A Survey on Fault-Tolerant Methodologies for Deep Neural Networks

Rizwan Tariq Syed, Markus Ulbricht, Krzysztof Piotrowski, Milos Krstic
IHP – Leibniz-Institut f¨ur innovative Mikroelektronik, Frankfurt (Oder), Germany

Abstract: Asignificant rise in Artificial Intelligence (AI) has impacted many applications around us, so much so that AI has now been increasingly used in safety-critical applications. AI at the edge is the reality, which means performing the data computation closer to the source of the data, as opposed to performing it on the cloud. Safety-critical applications have strict reliability requirements; therefore, it is essential that AI models running on the edge (i.e., hardware) must fulfill the required safety standards. In the vast field of AI, Deep Neural Networks (DNNs) are the focal point of this survey as it has continued to produce extraordinary outcomes in various applications .i.e medical, automotive, aerospace, defense, etc. Traditional reliability techniques for DNNs implementation are not always practical, as they fail to exploit the unique characteristics of the DNNs. Furthermore, it is also essential to understand the targeted edge hardware because the impact of the faults can be different in ASICs and FPGAs. Therefore, in this survey, first, we have examined the impact of the fault in ASICs and FPGAs, and then we seek to provide a glimpse of the recent progress made towards the fault-tolerant DNNs. We have discussed several factors that can impact the reliability of the DNNs. Further, we have extended this discussion to shed light on many state-of-the-art fault mitigation techniques for DNNs.

Keywords: fault tolerance, reliability, FPGAs, ASICs, neural networks

1. Introduction

Deep Learning (DL), as a subset of Machine Learning, has revolutionized many tasks in recent years, ranging from data analytics, natural language processing, image classification, video processing to speech recognition, recommender systems, etc. These DL models learn from the data during the training phase and make output predictions during the inference phase. In cases such as image classification, DNN algorithms have surpassed human-level accuracy. This and many other similar breakthroughs in deep learning have motivated researchers to explore deep learning in safety-critical applications, i.e., automotive, space, defense, drones, industry, and health.

DNN models will eventually be deployed on the hardware (ASICs, FPGAs). There are two significant challenges at hand in deploying these models on hardware. a) First is the resource--intensive nature of running deep learning models on the hardware. Deep Learning models are getting bigger and bigger to solve more complex problems. The training phase of the model generally happens on the cloud or powerful machines. During the inference phase, DNN models are deployed on the target hardware. The target hardware should be able to process a massive amount of data, i.e., roughly 4 TB in case of a self-driving car scenario. b) The second challenge comes in the form of the reliability of the hardware on which these models are deployed. The reliability analysis of the DNNs is extremely important for the reason that, during the inference phase, a DNN model needs to take essential and critical decisions. e.g., apply brakes if a pedestrian is detected in front of the car.

To fulfill our high-performance needs, over the years, we have observed the trend of transistor size shrinking to the Very Deep Sub-Micron (VDSM) level. On the one hand, this scaling has increased the computing performance and helped move the DNN inference processing from the cloud to the edge. On the other hand, transistor scaling leads to increased sensibility to transient faults due to lower threshold voltages and tighter noise margins [42]. Previously, this problem was relevant to the hostile environments, i.e., space, but now due to transistors scaling to VDSM level, safety-critical applications at the ground level are also prone to transient faults [8]. Most of the studies are based on single-bit errors. Due to the technology scaling, multi-bit errors are also on the rise [30]. Thus, for a reliable DNN inference on edge, a thoroughly verified fault-tolerant hardware is required.

In this survey, we have investigated state-of-the-art fault-tolerant methodologies for characterizing and improving the resilience of DNN algorithms processing on edge. Figure 1 provides an overview of the study. Starting from section 2, we have discussed the impact of faults in integrated circuits. This section

Autor korespondujący:
Krzysztof Piotrowski, piotrowski@ihp-microelectronics.com

Artykuł recenzowany
nadesłany 19.02.2023 r., przyjęty do druku 16.04.2023 r.

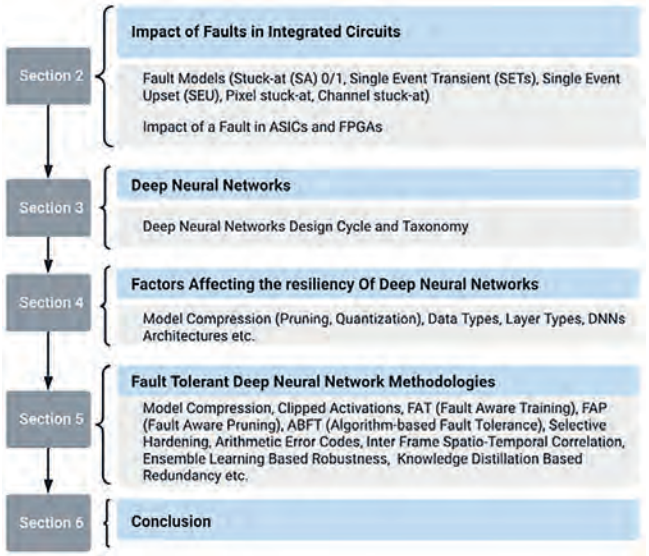


Fig. 1. Organization of this survey
Rys. 1. Organizacja niniejszego przeglądu

includes the discussion related to the importance and classifications of fault models. Additionally, there is a discussion around the impact of fault in ASICs and FPGAs. Understanding the difference between these two hardware platforms is essential because a fault impact in ASICs and FPGAs can be different. Section 3 has shed some light on the deep neural networks design cycle and taxonomy. A discussion about MLPs, CNNs, and SNNs is also part of the section. In section 4 and 5, we have extended the discussion towards the fault resiliency of the deep neural networks and various fault-tolerant methodologies. There are so many other factors that can affect the reliability of the deep learning models. i.e., network architecture, layer type, data type, bit position, pruning, quantization, and depth of the model. Therefore, it is necessary to analyze the DNNs from different perspectives. Section 6 concludes this survey.

2. Impact of Faults in Integrated Circuits

2.1. Fault Models

It is not easy to identify all the potential types of faults which can occur in an electronic circuit. To evaluate a design against faults, faults are assumed to behave according to some fault model [14]. A fault model attempts to describe the effect of the fault that can occur. With the help of the fault model, the design engineers can efficiently predict the consequences of this particular fault.

Electronic chips are becoming increasingly complex. Computation demands are increasing day by day increase because of the rise in AI workloads. These modern highperformance chips consist of billions of transistors. The physical defects can be of different types. It is very extremely difficult to investigate all possible faults. The advantages of fault models are: a) It helps analyze the circuit behavior under a given fault model. b) Drastically reduces the number of faults to be considered. c) Fault Coverage of design. d) It enables the designer to find the root cause of the failure of a design. e) With fault simulations, it helps generate the test metrics, which helps in the reliability evaluation of the design. Some of the most commonly used fault models are discussed below.

Stuck-at (SA) 0/1: The stuck-at fault model is one of the most common fault models used for the reliability analysis of a VLSI design. In this fault model, if a circuit line is permanently stuck at 'logic low', it will be called Stuckat-0, and if it is permanently stuck at 'logic high', it will be termed Stuck-at-one. These faults are permanent in nature and are caused by post-manufacturing defects and transistor aging [41].

