related tasks in areas like autonomous vehicles, industrial robotics, medical diagnosis, and intelligent transportation [2–5]. Deep Neural Networks (DNNs) play a central role because they facilitate perception, prediction, and control capabilities that have a direct impact on the system safety and reliability. Nonetheless, the adoption of AI into safety-critical systems introduces challenges that are beyond conventional safety engineering. The DNN-based components are non-deterministic and data-dependent, unlike deterministic systems, so verification and validation of such components is complex in nature. State-of-the-art DNN models can be made with millions of parameters and require massive parallel matrix and multiply-accumulate operations for inference, adding hardware complexity and vulnerability to errors. Consequently, specific accelerators like GPUs, FPGAs, ASICs, and TPUs are frequently employed to increase throughput, minimize latency, optimize energy efficiency, and enable real-time inference in industrial safety-critical autonomous systems. However, in the case of computationally intensive DNN workload, these accelerators can be susceptible to transient faults, permanent faults, timing errors, and aging-related degradation that can impact the reliability of the inferences and the safety of the industrial autonomous system [6–9].

Faults in DNN accelerators can affect the processing elements, memory arrays, datapaths, and communication buses. These include transient faults caused by radiation-induced soft errors, permanent faults due to hardware degradation, timing faults caused by voltage and/or frequency instability, and bit-flips in memory that alter the stored weights and activations [10]. The computations of the DNN inference are quite interconnected in each layer, and the errors introduced in one layer propagate to the others and eventually result in errors or uncertainties in the system's prediction or operation. One of the most serious failure modes of DNN accelerators is Silent Data Corruption (SDC), in which the system fails to detect that computation results are incorrect. However, in DNNs for safety-critical applications, small numeric errors or hardware defects could be silently mis-propagated through the neural computation and lead to wrong and unsafe decisions at the system level. Furthermore, SDC frequently occurs due to errors that happen in DNN accelerator hardware components such as accelerator memories, registers, and arithmetic units, making it very relevant to fault-tolerant DNN accelerator design.

Although the utilization of AI technologies is high [11,12], the maturity of AI-based systems in terms of safety is at the initial level [13–15]. For instance, DNN accelerators are utilized by autonomous driving platforms like NVIDIA DRIVE to detect, perceive, and decide on objects in real time. In such systems, hardware failures can introduce inference errors which can result in inaccurate perception and even unsafe driving actions [16]. Safety entails following organized lifecycle activities, such as hazard identification, verification, validation, and the creation of safety cases, which are stipulated in standardized requirements, such as IEC 61508 [17], ISO 26262 [18], and DO-254 [2,3,19,20]. Nevertheless, the current methods tend to be methodologically weak, non-scalable, and not in line with the certification practice, which leaves a consistent gap between the development of academic research and its integration in industrial safety-critical systems. Meanwhile, the position of AI in safety engineering has shifted, no longer as an adjunct to safety analysis, but now, more commonly as a component of safety-critical control loops, and, as of relatively recently, itself as an object of safety assurance. This development represents a paradigm change in the design of the system, in which AI is not viewed as a complementary aspect, but one of the key safety-critical elements that should meet high reliability and assurance standards. Simultaneously, the work of autonomous systems demonstrates the

importance of clear fallback plans and gradual transitions to safe or degraded operation, to ensure acceptable risk levels within the faulty conditions [21].

Although these developments have occurred, there are still a number of significant constraints. First, most of the current literature focuses on performance-based measures, including accuracy, latency, and energy efficiency, whereas the factors of reliability, certification, and fault tolerance are relatively unexplored [7,22]. Second, the solutions available are commonly fragmented, focusing on individual layers of the problem, like algorithm-level robustness or hardware-level protection, without taking a cross-layer view [9,23]. Third, fault modeling is normally confined to simplified scenarios, especially single transient faults, but realistic conditions like multi-fault events, common-cause failures, and long-term degradation are largely ignored [6–8]. Lastly, the issue of safety behavior enforcement (e.g., fail-safe and fail-operational modes) and certification readiness are not adequately covered and restrict the generalizability of existing methods in industrial settings [3,21]. This highlights the need for a systematic and evidence-based study that summarizes the available literature, outlines the main trends, and assesses their appropriateness to the conditions of industrial safety-critical systems. It is against this background that this paper will provide a systematic review of fault-tolerant DNN accelerator architectures, specifically hardware-level and cross-layer resilience mechanisms.

To our best knowledge, this research work is the first systematic survey of fault-tolerant DNN accelerator architectures that analyzes a combination of architectural, fault modeling, and platform-level trade-offs, runtime resilience, validation methodologies, and certification perspectives, specifically in the context of industrial safety-critical systems. The main contributions of this work are as follows:

- (i) A systematic PRISMA-based review of fault-tolerant DNN accelerator architectures for next-generation industrial safety-critical autonomous systems.
- (ii) A taxonomy of fault-tolerant DNN accelerator architectures based on architectural patterns and protection scope.
- (iii) A comprehensive analysis of hardware fault models and fault injection techniques.
- (iv) An evaluation of performance–reliability trade-offs across different implementation platforms.
- (v) An in-depth investigation of runtime detection, recovery, and safety behavior mechanisms.
- (vi) A critical assessment of verification, validation, and certification readiness, highlighting key gaps and future research directions.

The work contributes to the field of reliability of AI and responds to the requirements of industrial safety. It shows how AI reliability studies can be implemented into a next-generation DNN accelerator platform that is not only efficient, but also reliable, certifiably safe, and industrially applicable. Figure 1 illustrates a high-level architectural comparison to offer an intuitive insight into the underlying distinctions between traditional hardware-level fault tolerance and DNN-based model-level fault tolerance.

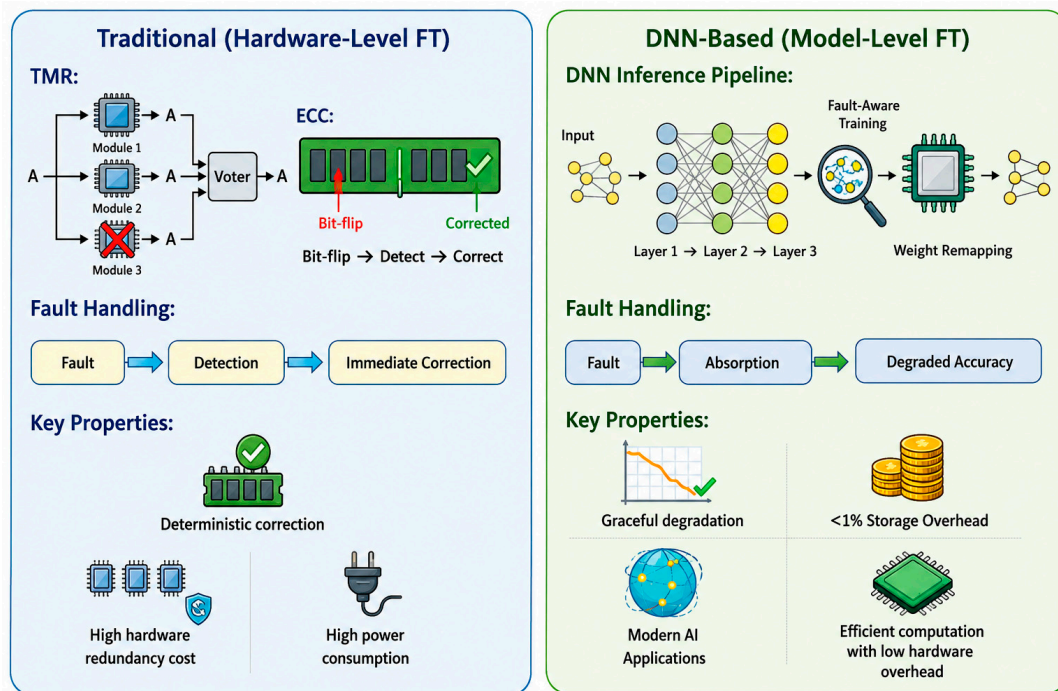


Figure 1. Traditional vs. DNN-based fault tolerance (architectural perspective).

The rest of this paper is structured as follows. Section 2 outlines the methodology of research, the systematic review, and the criteria used to select the studies. Section 3 shows the analytical framework and notes the results with regard to the specified research questions (RQ1-RQ7). Section 4 covers the cross-RQ synthesis, limitations, and future research directions. Lastly, Section 5 is the conclusion of the paper.

This survey is organized around seven research questions (RQ1 to RQ7) as follows:

- (RQ1): What architectural design strategies are used in fault-tolerant DNN accelerators?
- (RQ2): Which kinds of hardware failures are taken into account and how do they get modeled?
- (RQ3): What are the performance, hardware cost, and reliability trade-offs at the platform level?
- (RQ4): What mechanisms are in place for runtime fault detection and recovery?
- (RQ5): How well do the existing methods and techniques meet the safety-critical system requirements?
- (RQ6): How do these systems get validated and are they reproducible?
- (RQ7): What is the level of certification readiness of existing approaches?

2. Materials and Methods

This systematic survey was conducted in accordance with the PRISMA 2020 guidelines and checklist as shown in the Supplementary Materials. The process of selecting studies was carried out in accordance with the inclusion and exclusion criteria and the structured evidence-based methodology.

2.1. Search Strategy

Three major scientific databases (IEEE Xplore, Scopus, and Web of Science) were used in a systematic literature search. The search process aimed at finding studies related to fault-tolerant Deep Neural Network (DNN) accelerator architecture and its application

to safety-critical systems in industry. In order to achieve transparency and reproducibility, search strings were developed and adapted to the syntax requirements of each database, but they kept the same conceptual scope.

- IEEE Xplore:
 (“fault tolerance” OR “fault-tolerant” OR reliability OR resilience OR dependable OR “soft error” OR “fault injection”) AND (“deep neural network” OR DNN OR CNN OR “deep learning”) AND (accelerator OR “DNN accelerator” OR FPGA OR ASIC OR “systolic array” OR “compute-in-memory”) AND (“safety-critical” OR industrial OR automotive OR robotics OR transportation OR medical)
- Scopus:
 TITLE-ABS-KEY ((“fault tolerance” OR “fault-tolerant” OR reliability OR resilience OR dependable OR “soft error” OR “fault injection”) AND (“deep neural network” OR DNN OR CNN OR “deep learning”) AND (accelerator OR “DNN accelerator” OR FPGA OR ASIC OR “systolic array” OR “compute-in-memory”) AND (“safety-critical” OR industrial OR automotive OR robotics OR transportation OR medical))
- Web of Science:
 TS = ((“fault tolerance” OR “fault-tolerant” OR reliability OR resilience OR dependable OR “soft error” OR “fault injection”) AND (“deep neural network” OR DNN OR CNN OR “deep learning”) AND (accelerator OR “DNN accelerator” OR FPGA OR ASIC OR “systolic array” OR “compute-in-memory”) AND (“safety-critical” OR industrial OR automotive OR robotics OR transportation OR medical))

The database search was conducted between October 2025 and March 2026. Studies published from January 2017 to March 2026 were considered for inclusion, subject to the predefined eligibility criteria. This publication range was selected to capture both foundational and recent developments in fault-tolerant DNN accelerator architectures and their application to industrial safety-critical systems.

The database search was conducted between October 2025 and March 2026. Publication between January 2017 and March 2026 according to the pre-established criteria was considered. The range of this publication was chosen to reflect core and recent advances of fault-tolerant DNN accelerator architectures and their use in industrial safety-critical systems. Only peer-reviewed studies published in English were considered. To find studies that were not retrieved by the initial database searches, additional manual searches, iterative keyword revisions, and backward reference screening were conducted. The information collected was thereafter filtered based on the inclusion and exclusion criteria established prior to the process, and duplicate data were eliminated before full-text screening.

2.2. Inclusion and Exclusion Criteria

In order to achieve methodological rigor and consistency with the objectives of this systematic review, a set of explicit inclusion and exclusion criteria was defined a priori and applied consistently throughout the study selection and screening process.

2.2.1. Inclusion Criteria

A study is included if it meets all of the following conditions:

1. Fault-Tolerant Mechanism (Core Requirement): The study proposes, implements, or evaluates a fault-tolerant architecture or mechanism for deep neural networks (DNNs) at the hardware level or cross-layer level (e.g., combination of hardware and algorithmic resilience methods).
2. Industrial Safety-Critical Relevance: The study is relevant to, or applicable to, industrial safety-critical systems, such as, but not limited to, autonomous vehicles, industrial robotics, safety-critical embedded platforms, and real-time decision-making sys-

tems. The studies including only generic or non-industrial contexts, without clear applicability to safety-critical industrial systems, are excluded.

3. **Hardware-Oriented Implementation:** The suggested implementation is tested on, or has architectural insights to, hardware platforms like FPGA-based systems, embedded accelerators, or ASIC-based DNN accelerators.
4. **Explicit Hardware Fault Model:** The study explicitly considers hardware-based faults, such as transient faults (e.g., bit-flips, Single Event Upsets), permanent faults, or configuration memory errors.
5. **Experimental Validation:** The study includes quantitative evaluation using fault injection campaigns, RTL-level or architectural simulation, emulation, or real hardware experiments, and quantifiable effect on accuracy, reliability, or system performance.

2.2.2. Exclusion Criteria

A study is excluded if it meets any of the following conditions:

1. **Software-Only or Algorithmic Robustness:** The paper concentrates solely on adversarial robustness, model-level advancements, or training/inference optimization without addressing hardware-induced faults.
2. **Absence of Fault-Tolerant Mechanism:** The study only analyzes the faults, characterizes vulnerability, or examines reliability without a proposed fault-tolerant solution or mitigation mechanism. Such studies may be retained as supporting references but cannot be regarded as primary works.
3. **Non-Industrial Context (Domain Misalignment):** The research is confined to space-oriented environments (e.g., radiation- focused analysis without industrial mapping) or general-purpose computing systems lacking safety-critical constraints.
4. **Lack of Hardware-Level Consideration:** The research is not based on the hardware level behavior, accelerator level analysis, or FPGA/embedded implementation.
5. **Insufficient Experimental Validation:** There is no quantitative verification of the study, including fault injection, simulation, or empirical analysis of system reliability.

This review is mainly based on hardware faults rather than software-related faults, since the goal is to understand the hardware-centric reliability challenges related to DNN accelerator architectures and their computing platforms. Processing elements, datapaths, and on-chip memories can suffer from transient and permanent faults, timing violations, memory bit-flips, etc., which can cause Silent Data Corruption (SDC), incorrect inference results, and unsafe system behavior in safety-critical applications. Software faults, on the other hand, are typically those involving software engineering, cybersecurity, or algorithmic robustness, such as programming bugs, attacks, and software misconfigurations, and they fall into a different category of reliability and security. Furthermore, hardware faults can be very significant in DNN accelerators since very large-scale parallel computation and huge data reuse may cause the propagation of faults throughout multiple layers of the network. For this reason, this review only considers hardware-level and cross-layer fault tolerance mechanisms, thereby limiting the scope of the review to a focused and technically consistent analysis for industrial safety-critical systems.

2.3. Quality Assessment and Evidence Synthesis

The studies were filtered with the specified inclusion and exclusion criteria and then analyzed with RQ1-RQ7 evidence-based classification framework. No formal quality assessment tool (CASP, JBI, or AMSTAR) was used because the review was conducted primarily to compare and synthesize architectural, reliability, validation, and certification parameters without methodological appraisal with formal quality assessment tools based on standardized scoring systems. To minimize potential sources of bias, predefined eligi-

bility criteria, independent screening, duplicate removal, and multiple rounds of full-text evaluation were used.

The Technology Readiness Levels (TRLs) and certification readiness were evaluated by a qualitative evidence-based rubric based on reported validation environment, implementation platform, fault-injection methodology, validation rigor, and deployment evidence. Studies that were mainly based on software simulation or fault-injection frameworks were identified as TRL 3–4 (laboratory validation). FPGA prototypes, RTL level validation, and hardware-aware fault injection were considered as TRL 4–7 (system-level validation). Industrial deployment evidence, silicon validation and explicit consideration of functional-safety requirements were considered as TRL 7–9 (industrial deployment readiness). To achieve transparency and comparability of the process of assessing the maturity, the same rubric has been used across all included studies.

2.4. Study Selection and Analysis Framework

The process of selecting the studies was performed according to the PRISMA 2020 guidelines shown in Figure 2. A total of 200 records were first identified from various databases. Removing the 6 duplicate records left 194 studies to be screened. After title and abstract screening, 74 records were discarded because they were not relevant to industrial safety-critical systems or because they did not include hardware-level fault tolerance. Full-text review was done on the remaining 120 studies. Thirty-eight studies were excluded after applying the predefined inclusion and exclusion criteria. Finally, 82 studies were selected for the systematic review and were used for further analysis. The screening process was conducted in two stages by the authors using an Excel work-sheet for duplicate removal and record management. Records retrieved were first checked for the title and abstract using inclusion and exclusion criteria. Studies that did not focus on hardware-level fault tolerance, DNN accelerator architectures and industrial safety-critical applications were omitted. In the second stage, the full texts of the remaining studies were reviewed to confirm eligibility. Screening decisions were independently reviewed by the authors, and any uncertainties were resolved through discussion and consensus. Several screenings were conducted to increase consistency and mitigate the possibility of selection bias.

All included studies had a structured data extraction process. For each study, information related to publication year, target application domain, DNN model, accelerator architecture, hardware platform, fault model, fault tolerance mechanism, validation methodology, performance overhead, reliability metrics, safety-related characteristics, and certification evidence was recorded. The information extracted was later analyzed, classified, and mapped to the evidence-based analytical framework used by RQ1–RQ7, which covers architectural design, fault modeling, platform-level trade-offs, runtime behavior, validation, and certification readiness. The level of confidence in the collected evidence was qualitatively analyzed using the validation methodology, industrial readiness, reproducibility, and characteristics of the certification in the studies included. It should be noted that the studies included are not mutually exclusive among the research questions. The number of studies associated with each RQ can be greater than the number of studies included in the study because a single study could be relevant to more than one RQ.

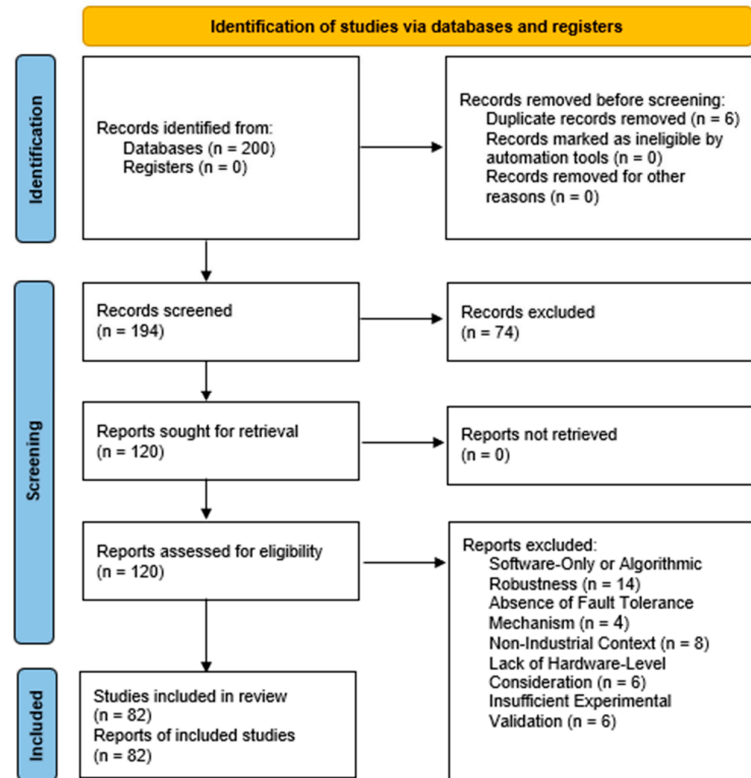


Figure 2. PRISMA 2020 flow diagram of the literature selection process for fault-tolerant DNN accelerator.

3. Research Question Analysis

This review uses an analytical framework to ensure a structured and evidence-based analysis. Each research question (RQ1–RQ7) is considered a different dimension of evaluation. The framework divides the literature into seven mutually reinforcing perspectives, such as architectural design, fault modeling, platform trade-offs, runtime resilience, safety-critical requirements, validation maturity, and certification readiness. The assessment of all dimensions is carried out with the help of special comparison tables, which guarantee the required consistency, traceability, and distinction of analyses. Classifications are based on the explicit evidence in the original studies. In the case of missing information, one would mark it as not reported (NR) to be transparent. On the whole, this method allows a comparative evaluation of fault-tolerant DNN accelerator architectures that are consistent and repeatable. It is also used to determine gaps in research and to assist in creating certification-aware architectures.

3.1. RQ1: Architecture and Fault Tolerance Strategy

The architectural features and fault tolerance approaches used in current DNN accelerator architectures for industrial safety-critical systems are explored in RQ1. The reviewed studies were compared to each other in terms of architectural organization, fault tolerance strategy, protected scope, application domain, and industrial applicability to determine the prevailing architectural trends and the prevailing resilience patterns. The fault-tolerant DNN accelerator systems are discussed based on the properly selected number of works and supported by the other analysis elements. The findings reveal a definite transition to more adaptive and cross-layer resilience designs that are also focused on application needs as opposed to traditional designs that rely on redundancy. Based on the reviewed studies, three main architectural paradigms can be identified: (i) redundancy-

driven architectures, (ii) detection and mitigation architectures, and (iii) cross-layer adaptive architectures. In order to give a systematic review of the state of the art of the field, Figure 3 shows a taxonomy of fault-tolerant DNN accelerator designs. This taxonomy categorizes the available approaches based on their fundamental design and protection strategies. It also represents the development of the field, as the traditional methods of redundancy-driven solutions are replaced by more modern detection-driven and adaptive methods. On the whole, this trend reflects a shift towards basic reliability via replication to more scalable and application-aware resilience designs.

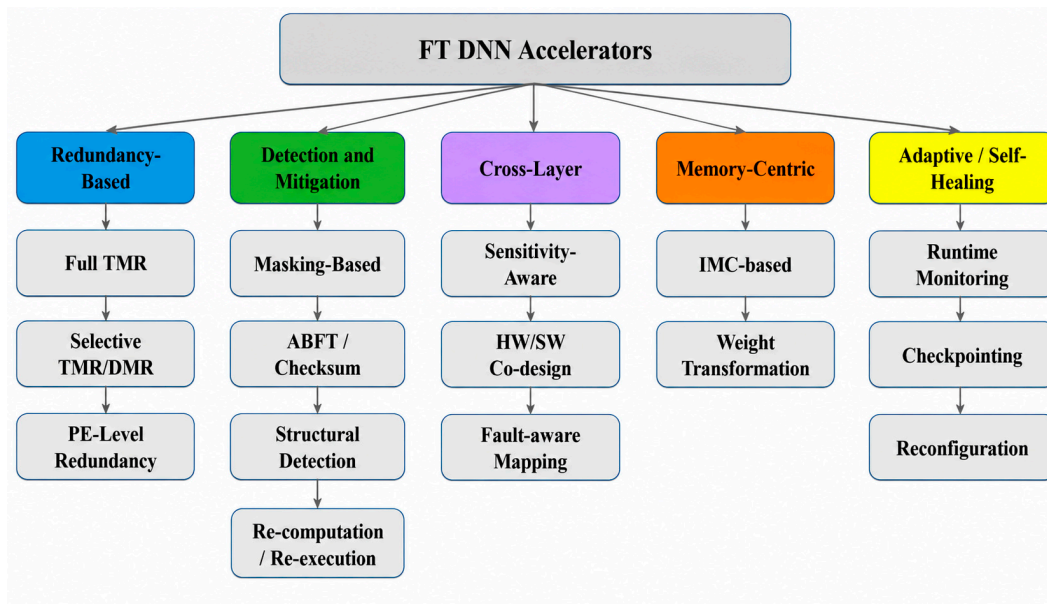


Figure 3. Taxonomy of fault-tolerant DNN accelerator architectures based on architectural patterns and protection strategies.

There are five major groups of fault-tolerant architectures, as illustrated in Figure 3. They are redundancy-based, detection and mitigation, cross-layer, memory-centric, and adaptive or self-healing. Methods based on redundancy offer high reliability, but are expensive in terms of hardware. Detection-based solutions are less heavy, yet their protection is also reduced. Cross-layer designs enhance efficiency because it takes the application into account. Memory-centric designs aim to identify faults in stored weights and adaptive designs assist in recovery of dynamism during execution. Accelerators based on systolic arrays are common at the hardware level [24], as seen in [25–27]. Their periodic structure allows the containment of faults at the processing element level and mitigates the spread of errors. In one instance, FORTALESIA [25] provides runtime reconfiguration to convert between a dual and triple redundancy. FSA [26] is an algorithm that recovers erroneous outputs at a minimal performance cost through re-computation. Redundancy-based designs like Selective Triple Modular Redundancy (STMR) [28] enhance reliability by duplicating important calculations. They are more efficient than full TMR, yet area and power are still increased, restricting scalability. More recent approaches use finer-grained redundancy. As an example, D2NN [29] uses neuron-level duplication with fault sensitivity analysis to strike a more optimal balance between reliability and overhead.

Another group of approaches looks at simple fault detection and correction inside the datapath. For example, MOZART/MOZART+ [30,31] use zero-masking and online testing to find faults inside individual processing elements (PEs) without stopping the system. A similar concept is the Detect-and-Replace (DR) architecture [27], in which the faulty PEs are monitored and replaced in a short period, but with a very minimal latency.

The more recent work is not just based on hardware. Rather, it is a combination of levels, commonly referred to as cross-layer fault tolerance. For example, HyCA [32] and FlexHyCA [33] only secure the most critical parts of the system. This reduces the amount of redundancy, but still keeps good reliability. Saliency-based methods also exist [34]. This approach manages permanent faults by remapping or using bit-level correction methods without retraining the neural network, thereby reducing computational overhead and enhancing the practicality of the system during runtime for industrial safety-critical applications. Generally, the majority of these strategies can be classified as redundancy-based or cross-layer techniques. This is also an indication that research work and real industrial systems are still out of touch with each other. Cross-layer solutions appear to be more realistic, as they are more balanced in terms of reliability, expenses, and scalability.

One more direction is memory-centric and in-memory computing (IMC), particularly of binary neural networks (BNNs) [35,36]. These methods deal with faults in stored weights using techniques like bit-flipping or redundancy encoding, without retraining. They are also energy efficient and yet the usage has not been fully exploited due to design limitations. A more recent concept is self-healing or adaptive systems [37,38]. An example is Auto-Healer [39], which applies checkpointing and re-execution with runtime detection to auto-heal faults. This form of design is more in line with what the industrial standards (such as ISO 26262) are demanding, particularly in real-time systems; in addition, it automatically recovers the temporary as well as permanent faults.

In general, hardware redundancy cannot be considered sufficient any longer. The majority of the current development is that of hybrid and cross-layer solutions. Regarding applications, autonomous driving and vision tasks are the most common, i.e., the detection and classification of objects. This is mainly because datasets are available and safety is important in these areas. Other industrial applications, such as robotics or real-time control systems are still unexplored. The other aspect is that most of the research utilizes benchmark data rather than actual data. This increases the difference between real deployment and research. Hence, the application-aware fault tolerance is not thoroughly researched yet. Lastly, the majority of these still have low Technology Readiness Levels (TRL 3–4), and there is little actual industrial validation done. This is important in safety-critical systems, where standards like ISO 26262 require predictable behavior and proper validation. Table 1 summarizes the main categories of fault-tolerant DNN accelerator architectures, including their protection strategies, strengths, and limitations.

Table 1. Taxonomy of fault-tolerant architectures in DNN accelerators.

Category	Architecture Type	Representative Papers	FT Strategy	Technical Assumptions	Coverage Limits and Failure Modes	Hardware Costs (Area/Power/Latency)
Redundancy-Based	Selective/Full TMR	[28,40]	Replication + Voting	Fault-free voters; no common-cause failures (CCF).	Limits: Poor area scaling. Failure: Voter is single point of failure.	High: >200% for full; 21–56% for selective.
PE-Level Protection	Systolic Array FT	[25–27]	Re-compute/Replace/Redundancy	PE localized containment; available spare units.	Failure: WS dataflow propagates errors across channels.	Moderate: FSA 3.1% area; FORTA-LESA 6x < TMR.
Masking-Based	Output Masking	[30,31,41]	Zero-masking/Dropout	DNN resilience to zeroed outputs.	Limits: Fails in small models (LeNet-5). Failure: MSB faults critical.	Low: <8% area overhead.

Detection-Based	ABFT/Check sum	[42,43]	Arithmetic checks	Single-fault per layer; linear arithmetic.	Limits: No ReLU/Pooling coverage. Failure: Misses MBUs.	Runtime: 4–8% to 23% execution overhead.
Cross-Layer	Hybrid FT	[32,33,44]	Sensitivity-aware protection	Non-uniform vulnerability; bit/neuron priority.	Limits: Repair capacity limited by DPPU size.	Efficient: 5–40% area based on target.
Memory-Centric	IMC/BNN	[35,36]	Weight transformation	Stuck-At Faults (SAFs) dominate over logic errors.	Limits: No transient logic error coverage. Restricted to BNNs.	Negligible: <3% energy; <7% area.
Self-Healing	Adaptive systems	[39]	Re-execution + checkpointing	Stable hardware checkpoints; OODA loop diagnosis.	Failure: Complex diagnosis may exceed safety time (FTTI).	Minimal: <2% resources; 40–120 ns MTTR.

In summary, as synthesized in Table 1, a clear technological shift from deterministic redundancy (TMR), which suffers a hardware overhead of 200%, to adaptive and cross-layer strategies can be seen. Detection-based methods, such as ABFT, offer lightweight protection, but our critical analysis shows that these methods have a significant coverage limit, since they make an assumption of single fault which is often not met in harsh industrial environments where MBUs are present. In addition, masking-based techniques, which are relatively inexpensive (<8% area), have an inherent failure mode in small-scale safety networks (such as LeNet-5), where masking a single faulty PE can remove an entire classification category. Self-healing architectures like Auto-Healer are then well suited for certification-oriented and ASIL-D oriented systems, with almost instant recovery and minimal impact on resources.

3.1.1. Comparative Analysis of Fault-Tolerant Architectures

Industrial fields with safety-critical applications have some additional requirements. A more in-depth analysis of fault-tolerant architectures shows that these differ significantly in reliability, overhead, scalability and adaptability, which is directly relevant to their suitability for such applications.

Redundancy vs. Re-Computation-Based Architectures

Currently, the employment of redundancy-based architectures compares to re-computation-based architectures. Older forms of redundancy-based mechanisms (most importantly, Triple Modular Redundancy (TMR)) [28,40] provide deterministic fault masking using replication and majority voting. This renders them applicable with high-integrity systems. This, however, incurs a cost by adding large area, power, and latency overheads, especially in large-scale accelerators. The re-computation-based architectures are on a different path in such a way that FSA-based systems like FSA [26] and FORTALESA [25] are selective in re-evaluating corrupted products, by either employing redundant processing units or dynamic redundancy grouping. Therefore, this minimizes resource overhead, and acceptable reliability is achieved. Consequently, re-computation techniques can typically provide a more balanced of reliability and efficiency trade-off than TMR, particularly in larger systems.

Masking-Based vs. Detection-Based Strategies

Some strategies are aimed at preventing the spread of the error but not at the fault detection. An example is MOZART/MOZART+ [30,31] nullifying corrupted neuron outputs, useful in containing fault propagation, and allowing graceful degradation at low over-

head. Conversely, detection-based methods, e.g., ABFT methods [42,43], are based on arithmetic invariants or checksum-based validation to detect faults. Such techniques offer extensive coverage in detection, and in certain instances, complete coverage of SDC is achieved. Nevertheless, they need more computations, and are frequently constrained by simplifying assumptions, e.g., single-fault cases. Overall, masking measures are cheaper, whereas detecting-based techniques have more robust fault observability but are more complex.

Cross-Layer Architectures vs. Pure Hardware Approaches

In pure hardware fault tolerance, datapaths and processing elements are the primary targets of protection, and the application sensitivity is not taken into account. Consequently, these methods tend to be over-dimensioned, and cross-layer architectures take a more selective approach. Indicatively, FlexHyCA [33], HyCA [32], and SED+SLH [44] integrate application-aware sensitivity profiling to implement protection on only the critical bits, neurons, and layers. Similarly, model-level selective redundancy enhances fine-grained protection [29]. Reconfigurable accelerators have been proposed [45], a combination of redundancy and runtime adaptability, and kernel-level selective duplication [46], with lower overhead, but perhaps not necessarily tightly integrated hardware. The cross-layer fault tolerance concept takes protection into multiple layers of abstraction (hardware, runtime, algorithmic) to increase the reliability at minimum cost. In contrast, adaptive fault tolerance mechanisms perform fault protection, recovery, or resource allocation at run-time based on fault conditions and system behavior. Overall, cross-layer designs have a superior trade-off between reliability and cost which makes them well-suited in the industry. Fine-grained mechanisms are used to implement fault-tolerant architectures that are used to protect small, sensitive components of the architecture, like individual bits, individual neurons, individual weights, individual processing elements, etc. These strategies are selective and result in a lower hardware overhead and an improved resource efficiency. Coarse-grained techniques, on the other hand, provide protection for larger architectural components, such as whole layers of architecture, modules, or replicated accelerator blocks, usually by full redundancy or by module-level replication. Coarse-grained methods tend to give better deterministic protection, but with much greater area, power, and performance overheads. Hence, protection strategies at the finer grain level have become more popular in recent cross-layer architectures to achieve a more balanced trade-off between reliability, scalability, and hardware efficiency in industrial safety-critical systems.

Memory-Centric vs. Computation-Centric Fault Tolerance

Memory-centric architectures, like IMC-based BNN accelerators [35,36], shift fault tolerance schemes out of the computation units and into the weight storage. The fault effects are reduced by employing methods such as weight transformation (e.g., BNN-Flip). These solutions are energy-efficient and light. They are, though, not quite as generalized as the computation-centric fault-tolerance approaches, which can be used within a greater architecture space.

Static vs. Adaptive (Self-Healing) Architectures

Traditional fault-tolerant designs are grounded on the application of the fixed schemes of protection which are not reliant on the prevailing conditions. On the other hand, adaptive architecture, such as Auto-Healer [39], reacts to runtime failure by combining the monitoring, checkpointing, and recoveries. This creates dynamic detecting, diagnosing, and fixing. These are more associated with the functional safety requirements (e.g., ISO 26262), especially in real-time environments. Nonetheless, their ability to scale to large DNN models and hard timing limits is still largely unexplored. Table 2 gives this

comparison in detail and outlines some of the large differences in reliability, recovery speed, scalability, and compatibility with industry.

Table 2. A comparative study of fault-tolerant architectures in DNN accelerators.

Approach	Representative Studies	Reliability Behavior	Recovery Speed (MTTR)	Scalability for Large Models	Industrial Suitability
TMR-Based	[28,40]	Deterministic Masking: Masks all errors but lacks fault management.	Instantaneous: No detection cycle needed; masking happens in real-time.	Low: Area increases exponentially with model size.	Limited: Infeasible for edge FPGAs due to resource exhaustion.
Re-computation-Based	[25,26]	Selective Correction: Dynamically recalculates only faulty PE outputs.	Cycle-based: Depends on fault density and spare unit availability.	High: Balances throughput and area better than TMR.	High: Viable for systems allowing minor latency spikes.
Masking-Based	[30,31]	Graceful Degradation: DNNs absorb errors by zeroing faulty neurons.	N/A: No recovery; relies on intrinsic model resilience.	High: Effective for large redundant networks (VGG-16).	Moderate: “Fail-safe” only; unsuitable for small-scale safety networks.
ABFT-Based	[42,43]	Arithmetic Checking: Uses linear invariants to detect/correct SDC.	High: Requires checksum generation and comparison cycles.	Limited: Performance drops as matrix shapes become non-square.	Moderate: Fails to cover non-linear layers like ReLU/Pooling.
Cross-Layer	[32,33,44]	Sensitivity-Aware: Protects only “critical” bits/neurons based on gradients.	Varies: Detection/correction depends on protection level chosen.	Very High: Optimized for resource-constrained deployments.	Very High: Optimal for complex safety-critical systems.
Memory-Centric	[35,36]	Static Mitigation: Focuses on permanent SAFs in NVM arrays.	N/A: Applied offline; does not cover runtime transient errors.	Limited: Currently restricted to BNN/TNN quantized models.	Low to Moderate: Emerging for edge-AI with SAF-prone memories.
Adaptive/Self-Healing	[39]	Autonomous Recovery: Uses OODA loops and hardware check-pointing.	Ultra-Fast: 40 ns (transient) to 120 ns (permanent).	Uncertain: Performance overhead for very dense check-pointing is unexplored.	High: Aligned with ASIL-D functional safety requirements.

Compared to traditional TMR techniques, as demonstrated in Table 2, classical TMR-based approaches offer deterministic fault masking and instantaneous recovery but have a significant hardware overhead and are not well suited for resource constrained edge deployments. Re-computation techniques enhance scalability by selectively correcting faulty processing elements, while masking-based approaches provide low-cost protection but prove to be less effective in small scale safety-critical networks. Although ABFT-based methods offer arithmetic error detection and correction functions, they are not widely applicable due to the lack of support for non-linear arithmetic operations and the low scalability for non-regular matrix structures. Cross-layer techniques, on the other hand, are better suited to delivering a more balanced reliability-scalability-industrial suitability system, through selectively protecting critical parts of the system. Adaptive and self-healing architectures are further extensions of this capability, with autonomous means of diagnosis and recovery, providing rapid recovery and strong alignment with the requirements of high-integrity safety-critical systems. To further estimate these trade-offs, Figure 4 shows a multi-dimensional comparison of significant evaluation measures.