Single Event Transient (SET): The phenomenon of the Single Event Transient (SET) occurs when a high-energy particle strikes on a combinational circuit, it causes a transient voltage disturbance due to charge deposition. If the energy of the particle crosses a certain threshold, the end effect of it is a Single Event Transient (Glitch) in the combination circuit. SETs can occur in both ASICs and FPGAs. These transient faults are temporary and are also called soft errors. In [13] was investigated thet probability of SETs becoming an SEU. Generally, the analysis of SETs is very complex in large designs, which are composed of many paths. Techniques such as Timing Analysis can be used to investigate the SETs in large and complex designs.

Single Event Upset (SEU): We have discussed above the cause of transient faults. If the same transient fault can propagate to a storage element and gets latched, it becomes a Single Event Upset (SEU). Storage elements can be system memory, registers, or configuration memory cells in FPGAs. Based on the number of upsets that occur in a circuit, SEUs can lead to first, second, and third-order effects. A single SEU affecting a single bit is often classified as a first-order effect. When a charged particle affects multiple bits, it is considered as Multi-bit upset (MBU) and leads to second and third-order effects. A secondorder effect happens if an SEU simultaneously strikes two adjacent sensitive nodes located in two different memory cells. And when MBU occurs as a result of a single particle striking two adjacent sensitive nodes located in the same memory cell, it is considered a third-order effect.

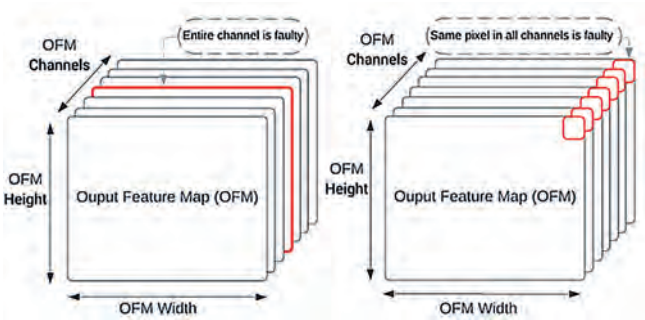


Fig. 2. Left: Channel stuck-at, Right: Pixel stuck-at [43]
Rys. 2. Po lewej: zablokowany kanał, po prawej: Piksel zablokowany

Sometimes fault model depends on the targeted application and how that application is implemented on the hardware. For instance, in [43] has proposed two fault models (Fig. 2) for the deep convolutional neural implementation on the FPGAs. a) Pixel stuck-at, which means the stuck-at fault in a single pixel of the CNN feature maps. b) Channel stuck-at, which means the whole channel in CNN feature maps is faulty.

2.2. Impact of a Fault in ASICs and FPGAs

FPGAs are becoming a valuable candidates for AI applications because of their high density, high performance, shorter time to market, and re-programmability. On the other hand, ASICs, which stand for Application-Specific Integrated Circuits, are designed for a specific application, and their functionality remains the same throughout their operating life.

In ASICs, the logic is permanently mapped to gates and flip-flops in silicon. Whereas in FPGAs, logic is mapped on the configurable logic blocks (CLBs). CLBs consist of Lookup Tables (LUTs), flip-flops (FFs), and routing resources (switch matrix, multiplexors, and connection segments). Most FPGAs also have dedicated memory blocks as hard macros called block RAMs (BRAMs). Unlike ASICs, FPGAs are programmable, and their functionality can be changed by uploading a new bitstream. The

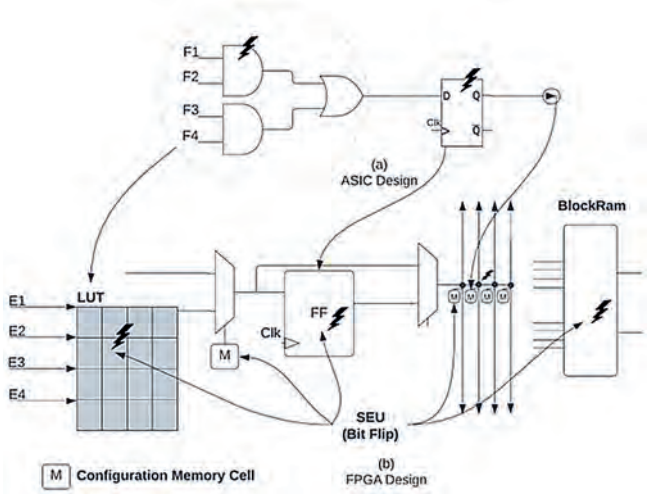


Fig. 3. Impact of a fault in ASIC and FPGA architecture [21]
Rys. 3. Wpływ błędu w architekturze ASIC i FPGA

bitstream contains configuration frames, which configure all the programmable and memory elements in the FPGA fabric. All these bits are potentially sensitive to radiation effects; therefore, the design should be thoroughly investigated against various fault models.

In [21] has investigated the effect of radiation in ASICs and FPGAs. Figure 3 has illustrated that effect of radiation in the combinational and sequential logic of ASICs is transient. Based on the pulse duration of the SETs (Glitch), transient faults in the combinational logic of the design may or may not be latched by a storage cell. On the other hand, faults in the sequential logic (SEUs) remain in the storage cell until the next load.

In the case of SRAM-based FPGAs, the user's logic is mapped on the CLBs. CLBs consist of LUTs, FFs, and routing resources. An SEU in the LUT memory cell modifies the implemented combinational logic and results in undesired program behavior. The end effect of SEUs in the CLBs is permanent, and it can only be mitigated by re-programming the bitstream. Similarly, an SEU in the routing matrix will lead to connecting/disconnecting a connection between CLBs. This can also be mitigated using bitstream reconfiguration. An SEU in block RAMs also has a permanent effect; therefore, block RAMs should be protected against faults using different error-detecting and correcting techniques. In a scenario, when an SEU occurs in the sequential logic synthesized in the FPGA, it will have a temporary effect, as the faulty value will be overwritten in the next load of the flip flop [21].

As we have discussed, the impact of the fault is sometimes different in ASICs and FPGAs for the reason that FPGA fabric is a bit different compared to ASICs. Several fault-tolerant methodologies, which have been proposed for ASICs, may not be directly applicable to FPGAs. Therefore, it is extremely important that fault modeling and fault analysis of a design should be done based on the targeted hardware platform. A clear understanding of the targeted hardware platform can also help in the design and development of relevant fault mitigation methodologies.

3. Deep Neural Networks

Artificial Intelligence (AI) has created an enormous impact on all aspects of life. AI, as described in [12], "is a system's ability to interpret external data correctly, to learn from such data, and to use those learnings to achieve specific goals and tasks through flexible adaptation." Machine Learning is a subset of AI, which deals with computer algorithms that can train a model to perform tasks and take decisions without explicitly programming to do so. Deep Learning comes under the umbrella of Machine Learning (ML), and it uses Artificial Neural Networks (ANNs), whose architecture is inspired by the structure and function of the brain. ANNs have recently become the standard tool for solving a variety of prediction and classification problems. They generally consist of an input layer, an output layer, and hidden layers. In past years, ANNs have grown in complexity, comprising of many hidden layers, and are able to solve many complex problems in computer vision, natural language processing, and medical science, etc. ANNs are also commonly known as Deep Neural Networks. The term 'Deep' refers to the use of multiple layers in the ANNs. Each layer consists of neurons that connect to other neurons in the corresponding layers via an activation function. Each neuron has its associated parameters, i.e., weight, bias, and/or filter coefficient. Authors of [23, 40] have investigated the ANNs in detail.