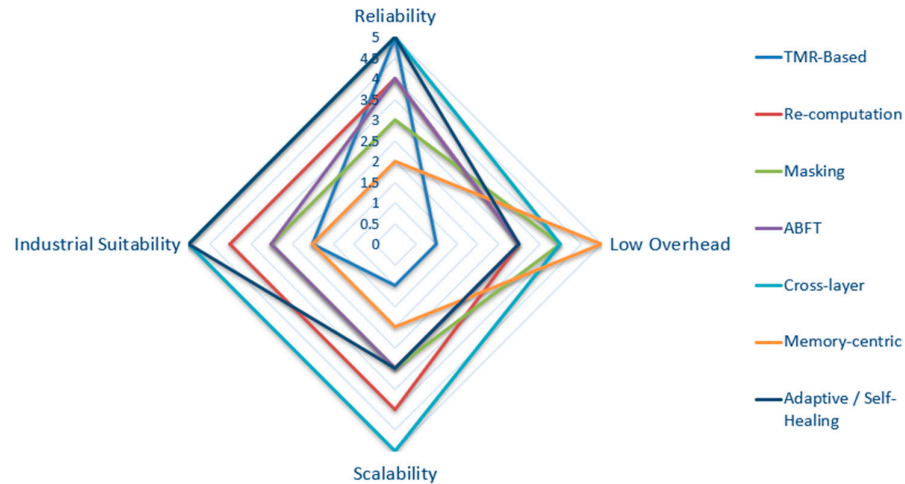


Figure 4. Multi-dimensional trade-off analysis of fault-tolerant DNN accelerator architectures.

3.2. RQ2: Fault Model and Mitigation Perspective in DNN Accelerators

RQ2 examines the fault models, fault manifestations, and mechanisms used to mitigate faults in fault-tolerant DNN accelerators for industrial safety-critical systems. The reviewed papers were compared to each other with respect to fault type, fault propagation, mitigation strategy, fault injection process, and industrial applicability to highlight the prevalent reliability issues and resilience trends. The analysis indicates a heterogeneous but structured environment of fault origin, manifestations, and resilience mechanisms. The faults can broadly be categorized as transient, permanent, and timing-related, which have different propagation characteristics and reliability implications, as shown in Figure 5. The studies reviewed indicate that fault tolerance entails close collaboration of fault modeling, fault detection, and mitigation and not individual protection techniques.

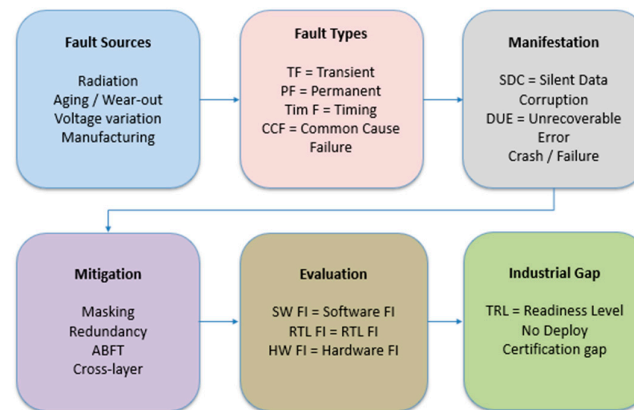


Figure 5. End-to-end framework of fault modeling, manifestation, mitigation, and evaluation in DNN accelerators.

3.2.1. Fault Modeling

On the fault modeling level, transient faults, especially Single Event Upsets (SEUs) and random bit-flips are the most common faults, most often in the form of Silent Data Corruption (SDC) which spreads via intermediate activations and degrades inference accuracy [43,44,47,48]. Recent research indicates that these errors can result in structured patterns of corruption including row-wise, column-wise, and feature-map-wise distortions that can have a much greater effect on perception tasks, including semantic segmen-

tation [49]. Cross-layer models also show not only that these patterns are random, but that there are architecture-specific spatial distributions which permit the use of high-level error models, and which can model hardware behavior, though at lower complexity in simulation. Despite the extensive use of fault injection [50,51], simplistic bit-level models (not founded on realistic error propagation) are employed in many studies. Cross-layer analysis demonstrates systemic, architecture-specific fault structures, and exhibits a disjoint between traditional models and physical hardware behavior. Furthermore, design exploration systems such as DeepAxe [52] employ random single-bit fault injection as a convenient model, which can be seen as a form of fault coverage in addition to avoiding the complexity of multi-bit fault modeling. The importance of SEUs in FPGA-based accelerators in computation modules is also confirmed in hardware-based fault injection experiments, but the faults of control components within the system are likely to cause system-level errors [53].

Some studies introduced permanent faults [51], such as stuck-at faults, which are generally associated with manufacturing errors and aging effects, which cause long-term degradation of computational units unless corrected deliberately [26,35,36]. Non-deterministic behavior to arithmetic operations, such as faults due to timing, voltage variation, overclocking errors, etc., make this even more difficult to analyze reliably [54]. Representation of fault sensitivity is also highly data-representation-sensitive, with floating-point formats becoming more vulnerable to bit-level perturbation (in particular, exponent and sign bits) than the less sensitive quantized models which provide more bounded error propagation. Nonetheless, the error can be more important due to quantization since significant weights become more sensitive to the faults, such as in radiation-tolerant works [55–57].

3.2.2. Fault Manifestation

There are mainly three major types of faults: Silent Data Corruption (SDC), Detected Unrecoverable Errors (DUEs), and system-level failures (e.g., stalls or crashes). Among them, SDC is the most prevalent failure mode, which provides erroneous outputs without any clear error indicators, thus being especially important in safety-critical applications. On the other hand, the DUE event failures and system failures provide definite indications of fault that can be used to recover.

Fault location, data representation, and network architecture are crucial determinants of the manifestation of fault. More sensitive components such as exponent bits, sign bits, and important neurons are more susceptible, and memory buffer faults are more dangerous than datapath faults due to data reuse [58,59], which enables a single fault to harm numerous computations [44,60]. Radiation experiments also indicate that the defects in memory may be burst or block errors of adjacent regions of memory, and not isolated bit-flips, and can have a significant impact on the outcomes of the inference [61].

Systematic defects of perception-based DNNs and, in particular, semantic segmentation models can result in spatially coherent errors in the prediction in the whole set of areas of the predicted scene [49]. Experiments based on architectural models show that architectural parallelism and data reuse are very important to fault propagation, and that the fault can propagate through processing units and increase the prediction error [44,62,63]. This behavior is further supported by activation-level analyses, where hardware faults induce high-magnitude activations that dominate network outputs and lead to severe misclassification if not properly bounded [64]. A less commonly studied but important feature is the existence of Common Cause Failures (CCFs), in which several components fail independently of each other due to a shared underlying cause (e.g., voltage fluctuations or correlated bit-flips). In contrast to independent fault assumptions, CCFs add both spatial and temporal correlations, which dramatically increase the effects of faults, especially in redundancy-based architectures (e.g., TMR), in which correlated faults

can fatally affect several replicas at the same time. Although CCFs are vital in safety-critical systems, explicit modeling of CCFs remains largely unaddressed in recent research of DNN accelerators [53].

3.2.3. Mitigation Strategies

DNN accelerator mitigation strategies can be broadly divided into masking-based, redundancy-based, detection-and-correction, and adaptive strategies. Zero-masking and activation clipping are masking techniques which offer low-overhead resilience by blocking corrupted outputs and allowing graceful degradation [30,31,41]. However, their effectiveness is limited under large-scale or correlated faults. Redundancy-based methods, such as TMR and ensemble techniques, achieve strong fault coverage through replication and voting but incur significant hardware overhead, which makes them hard to scale to large models [28,65]. Lightweight fault detection with moderate overhead is achievable using detection-and-correction techniques, most famously using checksums and ABFT, but such techniques frequently make assumptions like fault sparsity and fault independence [42,43]. Activation-level mitigation methods also increase resilience by preventing anomalies in intermediate computations caused by faults. One such method is the use of clipped activations to decrease high-magnitude activations which occur as a result of corrupted weights, which reduces the error propagation without retraining [63].

Further, permanent faults are effectively reduced through mitigation measures that are carried out through mapping. Other approaches like saliency-based remapping shift significant computation to non-faulty hardware resources by using hardware fault maps, with the least accuracy penalty, without hardware redundancy [34]. More advanced approaches, like cross-layer, adaptive techniques, integrate hardware-level protection techniques with algorithmic knowledge to offer selective hardening, re-execution, and dynamic restoration, at a more affordable trade-off between reliability and resource effectiveness [33,39,44].

3.2.4. Fault Injection and Evaluation

Fault testing in fault-tolerant DNN accelerators is essential and should be done through fault injection (FI). The current literature uses numerous different methods, such as software-level bit-flip injection (e.g., PyTorchFI, TensorFI), RTL-level fault injection [66], and FPGA emulation. Quick exploration of fault sensitivity can be undertaken using software-based FI, though it does not offer fidelity to fault behavior of hardware. On the other hand, RTL and hardware-based FI are more precise but more complicated and expensive in regard to overhead of the experiment. Recent resilience analysis frameworks address this gap by capitalizing on the architectural characteristics of DNN accelerators to build realistic fault propagation models, and can be scaled to evaluate with an accuracy comparable to RTL-based FI [60]. The efficiency of advanced hardware-level techniques is also increased by specific fault injection; only active logic components (e.g., LUTs) are corrupted, rather than exhaustively corrupting the bitstream. This method greatly minimizes experimental overhead as well as being able to maintain realistic fault behavior [67]. Despite these advances, a major limitation on the reproducibility and benchmarking of DNN accelerator studies is the lack of standardized methods of FI.

3.2.5. Industrial Readiness and Gaps

Industrially, most of the fault mitigation techniques are on moderate maturity levels (TRL 3–4) and only a few have been tested in relation to real-life deployment factors. Although there are studies that have given promising results under controlled conditions, only a few have in-depth assessment under safety-critical conditions, including real-time execution, certification conditions, and long-term reliability. Moreover, the existing prac-

tices of mitigation are often executed using a single type of faults, and the combination impacts of the transient, permanent, and environmental faults are rarely considered. This shortcoming demonstrates the need to have incorporated integrated fault models and integrated resilience models that have the capacity to assist multi-fault and correlated fault situations in industrial systems. This overall analysis indicates that more studies should aim to fill the gap between academia and industry work by concentrating on application-sensitive fault models, standard processes of evaluation, and application-level resilience mechanisms. This is a critical impediment to certification, as current approaches often provide empirical performance improvements without verifiable safety assurances.

To provide a systematic comparison of fault models and mitigation strategies, Table 3 provides an overview of the main features of representative studies that may be present in different types of faults.

Table 3. Comparative summary of fault modeling and mitigation strategies in DNN accelerators.

Category	Representative Works	Fault Type	Mitigation Strategy	Evaluation Method	Industrial Readiness
Transient Faults	[43,44,47,68]	SEU, Bit-flips	Range restriction, checksum, selective hardening	Software FI, RTL FI	TRL 3–4 (majority); few works reaching TRL 7–9 (industrial-grade implementations)
Permanent Faults	[26,34–36]	Stuck-at faults	Weight remapping, pruning, redundancy-based Correction saliency-driven mapping	Monte Carlo FI, hardware-aware FI	TRL 3–4 (majority); limited higher-readiness implementations (TRL 4–7)
Timing Faults	[54,69,70]	Voltage/clock errors	ABFT, re-execution, timing-aware mitigation	Physical FI, hardware testing	TRL 3–4 (majority); emerging toward TRL 4–7
Hybrid Faults	[25,33,39]	Mixed faults (including correlated/CCF scenarios)	Cross-layer [71], adaptive recovery, self-healing mechanisms	Software + RTL FI	TRL 4–7 (emerging approaches); few industrial-grade implementations (TRL 7–9)
Fault Analysis	[57,72–75]	Bit-level faults	Vulnerability analysis, sensitivity profiling	FI frameworks	Supporting (analysis-level only)

In order to confidently measure the maturity of implementation of the technologies, a Technology Readiness Level (TRL) was given for each study based on a description of the validation environment, implementation platform, and level of implementation. For this purpose, a qualitative evidence-based rubric was used. Studies were classified within TRL 3–4 (Laboratory Validation) when their evaluation relied primarily on software-based fault injection frameworks, such as TensorFI [72] and PyTorchFI [47], or cycle-accurate simulation tools such as SCALE-Sim [26], without deployment on physical hardware platforms. Most of these studies were based on metrics like inference accuracy, fault sensitivity, and Silent Data Corruption (SDC) rates [44] with datasets like MNIST and ImageNet [47,72]. System-level validation studies (TRL 4–7) were studies that showed hardware-oriented validation using FPGA prototypes, RTL-level evaluation, or hardware-aware fault injection. Well-known examples are implementations on Xilinx Zynq-7000 devices (Xilinx, San Jose, CA, USA) [54] and ZCU102 platforms (Xilinx, San Jose, CA, USA) [70] and studies using detailed RTL-based error-propagation analysis [75]. These

works offer more realistic answers to issues of performance/reliability trade-offs, timing behavior, and recovery methods when fault conditions arise. In particular, some studies reported timing-related reliability metrics such as the nanosecond-level Mean Time to Repair (MTTR) value for transient fault recovery [39].

Finally, the classification TRL 7–9 (Industrial Deployment) was reserved for studies where there was evidence of implementation at an industrial level, silicon validation, architectures oriented to deployment, or even explicit consideration of functional safety requirements. Examples include automotive-related platforms and systems that are developed based on standards like ISO 26262 [44]. In these studies, the level of certification readiness was evaluated based on the reported safety mechanisms, fault coverage, rigor of validation, and discussion of safety-related metrics such as Failures In Time (FIT) and Automotive Safety Integrity Level (ASIL) requirements, if available. These features suggest greater readiness for deployment in a real-world safety-critical application. The same qualitative TRL rubric was used throughout all reviewed studies to provide the basis for comparability and transparency of the maturity assessment process.

Table 3 indicates that transient faults are the dominate in the literature, whereas hybrid and cross-layer come up with more balanced resilience solutions. The vast majority of studies are still in their infancy (TRL 3–4), only a few more are depicted to be more ready (TRL 4–7), and only a few more prove to be industrial deployment (TRL 7–9). This means that there is a wide difference between research prototypes and industrial systems.

3.3. RQ3: Platform and Trade-Off Analysis

RQ3 explores the platforms for implementation and the associated trade-offs of fault-tolerant DNN accelerators for industrial safety-critical systems. Comparative analysis of the reviewed studies in terms of hardware platform, overhead of resources, performance impact, reliability improvement, and industrial applicability was performed to understand the practical limitations and deployment characteristics of the existing resilience techniques. Table 4 is a synopsis of platform-level trade-offs between the different fault-tolerance techniques in reliability, hardware overhead, performance, and industrial suitability.

Table 4. Condensed trade-off analysis across fault-tolerant architectures.

Approach	Reliability	Overhead	Performance	Industrial Suitability
TMR	Very High	High	Low	Limited
Re-computation	High	Moderate	High	High
Masking	Moderate	Low	High	Moderate
ABFT	High	Moderate	Moderate	Moderate
Cross-layer	High	Low–Moderate	High	Very High
Memory-centric	Moderate	Low	High	Low
Adaptive	Very High	Moderate	Moderate	High

The results show that the usefulness of fault-tolerant DNN architectures does not purely depend on the protection mechanism itself, but also on its ability to map to the target architecture with resource efficiency, runtime performance, and deployability [76]. Experimental experience also shows that the reliability directly depends on the parameters of architecture, such as the accuracy of information and the degree of parallelism. Fault resilience can be significantly improved in the case of reduced-precision implementations (e.g., INT8) by minimizing resource consumption and sensitive area, but highly parallel designs can pose a trade-off between performance and radiation sensitivity [56].

Design space exploration systems (e.g., DeepAxe) discover Pareto-optimal designs that balance the accuracy, reliability, and hardware efficiency of FPGA-based DNN accelerators [52]. In addition, resilience-aware frameworks allow the viability–performance

trade-offs to be explored in a well-formed way through quantifying the effect of failure in architectural parameters, and this can be applied to make informed design choices at the early steps of the progress [60]. This can be discussed based on the evidence presented by Table 4 and interpreted in four dimensions that are very closely interconnected: preference of platforms, overhead profile, performance preservation, and industrial readiness.

3.3.1. Platform View: FPGA Hegemony with Selective ASIC and GPU Roles

A new trend seems to be emerging in the considered literature: FPGA-based implementations constitute the most common experimental system of fault-tolerant DNN acceleration. Numerous representative works are built on Xilinx Zynq, ZCU102, Virtex UltraScale (Xilinx Inc., San Jose, CA, USA), Alinx ACU9EG (Alinx Co., Ltd., Shenzhen, China), and Intel Cyclone V devices (Intel Corporation, Santa Clara, CA, USA), reflecting a strong preference for reconfigurable logic in resilience-oriented accelerator research. This is mostly because of their controllable datapaths, architectural visibility, hardware-level fault injection, selective redundancy, and hardware-level recovery. Representative examples include Cyclone V implementations in [77,78], ZCU102 in [41], Zynq-7000/UltraScale platforms in [27,40,73,79], and high-end FPGA realizations in [39,71,80].

In comparison, the ASIC-based studies are represented in two forms predominantly. The first comprises synthesis-driven architectural evaluations at technology nodes such as 65 nm, 45 nm, 40 nm, 22 nm, 16 nm, 14 nm, and 12 nm, as seen in [25,32,33,44,65,81]. The second contains field-deployed automotive or vision SoCs, the most industrial-mature, especially [81,82]. Industrial deployment of these works is shown to be feasible, but also indicates that industrial-grade resilience is usually based on carefully engineered, domain-specific SoCs rather than generic research prototypes.

Platforms based on GPUs and CPUs are complementary. They are mostly applied to large scale fault analysis, statistical analysis, and runtime detectors studies, as opposed to deployment-oriented embedded architecture in works like [43,47,72,74,83–87]. In this way, they contribute more to fault observability and modeling rather than the certifiable system implementation.

3.3.2. Resource Overhead: Shift Toward Selective and Bounded Protection

The table demonstrates an evident shift of full redundancy to a selective, bounded - overhead resilience systems. Classical replication-based methods are still applicable but limited by high resource cost. An example of this is a 21–56% LUT overhead required by selective TMR in [28], whereas [88] documents a 3–5× overhead on protected modules. On the same note, [40] points out that full TMR may be infeasible in resource-constrained FPGA deployments, pointing to shortcomings in scalability.

Most new methods, on the other hand, attain fault tolerance at a small overhead—normally less than 10%. Examples include [77] (7.8% ALMs, 10.1% ALUTs, 3.1% registers), [30,31] (<8% area), [41] (<17% LUTs, <9% FFs), [33] (<5% to >40% depending on reliability target), [25] (~5–18% with significant reduction compared to TMR), [39] (<2% FPGA resources), [54] (<1% slice overhead), and [73] (<1% LUT/FF overhead). This trend is further supported with low-single-digit overhead by memory-centric IMC designs like the [35] and [36] designs.

This allocation highlights a key change: hardware cost is now considered as a key design constraint. The main goal is no longer the maximization of reliability, but attaining resilience without reducing area, power, or deployability. In this context, selective hardening, checksum-based protection, masking, and adaptive reconfiguration consistently outperform full redundancy.

3.3.3. Performance Impact: Toward Performance-Aware Fault Tolerance

A notable trend is that throughput and latency are maintained even with fault-tolerant mechanisms. Other works report small or near-zero runtime overheads, such as [27,30,31,39,41,44,65,77], in which the detection and recovery are either a parallel activity or embedded into an existing execution flow.

When overhead exists, it is generally bounded within acceptable limits, such as 4–8% in [43], 6–23% in [42], and 17.9% in [89]. Fault-aware designs sometimes can actually enhance performance or energy efficiency, as shown in [25,54,69]. It is worth noting that parallelism can be greatly increased to achieve higher speed, which can compensate higher radiation sensitivity, and may result in a reduced total failure rate despite the increased vulnerability of the hardware [56].

This implies a paradigm shift: fault tolerance is no longer a performance penalty, but an architecture feature that is performance compatible. The contemporary designs are efficient in both nominal and faulty conditions by taking advantage of idle cycles, hardware resource reuse, or powering on recovery only when required.

3.3.4. Reliability Impact: A Multi-Dimensional Perspective

The improvements in reliability are also claimed across multiple dimensions rather than a single unified metric. Some studies emphasize fault coverage and detection recall (e.g., [42,47,54,83,90,91]), while others focus on maintaining inference accuracy under fault conditions (e.g., [26,27,31,33,41,71,77]). Other publications measure reliability architecturally, including vulnerability reduction, FIT rate improvement, or recovery efficiency (e.g., [25,39,44,88]). Moreover, radiation-aware design research has shown that even non-trivial trade-offs, like quantization, cause an architectural design to become smaller at the expense of cross-section and more prone to critical misclassification errors [57].

This variety states that reliability in DNN accelerators is multi-dimensional in nature and encompasses detection capability, fault masking, accuracy maintenance, and recovery efficiency. As such, significant assessment requires an all-inclusive method instead of a single one.

3.3.5. Industrial Readiness: Strong Results, Limited Deployment Evidence

With the high level of performance in the experiments, the majority of studies are on the level of moderate maturity (TRL 3–4). This is true among FPGA, ASIC, and GPU-based works, and so the field has been vindicated in terms of architecture, but there is no overall evidence of large-scale industrial use. A small group of studies exhibit high readiness, especially those that have undergone silicon validation or have been deployed in the real world, e.g., [36,70,71,80–82,85,86,90]. These articles underscore the practicability of deployment, yet also underscore that resilience on an industrial grade needs to be validated at the system level, have deterministic behavior, and be certified.

This observation shows that there is a dire lack: the literature has many claims of deployability, but little deployment-grade evidence. Numerous works exhibit encouraging architectural properties, but not many of them offer the validation chain that can be fully verified and includes safety-critical systems.

3.3.6. Conclusion of RQ3

In general, the data in Table 4 shows that FPGA-based architectures have been the most effective platform to develop and evaluate fault-tolerant DNN accelerators, and it offers a trade-off between resource efficiency and preserving performance and flexibility. The highest industrial potential is shown by ASIC and custom SoC designs, although they remain limited in availability, and the analytical evaluation of most applications is made by the use of the GPU/CPU platforms which are not used in deployment.

Most significantly, it is evident that the field is now shifting from heavy redundancy toward selective, low-overhead, and performance-aware resilience mechanisms. Nonetheless, even with these developments, most solutions are at intermediate levels of readiness, indicating an acute disconnect between research solutions and commercial implementation. To close this gap, there is a need to have built-in, certification-aware, and system level-proven fault-tolerant system architectures to suit safety-critical applications.

3.4. RQ4: Runtime Detection, Recovery Mechanisms, and Safety Behavior

RQ4 aims to explore the mechanisms used to achieve runtime resilience in fault-tolerant DNN accelerators for safety-critical industrial systems with emphasis on fault detection, recovery behavior, and safety assurance in runtime. The analyzed studies have been compared in a detailed manner with respect to runtime detection capability, recovery strategy, safety-mode support, and industrial applicability to investigate the robustness of existing resilience architectures, especially the capacity to maintain a safe system behavior under runtime conditions. An analysis of runtime fault tolerance of DNN accelerators shows that resilience is not merely the presence of detection or recovery of failures, but rather the tightness between detection, recovery, and safety behavior enforcement. Three fundamental areas of concern—detection capability, recovery strategy, and safety-mode assurance—may be used to perform a systematic analysis of the literature.

3.4.1. Detection Mechanisms: From Lightweight Monitoring to Structured Hardware Checking

The work analyzed uses an extensive set of detection mechanisms, from lightweight statistical monitoring up to structured hardware-level checking. A number of works exist on the lightweight side that are based on range-based or statistical monitoring, such as checking activation thresholds [47] and symptom-based detection (SED) [44], where malfunctions in intermediate activations are used to identify the occurrence of faults. The methods have a high detection recall (mostly above 90%) at low hardware overhead, but may be constrained by accuracy in case of complex or correlated faults.

In contrast, hardware-based detection mechanisms provide stronger and more deterministic guarantees. Examples are checksum-based techniques [42,43], modulo arithmetic checking [83], shadow registers to detect timing violations [77], and lightweight redundancy-based detection techniques, such as dual modular redundancy (DMR), to identify faults by discrepancies in output between the replicated components. For example, neuron-level duplication strategies [29] and approximate DMR approaches [92] are methods to detect faults by measuring differences between redundant outputs, and can be used in a comparable way to provide runtime fault detection with moderate overhead. These methods provide deterministic fault detection with high coverage that can be as high as 100% on certain fault types, but add complexity to hardware and control.

A third category includes structural and architectural detection mechanisms, such as comparison between processing elements (PEs) [25,31,41], synchronization-based comparison, and watchdog or timer-based detection [39,88]. These approaches leverage architectural redundancy or system-level monitoring to detect faults with high reliability. In general, there is a basic trade-off between accuracy, overhead, and coverage among detection mechanisms. Methods that are aided by hardware offer more strong and deterministic guarantees, but statistical methods offer improved efficiency with lower implementation cost.

3.4.2. Recovery Strategies: From Masking to Dynamic Reconfiguration

Recovery mechanisms differ greatly in the literature, based on various assumptions about the persistence of faults and system requirements. The easiest is masking-based recovery, in which faulty outputs are masked or zero-masked (e.g., zero-masked, in [30,41,83]). These methods can allow the graceful degradation with minimal overhead but must not be fully correct. More intricate methods include recovery based on recomputation, in which a failed computation is re-run on spare units or on other datapaths, including [25,27,42,54]. These are more accurate methods of recovery, at a moderate performance time. A third method is architectural reconfiguration and bypass, e.g., rerouting datapaths to evade malfunctioning processing elements [27] or omitting malfunctioning ensemble elements [79]. These plans enable the system to keep on running.

An even longer paradigm is adaptive runtime mechanisms which can change the behavior of the system to different operating conditions dynamically [25]. In more recent work, there are proposals of adaptive reliability controls within CNN accelerators, and in this system, the system will dynamically switch to simplex (high-performance) and TMR (high-reliability) operation based on how sensitive the layers are to faults [93]. This allows trade-offs between performance and resilience under runtime conditions which are fine-grained. Indicatively, [45] proposes a multi-mode CNN accelerator, fault-tolerant (FT), high-performance (HP), and de-stress (DS), which enables trade-offs to be made in the reliability, performance, and energy efficiency during the runtime. Nonetheless, safety-guaranteed transitions and certification-oriented behavior are not explicitly defined in such approaches.

Continuing on these strategies, adaptive and self-healing processes can be considered to be the most progressive recovery paradigm. Checkpointing and rollback mechanisms can be used in systems like Auto-Healer [39], allowing recovery of both transient and multi-fault conditions. Similarly, system-level recovery of configuration memory faults is done by partial reconfiguration and scrubbing methods [88]. Beyond recovery effectiveness [94], recent studies highlight the critical impact of hardware faults on system-level behavior. As an example, [36] shows that faults in FPGA-based DNN accelerators may cause system exceptions like execution stalls, timeouts, and abnormal runtime behavior. Notably, such system-level failures tend to dominate the overall system reliability. System-level failures like stalls and abnormal execution can be caused by faults in non-computational modules, underscoring the need to ensure system-level resilience in addition to computation correctness [53]. This shows that proper runtime resilience should be implemented to tackle the correctness of computation and system stability. In general, these results indicate a strong transition of the static recovery strategies to the dynamic, context-aware, and system-aware mechanisms that can be used to handle complex and multi-fault situations.

3.4.3. Safety Behavior and Fault-Handling Modes

An important requirement in industrial safety-critical systems is that it is not only necessary to identify and recover from faults, but also to impose safe system behavior under fault conditions. Some studies directly use safety modes which are consistent with functional safety principles. For example, [78] shows a transition to an “absorbing safe state” (e.g., forced braking), and [82] shows how redundant SoC architectures can be used to implement a fail-operational behavior. Such methods are in accordance with functional safety standards, including ISO 26262, that demand systems to ensure safe operation even under fault conditions. Quantitative safety evidence is also available in other works, such as high fault coverage (>99%) [83], bounded accuracy degradation [28], and strong correlations among detection signals and system failures [47]. Although these metrics help in developing a safety case, they are usually limited to component-level validation rather than full system-level assurance. Nonetheless, many studies lack an explicit definition of

safety modes (e.g., fail-safe, fail-degraded, fail-operational) [95] and are not concerned with ensuring safe behavior of the system but instead with preserving the accuracy of the inferences. This represents a critical limitation, because preservation of accuracy, alone, is not enough to ensure safe system behavior or an adherence to industrial safety standards.

3.4.4. Integration Gap: Detection–Recovery–Safety Coupling

One of the key insights from Table 5 is that the majority of works concentrate on either detection or recovery, with relatively few studies providing a fully integrated pipeline that combines detection, fault isolation, recovery, and safety enforcement. For example, there are works that offer good detection and limited recovery (e.g., analysis frameworks [75,84]) and those with recovery but no explicit detection guarantees (e.g., retraining-based approaches [69]). A small group only combines the two components with coordinated runtime mechanisms (e.g., [27,39,88]). This fragmentation is indicative of one of the weaknesses of current research: a lack of end-to-end fault management platforms that are capable of providing a consistent, predictable, and safety-compliant system behavior when fault conditions occur.

Table 5. Mapping between detection mechanisms and recovery strategies in fault-tolerant DNN accelerators.

Detection Mechanisms/Recovery Strategies	Masking	Re-Compute	Reconfiguration	Checkpoint	System Level
Monitoring/SED	[44,47]	—	—	—	—
Checksum/ABFT	[43,83,96]	[42,54,56]	[27,32]	—	—
Hardware-based	[40,92,97,98]	[26]	—	—	—
Structural (PE/ traversal/ comparison)	[25,28,31,41,65,79]	[33]	[27]	—	—
Offline/analysis-based	[35,36,69,89]	—	—	—	—
System-level monitoring	—	—	[88]	[39]	[81,82,85,99,100]
Statistical/ reliability-driven	[64,90]	—	—	—	—
Hybrid/adaptive detection	[72,74]	[54]	[45,94]	—	—

3.4.5. Industrial Readiness and Certification Perspective

From an industrial perspective, the majority of detection and recovery systems are still at TRL 3–4 with minimal testing in real-world deployment scenarios. Although there are studies with high detection accuracy and successful recovery in controlled settings, few of those have the required certification-related evidence, including deterministic timing guarantees (WCET), system-wide safety validation, and compliance with safety standards, including ISO 26262 and IEC 61508. Some of the notable exceptions include those which demonstrate an indication of compliance with ASIL-D [83], real-world implementation of SoCs in automotive systems [82], and fault recovery systems that have been hardware-validated [88]. However, most approaches do not provide the detailed safety case needed to be certified, and there is a large gap between the academic literature and real-world implementation.

3.4.6. Conclusion of RQ4

The discussion of the runtime fault tolerance schemes shows that current DNN accelerators are moving away from the limited detection or recovery schemes to more integrated, adaptive, and safety-aware fault management systems. Although considerable advances

have been made in detection accuracy and recovery efficiency, most methods are still limited in their ability to guarantee safe system behavior under all fault conditions. Critically, the absence of tightly coupled detection–recovery–safety systems is an essential obstacle to the industrial adoption. Future studies are required to create certification-aware, deterministic, and system-level-validated fault management architectures that can be used to assure fail-safe, fail-degraded, or fail-operating behavior in safety-critical autonomous systems.

In general, the analysis confirms that the majority of available solutions prioritize localized detection and recovery mechanisms, whereas a detailed system-level fault management is a poorly explored area. This points to a severe discontinuity in the delivery of integrated detection–recovery–safety pipelines needed to deploy safety-critical systems. Table 5 demonstrates that the detection techniques may be divided into monitoring-based, checksum-based, hardware-assisted, structural, and system-level, and are aligned to the recovery strategies. A representative list of studies that reveal the co-occurrence of detection and recovery mechanism is provided in each cell. The mapping shows that there is a high level of concentration of works on masking and re-computation, with the more advanced recovery strategies like checkpointing and system-level recovery being relatively few. This implies a lack of fully integrated fault management pipelines required for safety-critical deployment.

3.5. RQ5: Safety—Critical Requirements and Standards Compliance

RQ5 aims to assess how fault tolerant DNN accelerator architectures meet the fundamental constraints of industrial safety-critical systems, namely, the notion of timing determinism, fault containment, safety enforcement, and standards awareness. All reviewed studies were compared in terms of safety behavior, architectural determinism, fault isolation capability, and the degree of compliance with industrial safety requirements to assess their practical applicability for a reliable deployment. The safety-critical requirement analysis of fault-tolerant DNN accelerators indicates that architectural resilience is not enough to ensure dependable operation in industrial safety-critical conditions. Fault-aware design involves detecting, mitigating, and tolerating hardware faults during the execution of the DNNs to enhance system reliability. Unlike fault tolerance, certification-aware design includes further industrial safety considerations such as deterministic behavior, traceable validation, fault containment, and adherence to standards like IEC 61508 and ISO 26262. The term certification is used to describe the formal testing and evaluation procedure that is conducted to determine whether a system meets specific safety and reliability criteria suitable for use in industry. Functional safety is described as the lack of unreasonable risk due to malfunctioning behaviors of electrical and electronic systems, which have been defined in standards like ISO 26262 which form a structured safety lifecycle [101], which includes hazard analysis, risk assessment, and safety goal derivation based on severity, exposure, and controllability factors [102].

Safety-critical systems place strict requirements on determinism, reliability, and fault containment, and such deployments need limited failure rates and verifiable operational guarantees to achieve safe system behavior [103,104]. Experimental investigations have indicated that the FIT rate of DNN accelerators can be orders of magnitude above safety levels (e.g., ISO 26262) without fault-mitigation mechanisms [44,60,105]. Further evidence suggests that approximation methods, including the decrease in data precision, may also be used to decrease the inherent resilience of DNNs, making them more vulnerable to radiation-induced faults and a higher number of false outputs in safety-critical applications [61].

Table 6 provides a summary of important dimensions of timing determinism, fault containment, safety enforcement, standards alignment, and industrial readiness among representative fault-tolerant DNN accelerator designs.

Table 6. Condensed analysis of safety-critical compliance in fault-tolerant DNN accelerators.

Dimension	Key Evidence	Representative Studies	Industrial Insight
Timing Determinism	Cycle-accurate execution, bounded latency, predictable overhead	[26,31,39,42,83]	Most works achieve deterministic execution; however, formal WCET verification is generally lacking
Fault Containment	PE-level isolation, layer-wise protection, architectural partitioning (e.g., FFI, Pblocks)	[25,27,30,40,41,88]	Fault containment is architecturally implemented, but rarely formally validated for certification purposes
Safety Enforcement	Masking, re-computation, self-healing, fail-operational behavior	[39,65,78,81,82]	Only a limited number of works explicitly implement well-defined safety modes (e.g., fail-safe, fail-operational)
Standards Alignment	ISO 26262, IEC 61508, DO-254 references, ASIL-D targets	[31,33,42,44,47]	Most studies reference safety standards, but lack full compliance or certification-oriented implementation
Industrial Readiness	TRL 3–4 (majority), limited TRL 7–9 implementations	[81,82]	A significant gap remains between research prototypes and industrial deployment

As reflected in Table 6, while most studies demonstrate deterministic execution and architectural fault containment, only a limited subset explicitly enforce safety-critical operational modes or provide certification-aligned validation, revealing a critical gap toward industrial deployment. Based on the reviewed studies, there are three fundamental dimensions that consistently become crucial enablers of safety-critical DNN accelerator design: (i) timing determinism, (ii) fault containment, and (iii) safety enforcement, with another level of standard's alignment and certification preparedness.

3.5.1. Timing Determinism: From Predictable Execution to Verifiable Real-Time Guarantees

Deterministic execution as a precondition to safety-critical deployment is a trend that recurs in the literature. The majority of architectures provide either cycle-accurate or bounded latency behavior, required by real-time systems like autonomous driving and industrial control. Representative works, like [24,25,31], have fixed execution cycles independent of the fault conditions, and predictable runtime behavior. Similarly, lightweight detectors, such as modulo arithmetic [83] and checksum-based verification [42], have limited and deterministic overhead with limited impact on throughput. More sophisticated strategies involve timing-based recovery strategies. One such system is Auto-Healer [39], which achieves repair latency of the order of nanoseconds, thus enabling fault recovery in real time without affecting system deadlines. Dynamic scaling strategies [70,88] are further used in adjusting the system frequency with predictable limits of execution.

However, despite the advances, the majority of the research works implicitly use the assumption of determinism and are not delivered in a formal manner in terms of the worst-case execution time (WCET). This is a major limitation to certification, because standards like ISO 26262 mandate formally verifiable timing guarantees across all operating conditions.

3.5.2. Fault Containment: To Verifiable Isolation and Freedom from Interference

Fault containment is a key requirement that needs to ensure that local hardware faults do not lead to system-level failures. Structural, system, and architectural processes combined bring about literary containment. The fault propagation inherently is localized to processing elements (PEs) levels to systolic and output-stationary architectures, as demonstrated in [25,30,31]. To minimize the extent of error propagation across network

levels, layer-wise containment, such as checksum verification [42], PE-level masking [41], and logic replacement [27] are employed.

System-level methods of partitioning, like FPGA Pblocks [40] and AXIS decoupling [88], impose spatial and functional isolation and are also equivalent to the ISO 26262 Freedom From Interference (FFI) requirement. Diversity-based redundancy approaches [65] deal with correlated failures on design, and reduce the effects of Common Cause Failures (CCFs), which are left mostly unaddressed in traditional redundancy schemes. Regardless of these mechanisms, fault containment is seldom formalized and validated in a certification framework. As a result, it is often treated as an architectural feature rather than a formally guaranteed safety property.