3.1. Deep Neural network design cycle

The design cycle of DNNs consists of two major stages: Training and Inference.

Training: DNNs model should be trained before its deployment on the targeted device. Training is a computeintensive process, generally carried out by high-performance computing machines, i.e., cloud servers, which involves the use of a training data set to find suitable values for the network parameters. After the training is done, the performance of the model is tested against a test data set.

Inference: After the model is trained and tested, it is ready to be deployed. At this stage, the NN performs classification/decision-making using actual, previously unseen data (i.e., in real-time). Target hardware for inference can vary based on the application. For applications such are movie recommendations on Netflix or social media Ads, inference happens on the cloud. In comparison, inference happens on edge in the case of Cyber-Physical Systems (e.g., autonomous vehicles and wearable healthcare devices).

3.2. Neural Network Taxonomy

Since their advent, NNs have progressively improved. The first generation was single-layer perceptron or multi-layer perceptron (MLP). MLP is also called a feedforward neural network for the reason that nodes in the network do not form a cycle. MLP can generally perform classification and regression problems.

The second-generation (Fig. 4) of the neural networks consists of convolutional neural networks (CNNs), RNNs, capsule networks (CapsNets) [16], and generative adversarial networks

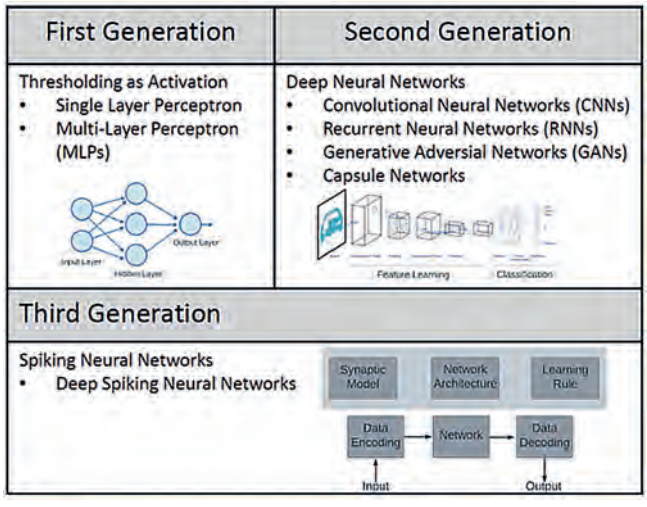


Fig. 4. 1st, 2nd and 3rd neural networks generations
Rys. 4. Pierwsza, druga i trzecia generacja sieci neuronowych

(GANs) [11]. CNNs have proven so effective in solving an image classification problem. CNNs have the ability to develop an internal representation of a two-dimensional image. This allows the CNN model to learn the position and scale-invariant structures in the image data, which is very important when working with images. Various CNN models have been proposed, namely, VGG16 [38], Alexnet [22], Resnet [15], etc., which have shown state-of-the-art performance against the ImageNet dataset [35].

The third generation (Fig. 4) of neural networks makes use of spiking NNs (SNNs) [29] in an attempt to emulate human brain-like functioning. The major difference between a traditional ANN and SNN is how the information propagates through the network. Therefore, instead of continuously changing in time values used in artificial neural networks, spiking NNs works with discrete events that occur at specific points of time. Spiking NNs receives a series of spikes as input and produces a series of spikes as the output, also referred to as spike trains [29]. The emphasis of this survey is on MLPs and CNNs.

4. Factors Affecting the Resiliency of Deep Neural Networks

We have discussed in the section 2 how the hardware is vulnerable and leads to incorrect results in the presence of faults. In the section 3, we have briefly explained the DNNs, their design cycle, and taxonomy. Eventually, the DNNs will be implemented on hardware. Therefore it is essential to analyze the neural networks under the influence of hardware faults. Deep Neural Networks are said to have some inherent resiliency. However, faults can still influence the accuracy of the DNN model, which can further lead to incorrect output classification or prediction. Therefore, for safety-critical applications, it becomes vital to do a thorough design analysis.

The term resiliency in the DNNs refers to the ability to maintain a given accuracy even in the presence of errors. There are many factors that can affect the resiliency of deep neural networks. i.e., Network Architecture, Layer Type, Data Type, the bit position of weights, pruning, quantization, etc. Authors of [6, 24] have shown that deeper networks are more resilient, and the use of batch normalization layers in the neural network architecture helps in generalizing and improving the resiliency of the network model. In [34] was figured that, that the impact of fault is more when it happens at the back of the network (i.e., in the last layers), whereas faults effects tend to be mitigated or neutralized if happening in the initial layers of the network (i.e., the first layer). In [6, 27] was demonstrated that pruning and quantization also assist in increasing the resiliency of the network model.

Datatype also has an enormous role to play in the resiliency of the DNN model. As shown in Fig. 5, we analyze how a fault can impact a DNN with an IEEE-754 floatingpoint 32 (FP32) data type versus a 4-bit Fixed Point (FxP) data type. According to the IEEE-754 standard, the FP32 data type consists of 8 exponent bits, 23 fraction bits 1 sign bit. On the other hand, the 4-bit FxP data type consists of 1 sign bit and 3 fraction bits. As discussed in the previous section, the DNN model consists of thousands of parameters. Consider 0.25 as one of the parameters (weight) of the DNN model (Fig. 5). If the weight value is represented as an FP32 number, a fault in the most significant exponent bit of the FP32 number can substantially change the value of the DNN's parameter either to a very high value or to a very low value. If not masked, this fault could propagate through the DNN network and drastically decrease the accuracy. The impact of a fault also causes a deviation in FxP numbers, which leads to a decrease in accuracy, but the overall impact would be less due to the less dynamic range of the FxP numbers [39]. Therefore it is crucial to define a data type and bit-width, which can fulfill the requirement of accuracy and reliability, and hardware resources.

Another aspect that impacts the reliability of the DNN model is the hardware architecture implemented on the targeted hardware, e.g., ASICs, FPGAs. The faults can occur in the datapath, i.e., latches, flip flops, etc., and also in the data buffers. Faults in both locations propagate differently. Faults in the data path will be read once and can get over-written by the correct value in the next load. Whereas faults in the buffers may be read multiple Times because of the reuse (reuse of weights, input feature maps, output feature maps, etc.), and hence the same faulty value can be spread to multiple locations very quickly [24].