3.5.3. Safety Enforcement: From Accuracy Preservation to Guaranteed Safe System Behavior

One of the main differences between research-oriented resilience and industrial safety is that there are well-defined safety modes established during fault conditions. Although most studies strive to maintain inference accuracy, it is important that safety-critical systems guarantee safe behavior explicitly, such as fail-safe, fail-degraded, or fail-operational modes. Several works directly incorporate safety enforcement measures. An example is [78], which follows a lifecycle-driven safety framework, and automotive-grade designs like [81,82] that implement a fail-operational behavior with redundant system designs. Other methods are based on high fault coverage and early fault detection techniques like checksum-based fault detection [42,83] and activation monitoring fault detection [44] so that faults do not affect system outputs. Nevertheless, a significant portion of the literature does not explicitly define safety modes or system-level safety policies. Fault tolerance, in most instances, is only tested based on the accuracy degradation or detection coverage, but does not show safe system transitions with failure conditions. This is a fundamental gap because the adherence to the safety standards demands not only the detection and mitigation of faults, but also the ensured safe system behavior under any operating conditions.

3.5.4. Standards Alignment and Certification Readiness

The reviewed studies demonstrate to be partially compliant with major safety standards, such as ISO 26262, IEC 61508, and DO-254. These standards are referenced in many works which adopt safety concepts like ASIL classification, fault coverage metrics, and structured safety arguments. For example, a number of architectures specifically aim at complying with the ASIL-D standard [31,33,42,44,83], and system-level implementations [81,82] have been shown to satisfy automotive-grade safety requirements. Nevertheless, the majority of studies are at moderate levels of maturity (TRL 3–4), with validation mostly restricted to controlled experiments environments. A small fraction of them show deployment-grade validation with increased TRLs. A key limitation is that there are no end-to-end safety cases that combine architectural design, fault modeling, verification, and certification evidence. In the absence of such an integration, it is difficult to show compliance in real-world systems.

3.5.5. Key Gap: Absence of Certification-Aware Design Integration

One of the central findings of this analysis is that the existing fault-tolerant designs of DNN accelerators are largely safety-inspired rather than certification-driven designs. The individual safety factors, like deterministic execution, fault containment, and error detection, are being widely researched, but they are rarely integrated into a unified, certification-aware design approach. In particular, timing determinism is rarely proven, fault containment is not systematically validated, safety behavior is usually implicit instead of

being enforced, and allusions to safety standards are typically not formalized in the design process. Such fragmentation is a major hitch to industrial deployment.

3.5.6. Conclusion of RQ5

In general, the discussion suggests that fault tolerance is not the only aspect which can be taken into consideration to provide safety of DNN accelerators. Industrial preparedness is a combination of deterministic execution, provable fault containment, and safety behavior of enforceability and a certification-aware paradigm. Future research needs to move beyond single resilience solutions to system-level, certification-driven systems that can survive multi-fault scenarios and provide testable safety guarantees.

3.6. RQ6: Verification, Validation, and Reproducibility

The research of RQ6 aims at analyzing the methods of verification and validation, and the degree of reproducibility of fault-tolerant DNN accelerator research in industrial safety-critical systems. The reviewed studies were comparatively analyzed with respect to fault injection methodology, evaluated models, realism with respect to the datasets, hardware-level validation, support for reproducibility, and industrial applicability, to evaluate the reliability and credibility of the existing evaluation practices. The rigor of verification procedures, the naturalness of fault injection (FI) testing, and experimental reproducibility naturally drive fault-tolerant DNN accelerator evaluation [85]. In contrast to traditional performance-driven studies, safety-critical systems must have validation pipelines that can be used to show reliability in a diverse and realistic fault conditions. Validation techniques are closely intertwined with fault models (RQ2) and runtime resilience mechanisms (RQ4) because the quality of fault detection and recovery is directly proportional to the realism, coverage, and granularity of fault injection methodologies.

To provide a structured synthesis of validation methodologies and reproducibility practices, Table 7 presents a condensed comparison across key dimensions, including fault injection strategies, validation platforms, dataset realism, tested models, reproducibility, and industrial readiness.

Table 7. Condensed analysis of validation and reproducibility in fault-tolerant DNN accelerators.

Dimension	Key Evidence	Representative Studies	Industrial Insight
Fault Injection (FI) Methodology	Software FI (PyTorchFI, TensorFI), RTL-level FI, hardware-based injection, radiation testing (neutron beam experiments)	[40,41,47,56,60,75,83]	FI approaches are diverse but lack standardization; most studies assume single-fault models and limited realism
Validation Platform	FPGA-based validation (Zynq, ZCU102, Cyclone V), limited ASIC/silicon validation, GPU/CPU simulation	[27,41,77,79,81,82]	FPGA dominates validation; true silicon-level evidence remains limited
Dataset Realism	Benchmark datasets (MNIST, CIFAR-10, ImageNet), limited industrial datasets (MLPerf-Tiny, driving data)	[33,47,68,71,90]	Evaluation is dominated by benchmarks; real-world industrial workloads are underrepresented
Tested Models	Predominantly CNN-based architectures (ResNet, VGG, AlexNet, MobileNet, SqueezeNet), with limited inclusion of detection models (YOLO) and emerging architectures (Transformers)	[33,41,43,60,65,75,83]	Evaluation is architecturally diverse but lacks representation of real-world safety-critical pipelines (e.g., perception-control systems); limited coverage of modern and system-level models

Reproducibility	Open-source frameworks (PyTorchFI, FINN, NVDLA), detailed experimental setups, some proprietary limitations	[28,47,73,75,89]	Reproducibility is generally strong, but industrial studies often lack transparency
Industrial Readiness	TRL 3–4 (majority), limited TRL 5–9 validation (FPGA prototypes, silicon systems)	[56,81,82,98]	Significant gap between experimental validation and certification-ready deployment

As reflected in Table 7, the absence of standardized, certification-oriented and system-level evaluation strategies, as well as the lack of realistic industrial scenarios, represents a major obstacle to certification-oriented deployment, particularly with respect to fault injection campaigns and hardware validation efforts.

According to the analyzed studies summarized in Table 7, it is possible to interpret the literature systematically in four major dimensions: (i) fault injection methodology, (ii) validation platform, (iii) dataset realism, and (iv) reproducibility and experimental transparency.

3.6.1. Fault Injection Methodologies: From Software Emulation to Hardware-Level Realism

The most prevalent validation method is fault injection (FI), with which a wide spectrum of methodologies has been applied to emulate hardware induced faults. At the software level, frameworks such as PyTorchFI and TensorFI are increasingly commonly used to conduct large-scale bit-flip injection campaigns, as shown in papers like [43,47,83,86]. These methods allow fast searching fault sensitivity at network layers and fault rates, and can use thousands to millions of injection experiments. Repeated experiments performed with statistical fault injection allow a reliable confidence level in resilience assessment [52]. Statistically robust evaluation is supported by large-scale campaigns (e.g., 1.5M injections in SiFI-AI [62]) while maintaining practical simulation time with hybrid techniques, allowing accurate assessment of architectural vulnerability and radiation sensitivity [57].

To overcome the shortcomings of pure software-level FI, more recent cross-layer validation frameworks bring hardware-level fault characteristics to high-level simulation. As an example, the CLASSES framework is about 99% accurate as RTL-based fault injection while significantly reducing execution time (44×–63× speedup), allowing reliability testing on a large scale that is otherwise impractical with conventional approaches [105]. Nonetheless, software-level FI does not capture low-level hardware behavior, such as timing violations, spatial correlations, and microarchitectural behavior. To overcome this shortcoming, some of the works use RTL-level and cycle-accurate simulation such as [26,42,75,91] that model datapath-level faults and control logic interactions more accurately. Notably, RTL-based models like SiFI-AI [62] can provide cycle-accurate resilience analysis by integrating hardware simulation with DNN inference, allowing precise fault targeting in both data and control paths.

More realistic validation is achieved through hardware-based fault injection. ICAP-based hardware-level fault injection and partial reconfiguration hardware-level fault injection facilitates realistic testing of configuration memory errors in FPGA and SoC hardware, although it incurs significant experimental complexity [53]. To enhance this efficiency further, the recent FPGA-based fault injection models use a targeted injection strategy to limit fault insertion to configuration bits that are related to the circuit under test (CUT). These approaches significantly reduce the number of injected faults and accelerate fault injection campaigns while maintaining high fidelity of hardware fault emulation by taking advantage of ICAP and dynamic partial reconfiguration at the LUT level [67]. Neutron radiation experiments can also be used to perform high-fidelity validation, as it has realistic fault injection at a physical level. Neutron beam testing allows proper estimation of failure rates in real-world conditions, both in transient and permanent faults, and their

effect on a system level [56]. More complex memory faults, including single-bit upsets, stuck-at faults, and block-level errors on adjacent memory blocks have also been shown to significantly affect the accuracy of DNN inference and system stability [29,61,89]. Examples can be classified representative ones, like [27,41,73,79,88], where the manipulation of bitstreams, ICAP based injection, or corruption of configuration memory is used to simulate real hardware faults.

Radiation-based fault injection (e.g., neutron beam experiments) [40,56] is the most dependable assessment method, in which the fault behavior is confirmed by ground truth with realistic conditions. Latest hybrid assessment systems are also based on integration of fault injection testing and neutron radiation testing of CNN accelerators to test architectural vulnerability in real-life conditions. The Architectural Vulnerability Factor (AVF) and Mean Work To Failure (MWTF) are some of the measures used in such studies to obtain a quantification of resilience [88,94]. This advancement points out the apparent trade-off between scalability and realism: software FI is far more covered; hardware and radiation-based FI are of far higher fidelity but at a much higher cost of experiment. In the majority of fault injection attempts, single-fault cases are targeted, and little consideration has been taken for correlated or multi-fault behaviors, which are important in safety-critical systems.

3.6.2. Validation Platforms: FPGA Dominance and Sparse Silicon-Level Evidence

One of the trends that prevails in the literature is the widespread information on FPGA-based platforms in validation. Table 7 shows that an incredible proportion of research uses Xilinx and Intel FPGA platforms (e.g., Zynq-7000, ZCU102, Cyclone V) to enable controlled fault injection and architectural observability. They are a realistic and flexible combination that contains the RTL-level experimentation and hardware execution. Representative FPGA-based validation studies include [27,41,73,77,79,88]. On the other hand, ASIC-level verification is not so extensive. A majority of works that have been ASIC-oriented are either implemented with post-synthesis estimation or cycle-accurate simulation, such as in [26,32,60]. There is very limited literature of full validation to silicon, such as automotive-grade systems [82] and [81], running at TRL 8-9 and a veritable deployment in the industry. Trying to use GPU- and CPU-based platforms is also common, but mainly in large-scale statistical analysis and fault sensitivity experiments, as in [43,47,86,87]. However, such platforms are less effective in assessing hardware-level fault behavior of safety-critical embedded systems.

3.6.3. Dataset Realism: Benchmark Dominance vs. Industrial Workloads

The majority of studies use standard benchmark datasets including MNIST, CIFAR-10, CIFAR-100, and ImageNet, which are demonstrated in [26,42,47,60,83]. These datasets are consistent and comparable across studies, but they are not comprehensive enough to reflect the complexity of real industrial environment. A more limited collection of works use domain-specific or industrial datasets, such as MLPerf-Tiny data benchmarks [68], automotive driving data [47,65], multi-camera fisheye datasets (Woodscape) [90], and remote sensing datasets [71]. Nevertheless, in practice, real-world deployment situations are underrepresented. Fault tolerance is assessed in many studies on the basis of static datasets without paying attention to time behavior, environmental variability, and the interaction of a closed-loop system. This drawback is very serious in the sense of credibility of safety claims in that safety-critical systems should be able to work reliably in dynamic and uncertain conditions. In addition, the majority of the evaluations are operated in open-loop conditions, ignoring a closed-loop system behavior, which is crucial in safety-critical applications like autonomous driving and industrial controls.

3.6.4. Tested Models: Architectural Diversity vs. Industrial Representativeness

The type of tested models is a very important aspect that determines the generalizability and industrial relevance of fault-tolerance evaluation. As Table 7 demonstrates, the majority of the studies are based on well-known CNN architectures, such as ResNet, VGG, AlexNet, and lightweight models, such as MobileNet and SqueezeNet [41,43,47,60,83,89]. In spite of this variety, assessment is still mostly limited to classification-oriented CNNs, with little attention to more recent designs like transformers and perception models (e.g., YOLO) [60,65,75]. The current methods of validation are therefore not representative enough of the real-world industrial systems, particularly in closed-loop perception and control conditions. This reveals a primary weakness: architecturally different tested models provide a limited ability to model in their simulation workloads of industrial safety-critical workloads, which limits the external validity of reported fault-tolerance results.

3.6.5. Reproducibility and Experimental Transparency

One of the apparent trends in the literature is the wide use of reproducible research practices. There exists much research presenting open-source architectures, descriptions of experiments, and available public datasets. Representative examples include PyTorchFI- and TensorFI-based frameworks [47,83,86], FINN- and NVDLA-based toolflows [28,42], and open-source simulators and toolchains [73,75,89]. Such contributions enhance reproducibility significantly and can rightfully make comparisons between methods. Industry-level experiments, and more specifically ones involving proprietary data or equipment (e.g., [81,82]), are partly transparent, and therefore it is hard to verify them out of context of the experiment.

3.6.6. Industrial Readiness and Validation Gap

From the safety-critical viewpoint, most of the validation efforts are at moderate levels of maturity (TRL 3–4), which is evident in Table 7. Although there are a high number of studies with good experimental results under controlled conditions, they have little end-to-end system validation, real-time closed-loop evaluation, or certification-aligned verification (e.g., ISO 26262 safety cases). A small set of works reaches higher levels of readiness (TRL 5–9), usually through hardware prototyping, silicon validation, or industrialization.

3.6.7. Key Gap: Lack of Standardized, Certification-Oriented Validation Frameworks

Among the most important conclusions of this discussion, it is necessary to mention the fact that the current methodologies of validation are not standardized, and to a large extent, they are fragmented. In particular, fault injection methods differ greatly, metrics of evaluation are inconsistent and tend to be incomparable, datasets are not chosen in an industrial-realistic way, and certification-oriented validation is seldom discussed. This disaggregation is one of the key obstacles to industrial adoption, since safety-critical systems need reproducible and standardized and certifiable validation pipelines.

3.6.8. Conclusion of RQ6

In general, the analysis shows that, although significant progress has been made to the concept of fault injection and hardware validation, the discipline of fault injection does not have a single, standardized, and certification-aware methodology of validation. Further studies should be done to come up with standardized fault injection models, real-world industrial data, and end-to-end validation pipelines which combine hardware-, system-, and safety-level evaluation. Such efforts are essential to bridge the gap between experimental validation and certification-ready deployment of fault-tolerant DNN accelerators.

3.7. RQ7: Certification, Standards Alignment, and Industrial Readiness

In RQ7, the architecture of fault-tolerant DNN accelerators is assessed with respect to its readiness for certification, standards compliance, and its applicability in safety-critical systems. The standards awareness, certification considerations, safety integrity relevance, deployment maturity, and industrial applicability were comparatively analyzed for the reviewed studies in order to evaluate the practical maturity of the existing resilience architectures for real-world industrial deployment. The adoption of fault-tolerant DNN accelerators in safety-critical industrial systems is fundamentally grounded in their adherence to the set safety standards, the presence of evidence that is certification-oriented, and their level of maturity in terms of deployment readiness. Unlike conventional performance-driven systems, safety-critical platforms must satisfy stringent certification requirements, including fault coverage guarantees, deterministic behavior, and traceable validation processes. A certified DNN accelerator architecture does not just happen by using fault-tolerant mechanisms or safety standards. The application domain, desired safety integrity level (ASIL or SIL), system-level validation, deterministic behavior, traceable verification evidence, and conformance with domain-specific certification processes are also among the factors that determine certification. Thus, the readiness to get certified needs to be evaluated within the operational and regulatory context of the target industrial application. Notably, verification and validation processes (RQ6) are necessarily required to guarantee certification readiness since certification evidence should be obtained through systematic, traceable, and reproducible pipelines of evaluation.

To increase transparency and reproducibility, a structured evidence-based framework, derived from the key requirements of functional safety standards such as ISO 26262, IEC 61508, DO-254 and industrial certification practice, was used in conducting the certification-readiness assessment summarized in Table 8. The assessment considered four complementary dimensions: (i) standards alignment, (ii) safety integrity targeting (e.g., ASIL/SIL objectives), (iii) certification evidence, and (iv) industrial readiness. Explicit standards mapping, FIT-rate analysis, metrics for fault detection and recovery (e.g., SPFM), Architectural Vulnerability Factor (AVF) modeling, deterministic recovery behaviour, and reported Technology Readiness Levels (TRLs) were all used as evidence of readiness for certification. Studies with explicit standards alignment and evidence of certification through the standards, and studies with higher levels of deployment maturity were regarded as higher in certification readiness; studies with no evidence, safety-related metrics, or standards were regarded as having lower readiness.

Table 8. Condensed analysis of certification and standards alignment in fault-tolerant DNN accelerators.

Dimension	Key Evidence	Representative Studies	Industrial Insight
Standards Mapping	Explicit mapping to ISO 26262, IEC 61508, DO-254, NASA RHA; partial or implicit alignment in some works	[31,42,47,60,78,83,88,99,100]	Strong trend toward standards awareness, but many studies lack formal compliance integration
SIL/ASIL Targeting	ASIL-D and SIL 3/4 targets using FIT, SPFM, AVF metrics; some works lack explicit safety level assignment	[26,31,39,42,44,68,78,83,98]	High-integrity targets are common, but inconsistently defined and not systematically validated
Certification Gaps	Overhead of TMR, lack of hardware-aware validation, mismatch between AI accuracy and safety metrics, limited adaptability	[13,25,30,39,65,74,75,78]	Research addresses isolated certification challenges rather than complete certification workflows

Certification Evidence	FIT rate analysis, fault coverage (SPFM), AVF modeling, reliability metrics; limited safety-case traceability	[31,42–44,72,84]	Evidence exists but is fragmented and insufficient for full certification arguments
Industrial Readiness	TRL 3–4 (majority), limited TRL 7–9 systems (automotive SoCs, certified platforms)	[56,81,82,85,100]	Significant gap between academic prototypes and certified industrial deployment

In accordance with Table 8, despite the high rates of compliance with the safety standards and the high level of the target integration, the absence of the integrated, end-to-end certification workflows and full evidence of the safety case is the critical obstacle to the implementation in the industries. The literature demonstrates a heterogeneous landscape of standards alignment, ranging from explicit certification-driven designs to loosely aligned or exploratory models. These four dimensions can be arranged to form an analysis: (i) standards mapping, (ii) safety integrity level (SIL/ASIL) targeting, (iii) certification evidence and gaps, and (iv) industrial readiness.

3.7.1. Standards Mapping: From Explicit Compliance to Implicit Alignment

A considerable part of the literature shows a clear compliance with significant safety standards, especially ISO 26262, regarding the automotive systems [106]. Direct correlations between their reliability metrics (e.g., FIT rates, SPFM) to ISO 26262 requirements, like [27,31,42,44,47,60,83] represent a definite trend towards certification-aware design. In other applications, this alignment has been expanded to larger safety standards, such as IEC 61508 and DO-254, as seen in [78,99], which also address both industrial and avionics safety domains. Moreover, the works in space and aerospace utilize NASA Radiation Hardness Assurance (RHA) frameworks and related standards [88,100]. Nevertheless, a significant part of the literature is still implicitly in compliance with safety standards. These approaches are aimed at safety-critical applications but do not formalize compliance requirements, as in [26,28,36,68]. Although such studies have a meaningful contribution to architectural study, they have deficiencies in terms of formal standards mapping, restricting their applicability in certified industrial environments.

3.7.2. Safety Integrity Levels (SIL/ASIL): Toward High-Integrity Targets

Numerous works directly aim at the high-integrity safety levels, especially ASIL-D in the automotive systems, the most demanding safety level. Research like [31,39,42,44,83] has shown compliance with requirements of ASIL-D, frequently by quantitatively measuring rates of FIT (e.g., 10 FIT or less) and fault coverage indicators. In the same way, SIL 3/4 targets are dealt with in an industrial context, as seen in [78], whereby, system-level reliability is mapped to hardware-level fault tolerance systems. Nevertheless, in a number of studies, SIL/ASIL assignment is not explicit, even when safety-critical domains are considered. Examples: [26,68,98], where the enhancement of reliability is proven but not directly related to certification levels. This gap highlights a disconnect between architectural innovation and certification-oriented evaluation.

3.7.3. Certification Gaps: From Architectural Innovation to Certifiable Evidence

One of the key findings of Table 8 is that the majority of studies do not offer full-scale certification-ready solutions, but focus on certain gaps in certification. Examples of common gaps are the high hardware cost of conventional redundancy schemes (such as TMR), mitigated by lightweight schemes [30,65,83]; the absence of hardware-awareness in certification procedures, mitigated by RTL-level tools [75]; the mismatch between accuracy measures of AI and safety requirements, mitigated in [74,78]; and the limited adaptability of static redundancy schemes, addressed by adaptive architectures [25,39]. While these

contributions are valuable, they are typically isolated and do not form a complete certification pipeline. Most importantly, the end-to-end safety cases, traceability between fault models and safety requirements, and formal verification in accordance to certification workflows are not given in most studies. This limitation indicates that the field is still in a transitional phase, where certification is considered but not fully integrated into the design process.

3.7.4. Industrial Readiness: From Research Prototypes to Certified Systems

Most of the studies are still at moderate levels of maturity (TRL 3-4), indicating validation in controlled experimental settings and not in real-world application. Nonetheless, there are a limited number of works that show high industrial readiness. Notably, automotive-grade systems such as [81,82] are certified or certification-ready, meeting AEC-Q100 and ISO 26262 requirements; industry-developed tools and platforms, including those in [85], have been shown to be deployed in real-world autonomous systems; and high-reliability frameworks of aerospace and mission-critical systems [56,100] provide direct applicability to certification processes. These publications mark the shift of research prototypes into systems that can be deployed, but such are very few, highlighting a critical gap in the field.

3.7.5. Key Gap: Absence of End-to-End Certification-Aware Design Methodologies

One of the key observations of the analysis is that current fault-tolerant DNN accelerator studies do not have end-to-end certification-aware design methodologies. In particular, there are a tendency to mention standards without operationalization; inconsistently definition of SIL/ASIL targets; fragmented certification evidence throughout the design, validation, and deployment stages; and little attention to multi-fault and Common Cause Failure (CCF) scenarios during certification. This disintegration does not allow complete safety cases to be formed that are needed for certification in industrial systems.

3.7.6. Conclusion of RQ7

Overall, the analysis shows that although significant progress has been made in aligning fault-tolerant DNN accelerators to safety standards, the area is mostly at the pre-certification maturity stage. Future research needs to change to a certification-driven design process and combine standards mapping, fault modeling, validation, and safety case generation into one unified methodology. This kind of integration is necessary in order to help close the disparity between architectural innovations and approved implementation in industrial safety-critical systems.

4. Cross-RQ Synthesis

The most important contribution of this review is that it is a detailed analysis of all the research questions, which reveals a strong interdependency between the architectural design, fault modeling, platform-level trade-offs, runtime resilience, validation methodologies, and certification readiness. Based on RQ1 and RQ2, it can be seen that the practicality of fault-tolerant architectures is inherently limited to the practicality of the fault models underlying them. The monoculture of single-fault assumptions is a grave drawback on the robustness of architectural solutions, particularly in multi-fault and common-cause failure (CCF) situations. The results of RQ3 and RQ4 additionally prove the fact that platform selection and runtime resilience mechanisms are highly interconnected. Though implementations with FPGA can be used to enable flexible fault injection and adaptive recovery, most runtime systems are localized and lack coordination across the system between detecting, recovering, and enforcing safety. Interestingly, both RQ6 and RQ7 make it clear that the problem of validation and certification readiness are two separate things

that are interrelated. The key limitations to generating certification-grade evidence are the fact that no standardized methodologies exist to approach fault injection, realistic datasets, and end-to-end validation pipelines. To offer a comprehensive and holistic perspective across all the research dimensions, Figure 6 shows how the key research questions (RQ1–RQ7) interact, with the relationships between architectural design, fault modeling, platform trade-offs, run-time resilience, safety requirements, validation methodologies, and certification readiness.

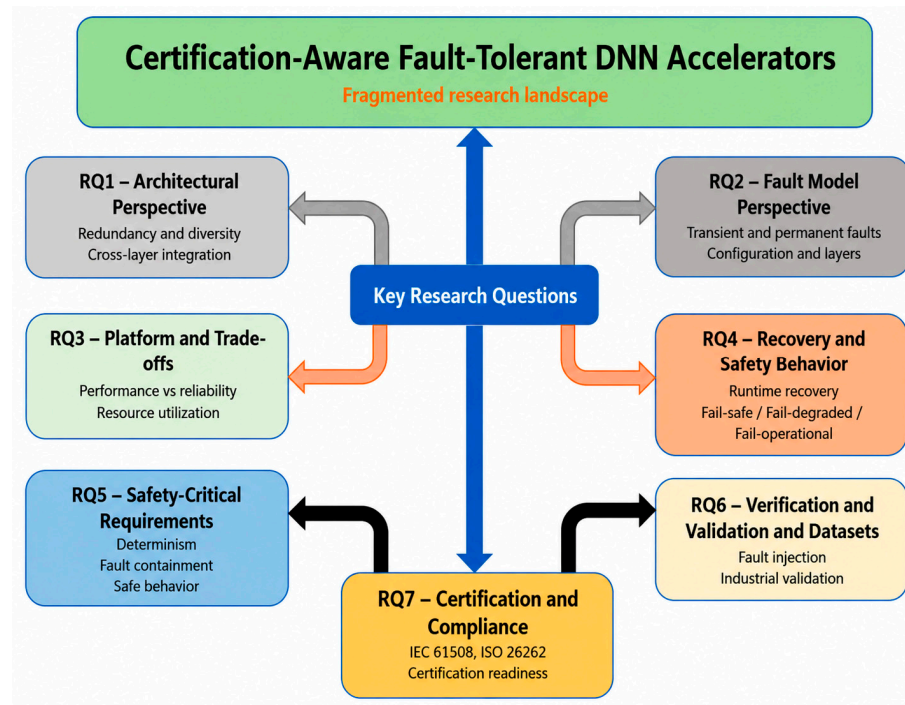


Figure 6. Cross-RQ interaction framework for fault-tolerant DNN accelerators toward certification-aware design.

In summary, the cross-analysis demonstrates that the existing studies are disjointed at the architectural, modeling, and validation levels without a unified, certification-aware design approach. To close this gap, a comprehensive framework that includes fault modeling, cross-layer resilience, runtime fault management, and certification-oriented validation together in a coherent design process is needed. It is worth noting that most of the reviewed studies are still at early levels of maturity (TRL 3–4) and primarily rely on single-fault assumptions; only a small number of studies discuss multi-fault or system-level validation scenarios.

Figure 6 indicates that the area of research is organized into several interdependent dimensions, and each research question is related to a different aspect of system reliability and safety. The intersection of these dimensions is the certification readiness (RQ7), which indicates that the enhancement of single layers is ineffective without integrating the whole, end-to-end. These dimensions are, however, usually considered separately, leading to disjointed solutions which do not have such integration—specifically, to enable certification readiness, alignment between architectural design, fault models, resilience of runtime, validation, and safety-related mechanisms within a unified framework.

4.1. Positioning of the Present Review Relative to Existing Surveys

DNN reliability, resilience, and safety have been studied in several review studies from different angles. Previous reliability-oriented surveys include [7] and [8], which focus mainly on soft errors, fault models, and resilience mechanisms in the DNN accelerators, while [6] discusses reliability problems related to aging. A few other studies are also focused on resilience assessment and fault-injection methods [23] and acceleration frameworks and performance optimization [76]. AI safety reviews [2,3,50] focus primarily on assurance processes and lifecycle management and governance aspects, rather than on fault tolerance at the hardware level. Likewise, reviews [18], [103], and [106] concentrate on the connected vehicle, space application, or FPGA safety platforms, respectively, but do not offer a comprehensive look at fault-tolerant DNN accelerator architectures.

However, the present review differs from the existing literature in several important aspects. First, this review is focused on industrial safety-critical systems, where hardware faults can have a direct impact on system safety and operational dependability, rather than on performance, latency, energy efficiency, soft errors, or aging-related reliability problems, which are subjects of many performance-oriented surveys. Second, the review not only addresses the notion of fault-tolerance, but also explicitly addresses the certification-readiness and the alignment of resilience mechanisms to industrial standards for safety, like ISO 26262 or IEC 61508. Third, the review incorporates a qualitative Technology Readiness Level (TRL) assessment, enabling a systematic evaluation of the maturity of existing solutions and highlighting the gap between laboratory research prototypes and industrially deployable systems. Fourth, the methodology used is based on PRISMA and the evidence is structured around seven research questions (RQ1 to RQ7) that are complementary to each other, which enables a cross-dimensional synthesis of the research area, which connects architectural design, fault modeling, run-time resilience, validation methodologies, safety requirements, and certification readiness. Last, the review adopts a functional view of fault management, looking at the correlation between fault detection, fault mitigation, and fault recovery mechanisms in the industrial DNN accelerator architectures. These attributes together make the present review different from the previous surveys, and gives a more complete picture of the challenges and opportunities for the deployment of fault-tolerant DNN accelerators in industrial safety-critical systems.

4.2. Limitations of the Review

Although this review is systematic and comprehensive, there are a number of limitations that should be noted. First, publicly available studies are mainly used in the analysis, and this may result in publication bias because proprietary industrial implementations may not be as widespread as they are visible. Second, the majority of the evaluated works rely on benchmark datasets and controlled experimental environments, which do not fully capture the complexity and variability of real-world industrial deployments. In addition, several studies were identified as potentially relevant in the initial search but were excluded after full-text screening for a number of reasons, such as being software- or algorithmic robustness-only, not using fault-tolerant mechanisms, non-industrial contexts, no hardware-level consideration, or insufficient experimental validation [107–110]. Third, even though the review explicitly focuses on safety-critical systems, some studies were excluded because they indicated potential industrial applicability but did not provide industrial validation. Fourth, there is no standardization of fault injection methodologies, fault models, and evaluation metrics, which restricts the comparability and reproducibility of the results across studies. Lastly, current research is predominantly concerned with isolated fault scenarios and component-level validation, and does not pay much attention to system-level interrelations, multi-faulting, and common-cause failures (CCFs), which are essential to safety certification. These constraints highlight the importance of stand-

ardized, system-level, and certification-oriented evaluation frameworks, and increased cooperation between academia and industry. Such constraints are the driving forces behind the research directions outlined in Section 4.2.

4.3. Future Research Directions

To overcome such limitations, there are a number of important future research directions. To enhance the realism and completeness of fault modeling, first, future work should focus on realistic and comprehensive fault modeling, such as multi-fault scenarios and common-cause failures, to provide a more realistic picture of the industrial operating conditions. Second, this would require the formation of standardized and certification-focused fault evaluation structures and benchmarking strategies so that they can guarantee reproducibility, comparability, and certification-level validation. Third, studies ought to shift to implementation-level design approaches that are cross-layer and certification-sensitive: that is, they meet the features of hardware resilience, runtime fault management, safety behavior enforcement, and formal verification in a unified framework. Fourth, the development of industrial-grade datasets and closed-loop evaluation environments is essential to validate system behavior under realistic and dynamic conditions. Lastly, adaptive and self-healing should be built into future architectures with safety modes (e.g., fail-safe, fail-degraded, fail-operational) spelled out, allowing predictable and dependable operation in safety-critical systems. Another research area is to directly use AI- and DNN-inspired techniques in designing fault-tolerant architecture, where reliability is not only provided by external protection mechanisms, but also by adaptive and intelligent architectural behavior. In addition, the DNN architecture should be further enhanced for fault tolerance through methods such as fault-aware training, resilient quantization, saliency-aware neural mapping, and neuron-level selective protection. These are new concepts, shifting from protection-centric approaches to intelligent and adaptive resilience in future industrial safety-critical systems.

4.4. Emerging Trends in Large-Scale AI Systems

With recent progress in AI, there is a shift from CNN-oriented accelerators to novel architectures for Transformers and LLMs [111]. These systems demand much more memory and computation requirements, which have led to the adoption of heterogeneous chiplet-based architectures and advanced 2.5D/3D integration (such as CoWoS, FOVEROS) [112,113]. While improving performance, they introduce concerns like inter-chip communication reliability and thermal-related reliability challenges [113]. At the same time, in-memory computing (IMC) and methods such as TFix leverage the inherent redundancy of ternary cells to overcome failures at minimal overhead [114]. Furthermore, large-scale serving infrastructures introduce new challenges in resource concurrency and runtime state consistency [115], indicating that future research must expand toward system-scale resilience and certification-aware requirements.

5. Conclusions

This paper presents a comprehensive systematic survey of fault-tolerant DNN accelerator architectures for industrial safety-critical autonomous systems, covering architectural design, fault modeling, platform-level trade-offs, runtime resilience, validation methodologies, and certification readiness. The analysis shows that there is a substantial gap between research-oriented fault tolerance solutions and the requirements of industrial deployment, especially because of a lack of certification alignment, system-level validation, and simplified fault models. These results also indicate that architectural design, fault modeling, platform level trade-offs, runtime resilience, validation methodologies, and certification readiness are tightly coupled dimensions, yet are frequently addressed

in isolation across the literature. One important observation is that the existing research is shifting the focus on redundancy-centric designs to cross-layer and adaptive resilience models; however, this evolution remains largely disconnected from realistic fault modeling and certification-driven design requirements. Specifically, the absence of standardized validation frameworks, multi-fault and common-cause fault models, and end-to-end certification workflows are a basic obstacle to deployment. To cope with these issues, it is necessary to focus on holistic, certification-aware design methodologies that combine cross-layer resilience, runtime fault management, platform-aware optimization, and safety assurance in a unified and verifiable framework. Finally, this paper demonstrates that using traditional fault tolerance techniques in the design of DNN accelerator architectures embedded in industrial safety-critical systems is not sufficient to ensure safety. Instead, DNN accelerator architectures must be explicitly developed for certification through using industrial standards for safety like ISO 26262 or IEC 61508, determinism, and dependable behavior to permit real-world deployment in industrial safety-critical applications. In this regard, an exciting future avenue is the creation of an integrated, certification-aware design architecture that integrates fault modeling, cross-layer resilience, runtime fault management, and validation into a common end-to-end methodology. This would give a formal basis on how to bridge the gap between research prototypes and certifiable industrial systems, eventually facilitating a safe, dependable, and certifiable implementation of AI into next-generation industrial safety-critical autonomous systems.

Supplementary Materials: The following supporting information can be downloaded at: <https://www.mdpi.com/article/10.3390/digital6030054/s1>, Table S1. PRISMA_2020_abstract_checklist. Table S2. PRISMA_2020_checklist.

Author Contributions: Conceptualization, F.N.Q. and S.S.K.; methodology, F.N.Q. and S.A.D.; formal analysis, F.N.Q., S.S.K. and S.A.D.; investigation, F.N.Q., S.S.K. and S.A.D.; data curation, F.N.Q.; writing—original draft preparation, F.N.Q.; writing—review and editing, S.S.K. and S.A.D.; visualization, F.N.Q.; supervision, S.S.K. and S.A.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No new datasets were generated during this study. All data analyzed in this systematic review are available in the published articles cited in the reference list.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Windmann, A.; Wittenberg, P.; Schieseck, M.; Niggemann, O. Artificial Intelligence in Industry 4.0: A Review of Integration Challenges for Industrial Systems. In Proceedings of the 2024 IEEE 22nd International Conference on Industrial Informatics (INDIN), Beijing, China, 18–20 August 2024; pp. 1–8. <https://doi.org/10.1109/INDIN58382.2024.10774364>.
2. Perez-Cerrolaza, J.; Abella, J.; Borg, M.; Donzella, C.; Cerquides, J.; Cazorla, F.J.; Englund, C.; Tauber, M.; Nikolakopoulos, G.; Flores, J.L. Artificial Intelligence for Safety-Critical Systems in Industrial and Transportation Domains: A Survey. *ACM Comput. Surv.* **2024**, *56*, 3626314. <https://doi.org/10.1145/3626314>.
3. Neto, A.V.S.; Camargo, J.B.; Almeida, J.R.; Cugnasca, P.S. Safety Assurance of Artificial Intelligence-Based Systems: A Systematic Literature Review on the State of the Art and Guidelines for Future Work. *IEEE Access* **2022**, *10*, 130733–130770. <https://doi.org/10.1109/ACCESS.2022.3229233>.
4. Al Jbaar, M.A.; Dawwd, S.A. DCNN-based embedded models for parallel diagnosis of ocular diseases. *East.-Eur. J. Enterp. Technol.* **2023**, *4*, 53–69. <https://doi.org/10.15587/1729-4061.2023.281790>.
5. Wan, Z.; Lele, A.; Yu, B.; Liu, S.; Wang, Y.; Reddi, V.J.; Hao, C.; Raychowdhury, A. Robotic Computing on FPGAs: Current Progress, Research Challenges, and Opportunities. In *Proceedings of the 2022 IEEE 4th International Conference on Artificial Intelli-*