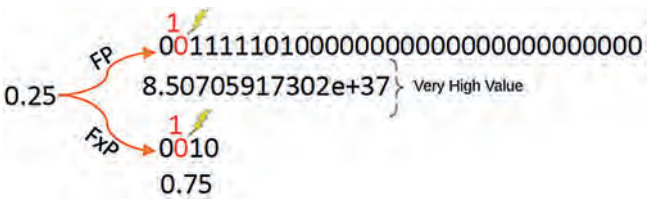
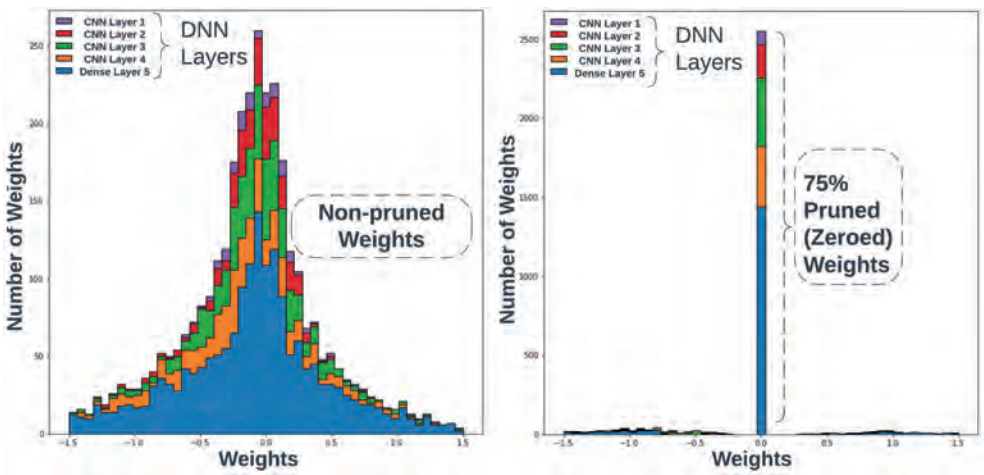


Fig. 5. Impact of a fault in floating point and fixed point number [39]
Rys. 5. Wpływ błędu na liczbę zmiennoprzecinkową i stałoprzecinkową

5. Fault Tolerant Deep Neural Network Methodologies

In this section, we will explore several state-of-the-art. fault--tolerant methodologies which were proposed by the corresponding research community.

Fig. 6. Pruning impact on the weights distribution of the CNN model
Rys. 6. Wpływ przycinania na rozkład wag modelu CNN



5.1. Model Compression

The challenge is that running deep learning models is a resource-intensive process and deploying these models, with millions of parameters, on edge devices is a growing concern. There are DNN models from a few thousand parameters to more than a billion parameters. Deploying these big models on the hardware is very challenging, especially in safety-critical applications. Due to these challenges, research in the area of model compression has been very actively pursued over the last few years. The goal of model compression is to reduce the model size so that it can be deployed on low power and resource constraint devices without a significant accuracy drop. Some model compression methods which has been proposed in recent years are parameter pruning, quantization, knowledge distillation, low-rank factorization, transferred/compact convolutional filters, etc. In this study, we focus on the two popular two model compression methods: a) Pruning b) Quantization. Pruning and quantization were not particularly proposed as a method to improve resiliency. It is exciting to study them for two reasons. a) Pruning and quantization affect the resiliency of the DNNs. b) Both methods have become the de-facto standard during the DNN deployment on the hardware; therefore, it is crucial to study it from the fault tolerance perspective.

Pruning: Many experiments have concluded that there are many parameters in the DNNs which are not important, and it is still possible to achieve the desired performance in the absence of these parameters. Thus, pruning is a way to remove unnecessary parameters, thereby making the deep neural networks sparsed. Figure 6 illustrates the impact of the magnitude-based weight pruning method, which gradually zeroes out weights of the model during the training process to achieve model sparsity. This sparsity in the neural network parameters due to the pruning has two advantages. a) It reduces the mode size, which further helps in reducing the computational complexity, leading to faster inference. b) It improves the resiliency of the DNN model [6]. Pruning is further classified into channel pruning, filter pruning, connection pruning, and layer pruning. Different pruning strategies can also have a different resiliency impact on neural networks.

Quantization: A typical deep neural network consists of weights in 32-bit floating-point values. FP32 computations require either a floating-point unit or additional hardware resources to perform dynamic range shifts computation. This will lead to an increase in hardware resources and latency, which gains makes it challenging to deploy these networks in hardware devices. DNN quantization comes to the rescue in this situation. DNN quantization refers to a method of approximating a neural network's parameters and activations to low bit-width fixed point (FxP) numbers as shown in Fig. 7. Because, in many cases, the dynamic range that the FP32 provides is not needed. FxP numbers are generally hardware-friendly. FxP computations are faster than FP32, and it also costs less area overhead as compared to FP32 Computations. Along with the benefits such as (a) lower model size (b) lower inference latency, DNN Quantization also results in improving the resiliency of the DNN Model.

Goldstein have studied the impact of SEUs in three different CNNs models with different sparsity [9]. They have concluded that Pruning and Quantization combined can increase the resiliency by up to 108.7 times. Other authors have explored further the impact of SEUs in homogenous and heterogeneous quantized models and concluded that, in general, quantization helps in improving the resiliency of the DNN model [29]. Resiliency between the models can also vary based on different levels of quantization and more vigorous quantization could sacrifice resiliency and accuracy. In [36] have aggressively quantized the DNN models (VGG16 and Lenet) to as low as the binary values and have reported an increase in the fault resiliency of DNN models (VGG16 and Lenet) by 10000x.

5.2. Clipped Activations

We have examined in section 4 Fig. 5, how a fault can impact a DNN with an FP32 data type versus a 4-bit FxP data type and illustrated that fault in the MSB of the exponent bit can saturate the DNN model and lead to undesirable results. In [18] was observed a similar impact of fault on the FP32 as illustrated in Fig. 5 and to solve the problem of saturation the technique of 'The Clipped Activations Function' has been proposed. By default, the output of the activation function is unbounded; therefore, in the presence of a fault, the faulty output of extremely high magnitude can propagate through the network. They have replaced the unbounded activations functions with a bounded activations function to restrict the output of the activation function to a specific threshold value. With this methodology, they have achieved 68.92 % improvement in accuracy compared to the baseline VGG-16 model at 1×10^{-2} fault rate.

Similarly, in [2, 7] have also explored the similar phenomenon of restricting the output of neurons to make them more resilient. Restricting the output of the neurons results in reduced deviations. Neural networks have the capability to tolerate small deviations due to their inherent resiliency.

5.3. FAT: Fault Aware Training

Machine Learning deals with computer algorithms that have the ability to learn to perform tasks and take decisions without explicitly programming to do so. The process of learning is called 'Training'. Authors in [43] have leveraged this idea and proposed Fault Aware Training (FAT). In other words, Fault Injection can be performed during the training phase. They treat resiliency as a learning problem, and they want the neural network to learn