- gence Circuits and Systems (AICAS), Incheon, Republic of Korea, 13–15 June 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 291–295. <https://doi.org/10.1109/AICAS54282.2022.9869951>.
6. Moghaddasi, I.; Gorgin, S.; Lee, J.-A. Dependable DNN Accelerator for Safety-Critical Systems: A Review on the Aging Perspective. *IEEE Access* **2023**, *11*, 89803–89834. <https://doi.org/10.1109/ACCESS.2023.3300376>.
7. Mittal, S. A survey on modeling and improving reliability of DNN algorithms and accelerators. *J. Syst. Archit.* **2020**, *104*, 101689. <https://doi.org/10.1016/j.sysarc.2019.101689>.
8. Ibrahim, Y.; Wang, H.; Liu, J.; Wei, J.; Chen, L.; Rech, P.; Adam, K.; Guo, G. Soft errors in DNN accelerators: A comprehensive review. *Microelectron. Reliab.* **2020**, *115*, 113969. <https://doi.org/10.1016/j.microrel.2020.113969>.
9. Hanif, M.A.; Shafique, M. Dependable Deep Learning: Towards Cost-Efficient Resilience of Deep Neural Network Accelerators against Soft Errors and Permanent Faults. In *Proceedings of the 2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS), Napoli, Italy, 13–16 July 2020*; IEEE: Piscataway, NJ, USA, 2020; pp. 1–4. <https://doi.org/10.1109/IOLTS50870.2020.9159734>.
10. Awad, S.R.; Khairullah, S.S. Soft errors detection and adaptive correction in real-time and dependable processor networks using quantized machine learning models. *J. Supercomput.* **2026**, *82*, 98. <https://doi.org/10.1007/s11227-026-08265-3>.
11. Al Mokhtar, Z.T.; Dawwd, S.A. Deep Learning Video Prediction Theories and Their Architecture: A Review. *J. Eng. Sustain. Dev.* **2025**, *29*, 771–784. <https://doi.org/10.31272/jeasd.2262>.
12. Al Mokhtar, Z.; Dawwd, S. 3D VAE Video Prediction Model with Kullback Leibler Loss Enhancement. *Int. Arab J. Inf. Technol.* **2024**, *21*, 879–888. <https://doi.org/10.34028/iajit/21/5/9>.
13. Ahmadilivani, M.H.; Mousavi, S.; Raik, J.; Daneshlab, M.; Jenihhin, M. Cost-Effective Fault Tolerance for CNNs Using Parameter Vulnerability Based Hardening and Pruning. *arXiv* **2024**, arXiv:2405.10658. <https://doi.org/10.48550/arXiv.2405.10658>.
14. Pedroza, G.; Morayo, A. Safe-by-Design Development Method for Artificial Intelligent Based Systems. In *Proceedings of the 31st International Conference on Software Engineering and Knowledge Engineering, Lisbon, Portugal, 10–12 July 2019*; pp. 391–397. <https://doi.org/10.18293/SEKE2019-094>.
15. Smith, C.; Denney, E.; Pai, G. Hazard Contribution Modes of Machine Learning Components. In *Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020*; pp. 1–9.
16. Kemsaram, N.; Das, A.; Dubbelman, G. An Integrated Framework for Autonomous Driving: Object Detection, Lane Detection, and Free Space Detection. In *Proceedings of the 2019 Third World Conference on Smart Trends in Systems Security and Sustainability (WorldS4), London, UK, 30–31 July 2019*; IEEE: Piscataway, NJ, USA, 2019; pp. 260–265. <https://doi.org/10.1109/WorldS4.2019.8904020>.
17. ISO 26262:2018; Road Vehicles—Functional Safety. International Organization for Standardization (ISO): Geneva, Switzerland, 2018.
18. IEC 61508:2010; Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems. International Electrotechnical Commission (IEC): Geneva, Switzerland, 2010.
19. RTCA DO-254; Design Assurance Guidance for Airborne Electronic Hardware. RTCA, Inc.: Washington, DC, USA, 2000.
20. Khairullah, S.S. Formal Verification of a Dependable State Machine-Based Hardware Architecture for Safety-Critical Cyber-Physical Systems: Analysis, Design, and Implementation. *J. Electron. Test.* **2024**, *40*, 509–523. <https://doi.org/10.1007/s10836-024-06126-6>.
21. Rodríguez-Arozamena, M.; Matute, J.; Araluce, J.; Rastelli, J.P.; Zubizarreta, A. Fault Tolerance and Fallback Strategies in Connected and Automated Vehicles: A Review. *IEEE Open J. Intell. Transp. Syst.* **2025**, *6*, 915–937. <https://doi.org/10.1109/OJITS.2025.3583787>.
22. Zhang, J.J.; Liu, K.; Khalid, F.; Hanif, M.A.; Rehman, S.; Theocharides, T.; Artussi, A.; Shafique, M.; Garg, S. Building Robust Machine Learning Systems: Current Progress, Research Challenges, and Opportunities. In *Proceedings of the 56th Annual Design Automation Conference 2019, Las Vegas, NV, USA, 2–6 June 2019*; ACM: New York, NY, USA, 2019; pp. 1–4. <https://doi.org/10.1145/3316781.3323472>.
23. Ruospo, A.; Sanchez, E.; Luza, L.M.; Dilillo, L.; Traiola, M.; Bosio, A. A Survey on Deep Learning Resilience Assessment Methodologies. *Computer* **2023**, *56*, 57–66. <https://doi.org/10.1109/MC.2022.3217841>.
24. Zhang, J.J.; Gu, T.; Basu, K.; Garg, S. Analyzing and mitigating the impact of permanent faults on a systolic array based neural network accelerator. In *Proceedings of the 2018 IEEE 36th VLSI Test Symposium (VTS), San Francisco, CA, USA, 22–25 April 2018*; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6. <https://doi.org/10.1109/VTS.2018.8368656>.
25. Cherezova, N.; Jutman, A.; Jenihhin, M. FORTALES: Fault-Tolerant Reconfigurable Systolic Array for DNN Inference. *Microprocess. Microsyst.* **2025**, *119*, 105222. <https://doi.org/10.1016/j.micpro.2025.105222>.

26. Zhao, Y.; Wang, K.; Louri, A. FSA: An Efficient Fault-tolerant Systolic Array-based DNN Accelerator Architecture. In *Proceedings of the 2022 IEEE 40th International Conference on Computer Design (ICCD), Olympic Valley, CA, USA, 23–26 October 2022*; IEEE: Piscataway, NJ, USA, 2022; pp. 545–552. <https://doi.org/10.1109/ICCD56317.2022.00086>.
27. Gao, Z.; Qi, Y.; Shi, J.; Liu, Q.; Ge, G.; Wang, Y.; Reviriego, P. Detect and Replace: Efficient Soft Error Protection of FPGA-Based CNN Accelerators. *IEEE Trans. Very Large Scale Integr. VLSI Syst.* **2025**, *33*, 66–74. <https://doi.org/10.1109/TVLSI.2024.3443834>.
28. Bertoa, T.G.; Gambardella, G.; Fraser, N.J.; Blott, M.; McAllister, J. Fault-Tolerant Neural Network Accelerators With Selective TMR. *IEEE Des. Test* **2023**, *40*, 67–74. <https://doi.org/10.1109/MDAT.2022.3174181>.
29. Li, Y.; Liu, Y.; Li, M.; Tian, Y.; Luo, B.; Xu, Q. D2NN: A fine-grained dual modular redundancy framework for deep neural networks. In *Proceedings of the 35th Annual Computer Security Applications Conference, San Juan, PR, USA, 9–13 December 2019*; ACM: New York, NY, USA, 2019; pp. 138–147. <https://doi.org/10.1145/3359789.3359831>.
30. Burel, S.; Evans, A.; Anghel, L. MOZART: Masking Outputs with Zeros for Architectural Robustness and Testing of DNN Accelerators. In *Proceedings of the 2021 IEEE 27th International Symposium on On-Line Testing and Robust System Design (IOLTS), Torino, Italy, 28–30 June 2021*; IEEE: Piscataway, NJ, USA, 2021; pp. 1–6. <https://doi.org/10.1109/IOLTS52814.2021.9486694>.
31. Burel, S.; Evans, A.; Anghel, L. MOZART+: Masking Outputs With Zeros for Improved Architectural Robustness and Testing of DNN Accelerators. *IEEE Trans. Device Mater. Reliab.* **2022**, *22*, 120–128. <https://doi.org/10.1109/TDMR.2022.3159089>.
32. Liu, C.; Chu, C.; Xu, D.; Wang, Y.; Wang, Q.; Li, H.; Li, X.; Cheng, K.T. HyCA: A Hybrid Computing Architecture for Fault-Tolerant Deep Learning. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2022**, *41*, 3400–3413. <https://doi.org/10.1109/TCAD.2021.3124763>.
33. Zhang, Q.; Liu, C.; Liu, B.; Huang, H.; Wang, Y.; Li, H.; Li, X. Cross-Layer Optimization for Fault-Tolerant Deep Learning Accelerators. *J. Comput. Res. Dev.* **2024**, *61*, 1370–1387. <https://doi.org/10.7544/issn1000-1239.202331005>.
34. Hanif, M.A.; Shafique, M. SalvageDNN: Salvaging deep neural network accelerators with permanent faults through saliency-driven fault-aware mapping. *Philos. Trans. R. Soc. Math. Phys. Eng. Sci.* **2020**, *378*, 20190164. <https://doi.org/10.1098/rsta.2019.0164>.
35. Malhotra, A.; Gupta, S.K. Fault Tolerant In-Memory Computing based on Emerging Technologies for Ultra-Low Precision Edge AI Accelerators. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design, Newark Liberty International Airport Marriott New York, NY, USA, 27–31 October 2024*; ACM: New York, NY, USA, 2024; pp. 1–9. <https://doi.org/10.1145/3676536.3697134>.
36. Malhotra, A.; Wang, C.; Gupta, S.K. BNN-Flip: Enhancing the Fault Tolerance and Security of Compute-in-Memory Enabled Binary Neural Network Accelerators. In *Proceedings of the 2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC), Incheon, Republic of Korea, 22–25 January 2024*; IEEE: Piscataway, NJ, USA, 2024; pp. 146–152. <https://doi.org/10.1109/ASP-DAC58780.2024.10473947>.
37. Khairullah, S.S. *Toward Biologically-Inspired Self-Healing, Resilient Architectures for Digital Instrumentation and Control Systems and Embedded Devices*; VCU Libraries: Richmond VA, USA, 2018. <https://doi.org/10.25772/7QKK-M119>.
38. Khairullah, S.S.; Elks, C.R. A Bio-Inspired, Self-Healing, Resilient Architecture for Digital Instrumentation and Control Systems and Embedded Devices. *Nucl. Technol.* **2018**, *202*, 141–152. <https://doi.org/10.1080/00295450.2018.1450014>.
39. Suvizi, A.; Venkataramani, G. Auto-Healer: Self-Healing Hardware for Perception Stage Faults in Autonomous Driving Systems. In *Proceedings of the 39th ACM International Conference on Supercomputing, Salt Lake City, UT, USA, 8–11 June 2025*; ACM: New York, NY, USA, 2025; pp. 1064–1078. <https://doi.org/10.1145/3721145.3725755>.
40. Libano, F.; Wilson, B.; Anderson, J.; Wirthlin, M.J.; Cazzaniga, C.; Frost, C.; Rech, P. Selective Hardening for Neural Networks in FPGAs. *IEEE Trans. Nucl. Sci.* **2019**, *66*, 216–222. <https://doi.org/10.1109/TNS.2018.2884460>.
41. Li, W.; Ge, G.; Guo, K.; Chen, X.; Wei, Q.; Gao, Z.; Wang, Y.; Yang, H. Soft Error Mitigation for Deep Convolution Neural Network on FPGA Accelerators. In *Proceedings of the 2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS), Genova, Italy, 31 August–2 September 2020*; IEEE: Piscataway, NJ, USA, 2020; pp. 1–5. <https://doi.org/10.1109/AI-CAS48895.2020.9073925>.
42. Sanic, M.T.; Guo, C.; Leng, J.; Guo, M.; Ma, W. Towards Reliable AI Applications via Algorithm-Based Fault Tolerance on NVDLA. In *Proceedings of the 2022 18th International Conference on Mobility, Sensing and Networking (MSN), Guangzhou, China, 14–16 December 2022*; IEEE: Piscataway, NJ, USA, 2022; pp. 736–743. <https://doi.org/10.1109/MSN57253.2022.00120>.
43. Zhao, K.; Di, S.; Li, S.; Liang, X.; Zhai, Y.; Chen, J.; Ouyang, K.; Cappello, F.; Chen, Z. FT-CNN: Algorithm-Based Fault Tolerance for Convolutional Neural Networks. *IEEE Trans. Parallel Distrib. Syst.* **2021**, *32*, 1677–1689. <https://doi.org/10.1109/TPDS.2020.3043449>.

44. Li, G.; Hari, S.K.S.; Sullivan, M.B.; Tsai, T.; Pattabiraman, K.; Emer, J.S.; Keckler, S.W. Understanding Error Propagation in Deep-Learning Neural Networks' Accelerators and Applications. *IEEE Des. Test* **2025**, *42*, 7–13. <https://doi.org/10.1109/MDAT.2025.3544537>.
45. Syed, R.T.; Zhao, Y.; Chen, J.; Andjelkovic, M.; Ulbricht, M.; Krstic, M. FPGA Implementation of a Fault-Tolerant Fused and Branched CNN Accelerator With Reconfigurable Capabilities. *IEEE Access* **2024**, *12*, 57847–57862. <https://doi.org/10.1109/ACCESS.2024.3392240>.
46. Baek, I.; Chen, W.; Zhu, Z.; Samii, S.; Rajkumar, R.R. FT-DeepNets: Fault-Tolerant Convolutional Neural Networks with Kernel-based Duplication. In *Proceedings of the 2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), Waikoloa, HI, USA, 3–8 January 2022*; IEEE: Piscataway, NJ, USA, 2022; pp. 1878–1887. <https://doi.org/10.1109/WACV51458.2022.00194>.
47. Geissler, F.; Qutub, S.; Roychowdhury, S.; Asgari, A.; Peng, Y.; Dhamasia, A.; Graefe, R.; Pattabiraman, K.; Paulitsch, M. Towards a Safety Case for Hardware Fault Tolerance in Convolutional Neural Networks Using Activation Range Supervision. *arXiv* **2021**, arXiv:2108.07019. <https://doi.org/10.48550/arXiv.2108.07019>.
48. Węgrzyn, M.; Kochan, O.; Maikiv, I. Fault Injection Tool for FPGA Reliability Testing: A Novel Method and Discovery of LUT-Specific Logical Redundancies. *Electronics* **2025**, *14*, 4600. <https://doi.org/10.3390/electronics14234600>.
49. Iurada, L.; Cavagnero, N.; Santos, F.F.D.; Averta, G.; Rech, P.; Tommasi, T. Transient Fault Tolerant Semantic Segmentation for Autonomous Driving. *arXiv* **2024**, arXiv:2408.16952. <https://doi.org/10.48550/arXiv.2408.16952>.
50. Ashmore, R.; Calinescu, R.; Paterson, C. Assuring the Machine Learning Lifecycle: Desiderata, Methods, and Challenges. *ACM Comput. Surv.* **2022**, *54*, 3453444. <https://doi.org/10.1145/3453444>.
51. Bosio, A.; Bernardi, P.; Ruospo, A.; Sanchez, E. A Reliability Analysis of a Deep Neural Network. In *Proceedings of the 2019 IEEE Latin American Test Symposium (LATS), Santiago, Chile, 11–13 March 2019*; IEEE: Piscataway, NJ, USA, 2019; pp. 1–6. <https://doi.org/10.1109/LATW.2019.8704548>.
52. Taheri, M.; Riazati, M.; Ahmadilivani, M.H.; Jenihhin, M.; Daneshlab, M.; Raik, J.; Sjödin, M.; Lisper, B. DeepAxe: A Framework for Exploration of Approximation and Reliability Trade-offs in DNN Accelerators. In *Proceedings of the 2023 24th International Symposium on Quality Electronic Design (ISQED), San Francisco, CA, USA, 5–7 April 2023*; IEEE: Piscataway, NJ, USA, 2023; pp. 1–8. <https://doi.org/10.1109/ISQED57927.2023.10129353>.
53. Gao, Z.; Gao, S.; Yao, Y.; Liu, Q.; Zeng, S.; Ge, G.; Wang, Y.; Ullah, A.; Reviriego, P. Systematic Reliability Evaluation of FPGA Implemented CNN Accelerators. *IEEE Trans. Device Mater. Reliab.* **2023**, *23*, 116–126. <https://doi.org/10.1109/TDMR.2023.3235767>.
54. Marty, T.; Yuki, T.; Derrien, S. Safe Overclocking for CNN Accelerators Through Algorithm-Level Error Detection. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2020**, *39*, 4777–4790. <https://doi.org/10.1109/TCAD.2020.2981056>.
55. Dawwd, S.A. The multi 2D systolic design and implementation of Convolutional Neural Networks. In *Proceedings of the 2013 IEEE 20th International Conference on Electronics, Circuits, and Systems (ICECS), Abu Dhabi, United Arab Emirates, 8–11 December 2013*; IEEE: Piscataway, NJ, USA, 2013; pp. 221–224. <https://doi.org/10.1109/ICECS.2013.6815394>.
56. Libano, F.; Rech, P.; Neuman, B.; Leavitt, J.; Wirthlin, M.; Brunhaver, J. How Reduced Data Precision and Degree of Parallelism Impact the Reliability of Convolutional Neural Networks on FPGAs. *IEEE Trans. Nucl. Sci.* **2021**, *68*, 865–872. <https://doi.org/10.1109/TNS.2021.3050707>.
57. Libano, F.; Wilson, B.; Wirthlin, M.; Rech, P.; Brunhaver, J. Understanding the Impact of Quantization, Accuracy, and Radiation on the Reliability of Convolutional Neural Networks on FPGAs. *IEEE Trans. Nucl. Sci.* **2020**, *67*, 1478–1484. <https://doi.org/10.1109/TNS.2020.2983662>.
58. Arechiga, A.P.; Michaels, A.J. The Robustness of Modern Deep Learning Architectures against Single Event Upset Errors. In *Proceedings of the 2018 IEEE High Performance extreme Computing Conference (HPEC), Waltham, MA, USA, 25–27 September 2018*; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6. <https://doi.org/10.1109/HPEC.2018.8547532>.
59. Khairullah, S.S.; Qassabbashi, F.N.; Kareem, J.A. Design and analysis of fault-tolerant sequential logic circuits for safety-critical applications. *Bull. Electr. Eng. Inform.* **2024**, *13*, 413–421. <https://doi.org/10.11591/eei.v13i1.5713>.
60. He, Y.; Balaprakash, P.; Li, Y. Fidelity: Efficient Resilience Analysis Framework for Deep Learning Accelerators. In *Proceedings of the 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Athens, Greece, 17–21 October 2020*; IEEE: Piscataway, NJ, USA, 2020; pp. 270–281. <https://doi.org/10.1109/MICRO50266.2020.00033>.
61. Luza, L.M.; Söderström, D.; Tsiligianis, G.; Puchner, H.; Cazzaniga, C.; Sanchez, E.; Bosio, A.; Dilillo, L. Investigating the Impact of Radiation-Induced Soft Errors on the Reliability of Approximate Computing Systems. In *Proceedings of the 2020 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), Frascati, Italy, 19–21 October 2020*; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6. <https://doi.org/10.1109/DFT50435.2020.9250865>.