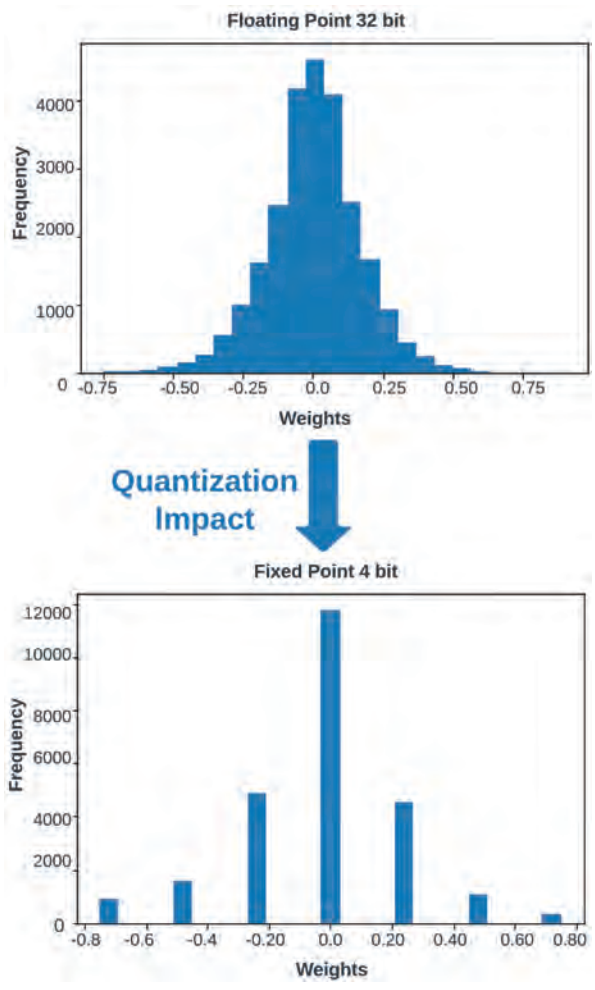


Fig. 7. Quantization impact on weights distribution [29]
Rys. 7. Wpływ kwantyzacji na rozkład wag

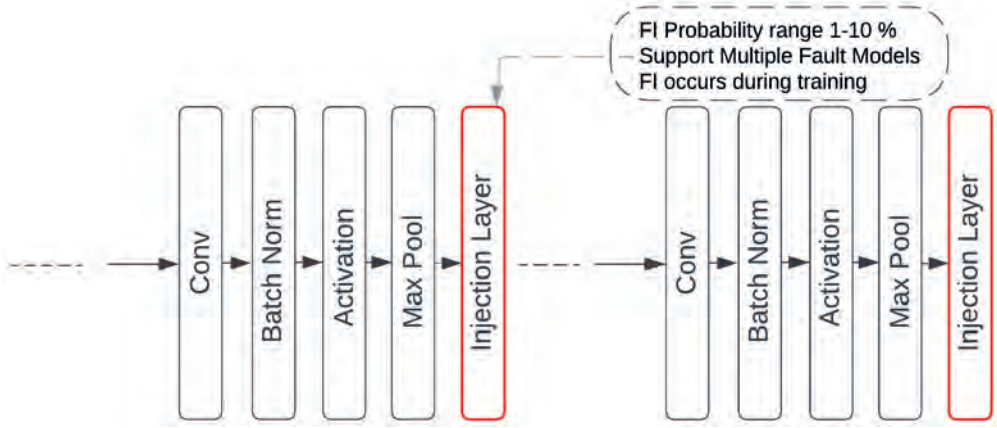


Fig. 8. Fault injection layer in DNN architecture [43]
Rys. 8. Warstwa wstrzykiwania błędów w architekturze DNN

the impact of faults during the training phase. Fault Injection layers have been added inside the DNN model (Fig. 8). Faults are injected into the neural network during the training phase with a probability in the range of 1–10 %.

There is the concept of using Dropout layers in the training of the neural networks to reduce overfitting and make the behavior of the model more generic. The fault injection layer during FAT behaves somewhat similarly to Dropout layers. FAT supports injecting many possible fault values into the DNNs, while dropout only inserts zeros during the training. The detailed analysis of two different fault models under the influence of faults with different probabilities is discussed in the paper.

5.4. FAP and FAP+T

In [44] was considered a DNN accelerator based on a systolic array, i.e., Google Tensor Processing Unit (TPU), and proposed two fault-tolerant methodologies. a) Fault Aware Pruning (FAP) b) Fault Aware Pruning + Retraining (FAP+T). They have considered permanent faults, which occurred in the integrated circuits due to process variations or manufacturing defects. After detailed gate-level simulations of stuck-at faults, they have concluded that the accuracy of the TPU drops drastically even if there are just four faulty MAC units among a total of 64K MAC units.

FAP: Pruning is a method to remove connections that are not important. It was leveraged this idea and used it to prune the fault MACs causing the accuracy degradation [44]. Using standard post-fabrication analysis, they find the location of the faulty MACs, and with some additional bypass circuity, they can bypass the faulty MACs. The area overhead of the new systolic array architecture due to the new bypass path is about 9 %.

FAP+T: With FAP, it is only possible to bypass the faulty MACs. An additional re-training step is added to recover the accuracy loss because of the missing MAC units. In this step, the model learns to adapt to the change caused due to missing MAC units and tries to attain its baseline classification accuracy. Authors have claimed that, even with 50 % faulty MAC units, FAP+T can provide close to baseline accuracy. Post-fabrication analysis of every chip can be different; therefore, FAP and FAP+T need to be performed for all the chips with manufacturing defects.

5.5. Selective Hardening

Hardware redundancy-based methods (i.e., DMR and TMR) generally involve full hardware replication. Triple Modular Redundancy (TMR) has been used in the industry for many decades. Triplication of a design does provide the required resiliency to the safety-critical system but at the cost of increased power consumption and a considerable area overhead of 200 %. There are different attractive alternatives to full TMR for a wide variety of safety-critical applications. Typically, not all safety-critical applications focus on very high resiliency requirements. Further, not all the components in the design are essential. We can focus on hardening only the essential parts of the design, in other words, only triplicating the sensitive part of the design. This method is commonly known as Selective Hardening. Selective hardening can be applied on various abstraction levels in the design. In neural networks, it can be applied in the layer-level [20, 26], neurons level (Fully Connected layers) or channel-level (CNN Layers) [5] and also at the Processing Element (PEs) level.

Authors [5, 26, 1] have applied selective hardening techniques in the CNNs and have made a similar argument that triplicating the whole DNN model would cost a 200 % increase in the area overhead. They have proposed Selective layer and Channel Triplication, respectively, in a two-step process. a) Identification of the vulnerable CNN channels, which causes a decrease in accuracy in the presence of Faults b) triplication of the identified CNN layers/channels. They have used different network architectures of various sizes for their analysis. In [26] was performed fault injection using the FPGA accelerated fault injection and neutron flux radiation setup. Selective layer hardening resulted in the masking of 40 % faults with 8 % additional area overhead. While in the study [5], they can reduce the area overhead from 200 % to 173 % for a worst-case accuracy drop of 0.5 %. For a worst-case accuracy drop of 1 % and 2 %, they have reported a 200 % to 129.7 % and 49.87 % reduction in area overhead, respectively. This also validates the argument that for more strict accuracy requirements, more hardware area will be needed and vice versa. Hence, depending upon the application requirements, selective hardening of the design can be performed.

On hardware, neural networks are mapped to multiple processing elements. These processing elements perform multiply and accumulate operations. Based on the hardware architecture of the DNN, a single neuron, singlechannel, single layer, or multiple layers can be mapped to these processing elements. A fully parallelized architecture of a neural network in which each neuron is mapped to one PE is very unlikely for bigger networks. Therefore, in most cases, these PEs are being shared between different neurons, channels, or layers. Therefore, even one faulty PE can cause a huge accuracy degradation. In [5] was investigated that the more PEs are being shared, the higher it will lead to a decrease in accuracy in case of a fault. Therefore, it would be interesting to explore selective hardening at the PE level.

5.6. Ensemble Learning Based Robustness

Ensemble learning methods were initially proposed to reduce overfitting or better generalize the results compared to the results from a single model instance. Multiple ML models are trained on the same dataset during the training phase. The output of each model is processed to estimate the best outcome during the inference phase. One way of selecting the best outcome is to take an average of predictions of individual models. In [33] was used an ensemble learning-based approach to incre-

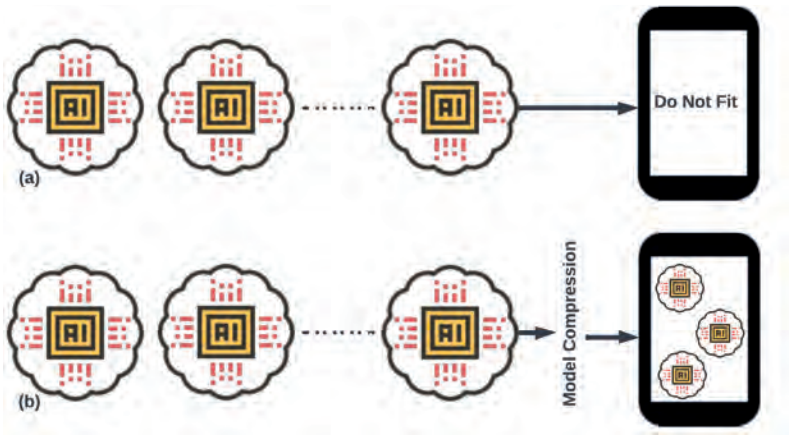


Fig. 9. (a) Traditional ensemble of AI models (b) Ensemble of compressed AI models
Rys. 9. (a) Tradycyjny zestaw modeli AI (b) Zestaw skompresowanych modeli AI

ase the robustness of the CNN model. The larger the number of CNN models in an ensemble, the greater will be the hardware resource utilization and power consumption. The author has used quantization and pruning to compress the model 1/N times its initial size. The hardware resources they save during the model compression are utilized to create an ensemble of N CNN models (Fig. 9). In contrast to traditional ensemble methods, their proposed approach of compressed ensembles can be deployed on constrained devices with no energy or memory overhead. The output of the CNN ensemble is averaged to compute the final predictions. Their target hardware is an edge device comprising of processor connected to main memory via a 32-bit bus. They have considered faults in memory that happen due to sub-nominal operating conditions.

5.7. Knowledge Distillation Based Redundancy

In [25] was developed an approach that uses knowledge distillation-based redundancy to detect faults. Knowledge distillation, proposed in [17], is a training-based solution for reducing the model's size, in which the knowledge from the teacher model is transferred to a simpler student model. In this way, a smaller student model can approximate the results of the bigger teacher model. The teacher model is generally a complex ML/DL model or an ensemble of models. In contrast, a student model is usually a single smaller model that is much more straightforward to deploy without substantial loss in performance. Their approach makes use of two DNN models, i.e., task DNN (teacher model) and checker DNN (student model). Instead of using an expensive DMRbased solution, have reduced the size of the checker DNN model by utilizing knowledge distillation and architecture compression approach [25]. Both of these models process each input sample (see Fig. 10). A comparator block compares the outcome of both models. If the results are consistent, they are considered for further processing; otherwise, re-computation on the task DNN is performer for potential recovery from the fault. They have performer the experiments from the security perspective and considered the fault model in which the attacker is trying to compromise the accuracy of a DNN system by maliciously injecting faults. Experimental results show that at the cost of 10 % overhead, their approach can reduce 90 % of the risks. This approach can be considered and studied from the reliability perspective as well.

5.8. ABFT: Algorithm-based Fault Tolerance

Matrix multiplications are the fundamental arithmetic operation in neural networks. In order to make this matrix multiplications fault-tolerant, Algorithm-based fault tolerance (ABFT) was proposed in [19]. ABFT cannot only detect the errors but also can correct the errors. ABFT is a very attractive solution to make the neural network fault-tolerant, and it costs low area overhead as compared to traditional TMR methods. The core idea behind ABFT can be thought of as an extension of ECC to numeric structures like vectors and matrices. In [45, 37] was applied the ABFT approach to CNN models. Zhao et al. (2020) have considered some of the widely used CNN models, i.e., AlexNet, VGG-19, ResNet-18, and YOLOv2, and demonstrated the results as per runtime overhead metric. Their ABFT approach can handle soft errors with a very small runtime overhead of 4 % to 8 %. In [37] was implemented CNNs on three GPU architectures, i.e., K40, Tegra X1, Titan X. Their ABFT approach is able to detect and correct 50 % to 60 % of radiation-induced corruptions.

Similar to ABFT, was proposed a lowoverhead error detection technique for matrix multiplications [28]. I.e., Light ABFT.

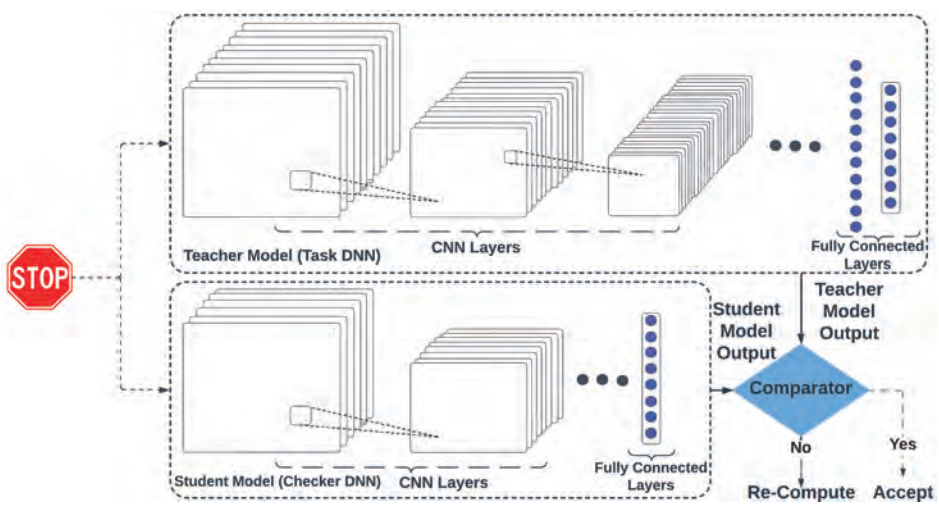


Fig. 10. Knowledge distillation based redundancy
Rys. 10. Redundancja oparta na destylacji wiedzy

Unlike ABFT, which can perform error detection and correction, the light ABFT approach can only detect errors. Author have targeted FPGAs, and he argues that as soon as the error is detected with Light ABFT, it can be corrected using fast partial reconfiguration of the FPGA bitstream.

Authors of [31] have made use of the linearity property of the fully connected layers and convolutions layers and have proposed a low overheard error detection method, called as Sanity-Check. The ABFT also inspires this approach, and in this approach, with the addition of two neurons (namely sanity-neuron and check-neuron), they can detect whether the layer's output is erroneous. The sanity-neuron acts as an additive inverse of the rest of the neuron in the layer, while the check-neuron sums the output to confirm if the result is zero or non-zero. A similar approach is applied in the case of convolution layers.