62. Hoefer, J.; Kempf, F.; Hotfilter, T.; Krefß, F.; Harbaum, T.; Becker, J. SiFI-AI: A Fast and Flexible RTL Fault Simulation Framework Tailored for AI Models and Accelerators. In *Proceedings of the Great Lakes Symposium on VLSI 2023, Knoxville, TN, USA, 5–7 June 2023*; ACM: New York, NY, USA, 2023; pp. 287–292. <https://doi.org/10.1145/3583781.3590226>.
63. Li, G.; Hari, S.K.S.; Sullivan, M.; Tsai, T.; Pattabiraman, K.; Emer, J.; Keckler, S.W. Understanding error propagation in deep learning neural network (DNN) accelerators and applications. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Denver, CO, USA, 12–17 November 2017*; ACM: New York, NY, USA, 2017; pp. 1–12. <https://doi.org/10.1145/3126908.3126964>.
64. Hoang, L.-H.; Hanif, M.A.; Shafique, M. FT-ClipAct: Resilience Analysis of Deep Neural Networks and Improving their Fault Tolerance using Clipped Activation. In *Proceedings of the 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 9–13 March 2020*; IEEE: Piscataway, NJ, USA, 2020; pp. 1241–1246. <https://doi.org/10.23919/DATE48585.2020.9116571>.
65. Caro, M.; Fornt, J.; Abella, J. Efficient Diverse Redundant DNNs for Autonomous Driving. In *Proceedings of the 2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC), Torino, Italy, 27–29 June 2023*; IEEE: Piscataway, NJ, USA, 2023; pp. 18–27. <https://doi.org/10.1109/COMPSAC57700.2023.00013>.
66. Agarwal, U.K.; Chan, A.; Asgari, A.; Pattabiraman, K. Towards Reliability Assessment of Systolic Arrays against Stuck-at Faults. In *Proceedings of the 2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks—Supplemental Volume (DSN-S), Porto, Portugal, 27–30 June 2023*; IEEE: Piscataway, NJ, USA, 2023; pp. 230–236. <https://doi.org/10.1109/DSN-S58398.2023.00063>.
67. Ferlini, F.; Viel, F.; Seman, L.O.; Pettenghi, H.; Bezerra, E.A.; Leithardt, V.R.Q. A Methodology for Accelerating FPGA Fault Injection Campaign Using ICAP. *Electronics* **2023**, *12*, 807. <https://doi.org/10.3390/electronics12040807>.
68. Sharif, U.; Mueller-Gritschneider, D.; Stahl, R.; Schlichtmann, U. Efficient Software-Implemented HW Fault Tolerance for TinyML Inference in Safety-critical Applications. In *Proceedings of the 2023 Design, Automation & Test in Europe Conference & Exhibition (DATE), Antwerp, Belgium, 17–19 April 2023*; IEEE: Piscataway, NJ, USA, 2023; pp. 1–6. <https://doi.org/10.23919/DATE56975.2023.10137207>.
69. Xing, K. Training for ‘Unstable’ CNN Accelerator: A Case Study on FPGA. *arXiv* **2018**, arXiv:1812.01689. <https://doi.org/10.48550/ARXIV.1812.01689>.
70. Salami, B.; Onural, E.B.; Yuksel, I.E.; Koc, F.; Ergin, O.; Kestelman, A.C.; Unsal, O.; Sarbazi-Azad, H.; Mutlu, O. An Experimental Study of Reduced-Voltage Operation in Modern FPGAs for Neural Network Acceleration. In *Proceedings of the 2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Valencia, Spain, 29 June–2 July 2020*; IEEE: Piscataway, NJ, USA, 2020; pp. 138–149. <https://doi.org/10.1109/DSN48063.2020.00032>.
71. Shao, Y.; Wang, J.; Han, X.; Li, Y.; Li, Y.; Tao, Z. Research on Spaceborne Neural Network Accelerator and Its Fault Tolerance Design. *Remote Sens.* **2024**, *17*, 69. <https://doi.org/10.3390/rs17010069>.
72. Chen, Z.; Li, G.; Pattabiraman, K.; DeBardeleben, N. BinFI: An efficient fault injector for safety-critical machine learning systems. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Denver, CO, USA, 17–19 November 2019*; ACM: New York, NY, USA, 2019; pp. 1–23. <https://doi.org/10.1145/3295500.3356177>.
73. Masar, F.; Mrazek, V.; Sekanina, L. Late Breaking Result: FPGA-Based Emulation and Fault Injection for CNN Inference Accelerators. In *Proceedings of the 2025 Design, Automation & Test in Europe Conference (DATE), Lyon, France, 31 March–2 April 2025*; pp. 1–2. <https://doi.org/10.23919/DATE64628.2025.10992992>.
74. Reagen, B.; Gupta, U.; Pentecost, L.; Whatmough, P.; Lee, S.K.; Mulholland, N.; Brooks, D.; Wei, G.Y. Ares: A framework for quantifying the resilience of deep neural networks. In *Proceedings of the 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 24–29 June 2018*; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6. <https://doi.org/10.1109/DAC.2018.8465834>.
75. Tonetto, R.B.; Traiola, M.; Santos, F.F.D.; Kritikakou, A. ENFOR-SA: End-to-end Cross-layer Transient Fault Injector for Efficient and Accurate DNN Reliability Assessment on Systolic Arrays. In *Proceedings of the IEEE VLSI Test Symposium (VTS), Napa, CA, USA, 27–29 April 2026*.
76. Ratul, I.J.; Zhou, Y.; Yang, K. Accelerating Deep Learning Inference: A Comparative Analysis of Modern Acceleration Frameworks. *Electronics* **2025**, *14*, 2977. <https://doi.org/10.3390/electronics14152977>.
77. Liu, W.; Chang, C.-H. A Forward Error Compensation Approach for Fault Resilient Deep Neural Network Accelerator Design. In *Proceedings of the 5th Workshop on Attacks and Solutions in Hardware Security, Virtual Event, Republic of Korea, 19 November 2021*; ACM: New York, NY, USA, 2021; pp. 41–50. <https://doi.org/10.1145/3474376.3487281>.

78. Neto, A.V.S.; Silva, H.L.; Camargo, J.B.; Almeida, J.R.; Cugnasca, P.S. Design and Assurance of Safety-Critical Systems with Artificial Intelligence in FPGAs: The Safety ArtISt Method and a Case Study of an FPGA-Based Autonomous Vehicle Braking Control System. *Electronics* **2023**, *12*, 4903. <https://doi.org/10.3390/electronics12244903>.
79. Gao, Z.; Zhang, H.; Yao, Y.; Xiao, J.; Zeng, S.; Ge, G.; Wang, Y.; Ullah, A.; Reviriego, P. Soft Error Tolerant Convolutional Neural Networks on FPGAs With Ensemble Learning. *IEEE Trans. Very Large Scale Integr. VLSI Syst.* **2022**, *30*, 291–302. <https://doi.org/10.1109/TVLSI.2021.3138491>.
80. Souvatzoglou, I.; Papadimitriou, A.; Sari, A.; Vlagkoulis, V.; Psarakis, M. Analyzing the Single Event Upset Vulnerability of Binarized Neural Networks on SRAM FPGAs. In *Proceedings of the 2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Athens, Greece, 6–8 October 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–6. <https://doi.org/10.1109/DFT52944.2021.9568280>.
81. Matsubara, K.; Lieske, H.; Kimura, M.; Nakamura, A.; Koike, M.; Morikawa, S.; Hotta, Y.; Irita, T.; Mochizuki, S.; Hamasaki, H.; et al. A 12-nm Autonomous Driving Processor With 60.4 TOPS, 13.8 TOPS/W CNN Executed by Task-Separated ASIL D Control. *IEEE J. Solid-State Circuits* **2022**, *57*, 115–126. <https://doi.org/10.1109/JSSC.2021.3120191>.
82. Talpes, E.; Sarma, D.D.; Venkataramanan, G.; Bannon, P.; McGee, B.; Floering, B.; Jalote, A.; Hsiong, C.; Arora, S.; Gorti, A.; et al. Compute Solution for Tesla's Full Self-Driving Computer. *IEEE Micro* **2020**, *40*, 25–35. <https://doi.org/10.1109/MM.2020.2975764>.
83. Goldstein, B.F.; Ferreira, V.C.; Srinivasan, S.; Das, D.; Nery, A.S.; Kundu, S.; França, F.M. A Lightweight Error-Resiliency Mechanism for Deep Neural Networks. In *Proceedings of the 2021 22nd International Symposium on Quality Electronic Design (ISQED)*, Santa Clara, CA, USA, 7–9 April 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 311–316. <https://doi.org/10.1109/ISQED51717.2021.9424287>.
84. Taheri, M.; Daneshmand, M.; Raik, J.; Jenihhin, M.; Pappalardo, S.; Jimenez, P.; Deveautour, B.; Bosio, A. SAFFIRA: A Framework for Assessing the Reliability of Systolic-Array-Based DNN Accelerators. In *Proceedings of the 2024 27th International Symposium on Design & Diagnostics of Electronic Circuits & Systems (DDECS)*, Kielce, Poland, 3–5 April 2024; IEEE: Piscataway, NJ, USA, 2024; pp. 19–24. <https://doi.org/10.1109/DDECS60919.2024.10508925>.
85. Tsai, T.; Hari, S.K.S.; Sullivan, M.; Villa, O.; Keckler, S.W. NVBitFI: Dynamic Fault Injection for GPUs. In *Proceedings of the 2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Taipei, Taiwan, 21–24 June 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 284–291. <https://doi.org/10.1109/DSN48987.2021.00041>.
86. Chen, Z.; Li, G.; Pattabiraman, K. A Low-cost Fault Corrector for Deep Neural Networks through Range Restriction. *IEEE Des. Test* **2025**. <https://doi.org/10.1109/MDAT.2025.3618758>.
87. Mahmoud, A.; Hari, S.K.S.; Fletcher, C.W.; Adve, S.V.; Sakr, C.; Shanbhag, N.; Molchanov, P.; Sullivan, M.B.; Tsai, T.; Keckler, S.W. HarDNN: Feature Map Vulnerability Evaluation in CNNs. *arXiv* **2020**, arXiv:2002.09786. <https://doi.org/10.48550/arXiv.2002.09786>.
88. Sabogal, S.; George, A.; Crum, G. Reconfigurable Framework for Resilient Semantic Segmentation for Space Applications. *ACM Trans. Reconfigurable Technol. Syst.* **2021**, *14*, 3472770. <https://doi.org/10.1145/3472770>.
89. Alama, Y.A.; Sakpal, S.; Wang, K.; Bunesco, R.; Karanth, A.; Louri, A. Algorithmic Strategies for Sustainable Reuse of Neural Network Accelerators with Permanent Faults. In *Proceedings of the 2025 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Barcelona, Spain, 21–23 October 2025; IEEE: Piscataway, NJ, USA, 2025; pp. 1–6. <https://doi.org/10.1109/DFT66274.2025.11257575>.
90. Burel, T.S.; Evans, T.A.; Anghel, L. Improving DNN Fault Tolerance in Semantic Segmentation Applications. In *Proceedings of the 2022 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Austin, TX, USA, 19–21 October 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–6. <https://doi.org/10.1109/DFT56152.2022.9962354>.
91. Jonckers, N.; Vinck, T.; Dekkers, G.; Karsmakers, P.; Prinzie, J. Single-Event Upset Analysis of a Systolic Array based Deep Neural Network Accelerator. *arXiv* **2024**, arXiv:2405.15381. <https://doi.org/10.48550/arXiv.2405.15381>.
92. Owada, Y.; Tomioka, Y.; Saito, H. Dual Modular Redundancy Unit of Convolutional Layer for Low-cost and Reliable CNNs. In *Proceedings of the 2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Prague, Czech Republic, 9–12 October 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 379–384. <https://doi.org/10.1109/SMC53654.2022.9945598>.
93. Avizienis, A.; Laprie, J.-C.; Randell, B.; Landwehr, C. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. Dependable Secur. Comput.* **2004**, *1*, 11–33. <https://doi.org/10.1109/TDSC.2004.2>.
94. Sabogal, S.; George, A.; Crum, G. ReCoN: A Reconfigurable CNN Acceleration Framework for Hybrid Semantic Segmentation on Hybrid SoCs for Space Applications. In *Proceedings of the 2019 IEEE Space Computing Conference (SCC)*, Pasadena, CA, USA, 30 July–1 August 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 41–52. <https://doi.org/10.1109/SpaceComp.2019.00010>.

95. Luo, Y.; Saberi, A.K.; Bijlsma, T.; Lukkien, J.J.; Van Den Brand, M. An architecture pattern for safety critical automated driving applications: Design and analysis. In *Proceedings of the 2017 Annual IEEE International Systems Conference (SysCon)*, Montreal, QC, Canada, 24–27 April 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1–7. <https://doi.org/10.1109/SYSCON.2017.7934739>.
96. Burel, S.; Evans, A.; Anghel, L. Zero-Overhead Protection for CNN Weights. In *Proceedings of the 2021 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Athens, Greece, 6–8 October 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–6. <https://doi.org/10.1109/DFT52944.2021.9568363>.
97. Salami, B.; Unsal, O.S.; Kestelman, A.C. On the Resilience of RTL NN Accelerators: Fault Characterization and Mitigation. In *Proceedings of the 2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, Lyon, France, 24–27 September 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 322–329. <https://doi.org/10.1109/CAHPC.2018.8645906>.
98. Khoshavi, N.; Roohi, A.; Broyles, C.; Sargolzaei, S.; Bi, Y.; Pan, D.Z. SHIELDeNN: Online Accelerated Framework for Fault-Tolerant Deep Neural Network Architectures. In *Proceedings of the 2020 57th ACM/IEEE Design Automation Conference (DAC)*, San Francisco, CA, USA, 20–24 July 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6. <https://doi.org/10.1109/DAC18072.2020.9218697>.
99. Du, B.; Azimi, S.; De Sio, C.; Bozzoli, L.; Sterpone, L. On the Reliability of Convolutional Neural Network Implementation on SRAM-based FPGA. In *Proceedings of the 2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Noordwijk, The Netherlands, 2–4 October 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–6. <https://doi.org/10.1109/DFT.2019.8875362>.
100. Bruhn, F.C.; Tsog, N.; Kunkel, F.; Flordal, O.; Troxel, I. Enabling radiation tolerant heterogeneous GPU-based onboard data processing in space. *CEAS Space J.* **2020**, *12*, 551–564. <https://doi.org/10.1007/s12567-020-00321-9>.
101. Biondi, A.; Nesti, F.; Cicero, G.; Casini, D.; Buttazzo, G. A Safe, Secure, and Predictable Software Architecture for Deep Learning in Safety-Critical Systems. *IEEE Embed. Syst. Lett.* **2020**, *12*, 78–82. <https://doi.org/10.1109/LES.2019.2953253>.
102. Debouk, R. Overview of the Second Edition of ISO 26262: Functional Safety—Road Vehicles. *J. Syst. Saf.* **2019**, *55*, 13–21. <https://doi.org/10.56094/jss.v55i1.55>.
103. Rech, P. Artificial Neural Networks for Space and Safety-Critical Applications: Reliability Issues and Potential Solutions. *IEEE Trans. Nucl. Sci.* **2024**, *71*, 377–404. <https://doi.org/10.1109/TNS.2024.3349956>.
104. Biondi, A.; Balsini, A.; Pagani, M.; Rossi, E.; Marinoni, M.; Buttazzo, G. A Framework for Supporting Real-Time Applications on Dynamic Reconfigurable FPGAs. In *Proceedings of the 2016 IEEE Real-Time Systems Symposium (RTSS)*, Porto, Portugal, 29 November–2 December 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1–12. <https://doi.org/10.1109/RTSS.2016.010>.
105. Bolchini, C.; Cassano, L.; Miele, A.; Toschi, A. Fast and Accurate Error Simulation for CNNs against Soft Errors. *IEEE Trans. Comput.* **2023**, *72*, 984–997. <https://doi.org/10.1109/TC.2022.3184274>.
106. Bernardeschi, C.; Cassano, L.; Domenici, A. SRAM-Based FPGA Systems for Safety-Critical Applications: A Survey on Design Standards and Proposed Methodologies. *J. Comput. Sci. Technol.* **2015**, *30*, 373–390. <https://doi.org/10.1007/s11390-015-1530-5>.
107. Zhu, Q.; Li, W.; Huang, C.; Chen, X.; Zhou, W.; Wang, Y.; Li, J.; Fu, F. Verification and Design of Robust and Safe Neural Network-enabled Autonomous Systems. In *Proceedings of the 2023 59th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Monticello, IL, USA, 27–29 September 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 1–8. <https://doi.org/10.1109/Allerton58177.2023.10313451>.
108. Stetter, R.; Pulm, U. Strategies and Methods for the Fault-Tolerant Function Development of Multi-Domain Systems. *Appl. Sci.* **2024**, *14*, 11646. <https://doi.org/10.3390/app142411646>.
109. Li, L.; Yuan, Y.; Zhang, X.; Wu, S.; Zhang, T. Fault-Tolerant Control Scheme for the Sensor Fault in the Acceleration Process of Variable Cycle Engine. *Appl. Sci.* **2022**, *12*, 2085. <https://doi.org/10.3390/app12042085>.
110. Brando, A.; Serra, I.; Mezzetti, E.; Cazorla, F.J.; Perez-Cerrolaza, J.; Abella, J. On Neural Networks Redundancy and Diversity for Their Use in Safety-Critical Systems. *Computer* **2023**, *56*, 41–50. <https://doi.org/10.1109/MC.2023.3236523>.
111. Chang, S.-W.; Kim, D.-S. Scalable Transformer Accelerator with Variable Systolic Array for Multiple Models in Voice Assistant Applications. *Electronics* **2024**, *13*, 4683. <https://doi.org/10.3390/electronics13234683>.
112. Mishty, K.; Sadi, M. System and Design Technology Co-optimization of Chiplet-based AI Accelerator with Machine Learning. In *Proceedings of the Great Lakes Symposium on VLSI 2023*, Knoxville, TN, USA, 5–7 June 2023; ACM: New York, NY, USA, 2023; pp. 697–702. <https://doi.org/10.1145/3583781.3590233>.
113. Mishty, K.; Sadi, M. Chiplet-Gym: Optimizing Chiplet-Based AI Accelerator Design With Reinforcement Learning. *IEEE Trans. Comput.* **2025**, *74*, 43–56. <https://doi.org/10.1109/TC.2024.3457740>.

114. Malhotra, A.; Wang, C.; Gupta, S.K. TFix: Exploiting the Natural Redundancy of Ternary Neural Networks for Fault Tolerant In-Memory Vector Matrix Multiplication. In *Proceedings of the 2023 60th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 9–13 July 2023*; IEEE: Piscataway, NJ, USA, 2023; pp. 1–6. <https://doi.org/10.1109/DAC56929.2023.10247835>.
115. Li, H.; Wang, Y. Reliability of LLM Inference Engines from a Static Perspective: Root Cause Analysis and Repair Suggestion via Natural Language Reports. *Big Data Cogn. Comput.* **2026**, *10*, 60. <https://doi.org/10.3390/bdcc10020060>.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.