5.9. Arithmetic Error Codes

Arithmetic error codes, which comes under the umbrella of ECCs, are an exciting way to detect and correct errors, as they are conserved during most arithmetic operation [32]. They have been used in various safety-critical applications to increase the reliability of the systems. In [10] authors have used a specific class of arithmetic codes, known as AN-Codes, in state-of-the-art DNN accelerators. They have exhibited that they can achieve 99 % fault coverage with a 5-bit arithmetic code with minimal area and power overhead.

5.10. Inter Frame Spatio-Temporal Correlation

In [4] was proposed a very different and unique approach to detect errors in CNNs. The general functionality of CNNs is that it takes an image as input and output the predictions. Each image is a frame, and many frames are captured and processed in one second. CNNs treat each frame independently and predict the output. Most of the time, these input frames are correlated, and hence the output predictions are also similar. Therefore, not only do the input frames correlate with each other but also the output predictions. They use both the input and output correlation information to detect errors in a frame as it is processed. If there is a difference in the correlation of output predictions, then there are two possibilities. a) The input frame is also different. In this case, the change in the output predictions is justified by the change in frames. b) Erroneous output prediction. Subsequent frames are identical, and hence the output predictions should also be identical. They have performed error analysis on two CNNS, i.e., YOLO and Faster R-CNN trained on Caltech Pedestrian Dataset [3]. They are able to detect 80 % of errors while keeping the area overhead low.

6. Conclusion

Safety-critical applications require fault-free execution of critical tasks. The increased use of DNNs in safety-critical applications demands a thorough understanding of targeted hardware (ASICs, FPGAs) and DNNs' characteristics. Thus, this survey paper has discussed and differentiated between ASICs and FPGA fault models. The essential concept of three generations of neural networks is explained. We extended this discussion and examined factors that impact the resiliency of neural networks and several state-of-the-art fault mitigation methodologies.

Improving the reliability of the DL accelerators is like "chasing a moving target". The design of an efficient faulttolerant AI accelerator will involve the combined effort of researchers in both the AI and reliability domains. Previously, AI models have been treated as black boxes. Now, the increased use of AI in safety-critical applications, i.e., Medical, Automotive, Industries, Defense, Space, etc., has led researchers to work on "Explainable AI". The insights obtained from these studies will help design efficient faulttolerant AI accelerators.

Acknowledgements

This work was supported by the European Regional Development Fund within the BB-PL INTERREG V A 2014-2020 Program, "reducing barriers using the common strengths", project SpaceRegion, grant number 85038043.

References

1. Adam K., Izeldin I.M., Ibrahim Y., *A selective mitigation technique of soft errors for DNN models used in healthcare applications: DenseNet201 case study.* „IEEE Access", Vol. 9, 2021, 65803–65823. DOI: 10.1109/ACCESS.2021.3076716.
2. Chen Z., Li G., Pattabiraman K., *A low-cost fault corrector for deep neural networks through range restriction.* [In:] 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2021, 1–13, DOI: 10.1109/DSN48987.2021.00018.
3. Dollar P., Wojek C., Schiele B., Perona P., *Pedestrian detection: An evaluation of the state of the art.* „IEEE Transactions on Pattern Analysis and Machine Intelligence", Vol. 34, No. 4, 2012, 743–761, DOI: 10.1109/TPAMI.2011.155.
4. Draghetti L.K., Santos F.F.D., Carro L., Rech P., *Detecting Errors in Convolutional Neural Networks Using Inter Frame Spatio-Temporal Correlation.* IEEE 25th International Symposium on On-Line Testing and Robust System Design, IOLTS 2019, 310–315, DOI: 10.1109/IOLTS.2019.8854431.
5. Gambardella G., Kappauf J., Blott M., Doehring C., Kumm M., Zipf P., Vissers K., *Efficient error-tolerant quantized neural network accelerators.* IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, DFT 2019. DOI: 10.1109/DFT.2019.8875314.
6. Gao Z., Wei X., Zhang H., Li W., Ge G., Wang Y., Reviriego P., *Reliability evaluation of pruned neural networks against errors on parameters.* IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, DFT 2020, DOI: 10.1109/DFT50435.2020.9250812.
7. Ghavami B., Sadati M., Fang Z., Shannon L., *FitAct: Error resilient deep neural networks via fine-grained post-trainable activation functions.* Design, Automation Test in Europe Conference Exhibition (DATE '22), 1239–1244, DOI: 10.23919/DATE54114.2022.9774635.
8. Gill B.S. *Design and Analysis Methodologies to Reduce Soft Errors in Nanometer VLSI Circuits.* Ph.D. thesis, 2006, Department of Electrical Engineering and Computer Science CASE WESTERN RESERVE UNIVERSITY.
9. Goldstein B.F., *Reliability evaluation of compressed deep learning models.* 2020 IEEE 11th Latin American Symposium on Circuits Systems (LASCAS), 1–5, DOI:10.1109/LASCAS45839.2020.9069026.
10. Goldstein B.F., Ferreira V.C., Srinivasan S., Das D., Nery A.S., Kundu S., Franca F.M.G., *A lightweight error--resiliency mechanism for deep neural networks.* 22nd International Symposium on Quality Electronic Design (ISQED), 2021, 311–316. DOI:10.1109/ISQED51717.2021.9424287.
11. Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde--Farley D., Ozair S., Courville A., Bengio Y., *Generative adversarial nets.* (Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger eds.), „Advances in Neural Information Processing Systems", Vol. 27, 2014, 2672–2680, DOI: 10.5555/2969033.2969125.
12. Haenlein M., Kaplan A., *A brief history of artificial intelligence: On the past, present, and future of artificial intelligence.* „California Management Review", Vol. 61, No. 4, 2019, 5–14. DOI: 10.1177/0008125619864925.

13. Hass K.J., *Probabilistic estimates of upset caused by single event transients*, 1999.
14. Hayes J., *Fault modeling for digital mos integrated circuits*. „IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems", Vol. 3, No. 3, 1984, 200–208, DOI: 10.1109/TCAD.1984.1270076.
15. He K., Zhang X., Ren S., Sun J., *Deep residual learning for image recognition*. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778, DOI: 10.1109/CVPR.2016.90.
16. Hinton G.E., Krizhevsky A., Wang S.D., *Transforming auto-encoders*. [In:] T. Honkela, W. Duch, M. Girolami, S. Kaski, (eds.), Artificial Neural Networks and Machine Learning – ICANN 2011, 44–51. DOI: 10.1007/978-3-642-21735-7_6.
17. Hinton G.E., Vinyals O., Dean J. *Distilling the knowledge in a neural network*. ArXiv, 2015, DOI: 10.48550/arXiv.1503.02531.
18. Hoang L.H., Hanif M.A., Shafique M., *FT-ClipAct: Resilience analysis of deep neural networks and improving their fault tolerance using clipped activation*. 2020 Design, Automation Test in Europe Conference Exhibition (DATE), 1241–1246. DOI:10.23919/DATE48585.2020.9116571.
19. Huang K.H., Abraham, J.A. (1984). Algorithm-based fault tolerance for matrix operations. IEEE Transactions on Computers, C-33(6), 518–528. DOI:10.1109/TC.1984.1676475.
20. Ibrahim Y., Wang H., Bai M., Liu Z., Wang J., Yang Z., Chen Z., *Soft Error Resilience of Deep Residual Networks for Object Recognition*. „IEEE Access", Vol. 8, 2020, 19490–19503, DOI: 10.1109/ACCESS.2020.2968129.
21. Kastensmidt F.L., Carro L., Reis R., *Fault-tolerance techniques for SRAM-based FPGAs*, 2006, Springer, DOI: 10.1007/978-0-387-31069-5.
22. Krizhevsky A., Sutskever I., Hinton G.E., *Imagenet classification with deep convolutional neural networks*. [In:] F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger (eds.), Advances in Neural Information Processing Systems, Vol. 25, 2012, Curran Associates, Inc.
23. LeCun Y., Bengio Y., Hinton G., *Deep learning*. „Nature", Vol. 521(7553), 2015, 436–444. DOI: 10.1038/nature14539.
24. 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, 2017, 1–12, DOI: 10.1145/3126908.3126964.
25. Li Y., Li M., Luo B., Tian Y., Xu Q., *DeepDyve: Dynamic verification for deep neural networks*. [In:] Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, CCS '20, 101–112. Association for Computing Machinery, New York, NY, USA, DOI: 10.1145/3372297.3423338.
26. Libano F., Wilson B., Anderson J., Wirthlin M.J., Cazzaniga C., Frost C., Rech P., *Selective hardening for neural networks in FPGAs*. „IEEE Transactions on Nuclear Science", Vol. 66, No. 1, 2019, 216–222. DOI: 10.1109/TNS.2018.2884460.
27. 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 Transactions on Nuclear Science", Vol. 67, No. 7, 2020, 1478–1484, DOI: 10.1109/TNS.2020.2983662.
28. Libano F., *Analyzing and Improving the Reliability of Matrix Multiplication and Neural Networks on FPGAs*. Ph.D. thesis, 2021, Arizona State University.
29. Lyashenko V., *Basic guide to spiking neural networks for deep learning*, 2020, https://cnvrg.io/spiking-neural-networks/.
30. Mittal S., Vetter J.S., *A survey of techniques for modeling and improving reliability of computing systems*. „IEEE Transactions on Parallel and Distributed Systems", Vol. 27, No. 4, 2016, 1226–1238. DOI: 10.1109/TPDS.2015.2426179.
31. Ozen E., Orailoglu A., *Sanity-check: Boosting the reliability of safety-critical deep neural network applications*. In 2019 IEEE 28th Asian Test Symposium (ATS), 7–75. DOI: 10.1109/ATS47505.2019.000-8.
32. Parhami, Avizienis, *Detection of storage errors in mass memories using low-cost arithmetic error codes*. „IEEE Transactions on Computers", Vol. C-27, No. 4, 1978, 302–308. DOI: 10.1109/TC.1978.1675102.
33. Ponzina F., Peón-Quirós M., Burg A., Atienza D., *E²CNNs: Ensembles of convolutional neural networks to improve robustness against memory*, errors in edge-computing devices. „IEEE Transactions on Computers", Vol. 70, No. 8, 2021, 1199–1212. DOI: 10.1109/TC.2021.3061086.
34. Ribes S., Malek A., Trancoso P., Sourdis I., *Reliability analysis of compressed CNNs*. 2021.
35. Russakovsky O., Deng J., Su H., Krause J., Satheesh S., Ma S., Huang Z., Karpathy A., Khosla A., Bernstein M., Berg A.C., Fei-Fei, L., *ImageNet Large Scale Visual Recognition Challenge*. „International Journal of Computer Vision", Vol. 115, No. 3, 2015, 211–252, DOI: 10.1007/s11263-015-0816-y.
36. Sabbagh M., *Evaluating fault resiliency of compressed deep neural networks*. 2019 IEEE International Conference on Embedded Software and Systems (ICESS), 2019, 1–7. DOI:10.1109/ICESS.2019.8782505.
37. Santos F.F.D., *Evaluation and Mitigation of Soft-Errors in Neural Network-Based Object Detection in Three GPU Architectures*. 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops, DSN-W 2017, 169–176. DOI:10.1109/DSN-W.2017.47.
38. Simonyan K., Zisserman A., *Very deep convolutional networks for large-scale image recognition*, 2015, DOI: 10.48550/arXiv.1409.1556.
39. Syed R.T., Ulbricht M., Piotrowski K., Krstic, M., *Fault resilience analysis of quantized deep neural networks*. IEEE 32nd International Conference on Microelectronics (MIEL), 2021, 275–279. DOI: 10.1109/MIEL52794.2021.9569094.
40. Sze V., Chen Y.H., Yang T.J., Emer J.S., *Efficient processing of deep neural networks: A tutorial and survey*. Proceedings of the IEEE, Vol. 105, No. 12, 2017, 2295–2329, DOI: .1109/JPROC.2017.2761740.
41. Werner S., Navaridas J., Luján M., *A survey on design approaches to circumvent permanent faults in networkson-chip*. „ACM Computing Surveys", Vol. 48, No. 4, 2016, DOI:10.1145/2886781.
42. Yi C.H., Kwon K.H., Jeon J., *Method of improved hardware redundancy for automotive system*. 14th International Symposium on Communications and Information Technologies (ISCIT), 2014, 204–207, DOI: 10.1109/ISCIT.2014.7011901.
43. Zahid U., Gambardella G., Fraser N.J., Blott M., Vissers K., *FAT: Training neural networks for reliable inference under hardware faults*. 2020 IEEE International Test Conference (ITC), 2020, 1–10. DOI: 10.1109/ITC44778.2020.9325249.
44. Zhang J., Gu T., Basu K., Garg S., *Analyzing and mitigating the impact of permanent faults on a systolic array based neural network accelerator*. Proceedings of the IEEE VLSI Test Symposium, 2018, 1–6, DOI: 10.1109/VTS.2018.8368656.
45. Zhao K., Di S., Li S., Liang X., Zhai Y., Chen J., Ouyang K., Cappello F., Chen Z., *Algorithm-Based Fault Tolerance for Convolutional Neural Networks*. 2020, 1–13, DOI: 10.1109/TPDS.2020.3043449.

Przegląd metod zapewniających odporność na błędy dla głębokich sieci neuronowych

Streszczenie: Znaczący rozwój sztucznej inteligencji (SI) wpływa na wiele otaczających nas aplikacji, do tego stopnia, że SI jest obecnie coraz częściej wykorzystywana w aplikacjach o krytycznym znaczeniu dla bezpieczeństwa. Sztuczna inteligencja na brzegu sieci (Edge) jest rzeczywistością, co oznacza wykonywanie obliczeń na danych bliżej źródła danych, w przeciwieństwie do wykonywania ich w chmurze. Aplikacje o krytycznym znaczeniu dla bezpieczeństwa mają wysokie wymagania dotyczące niezawodności; dlatego ważne jest, aby modele SI działające na brzegu sieci (tj. sprzęt) spełniały wymagane standardy bezpieczeństwa. Z rozległej dziedziny sztucznej inteligencji, głębokie sieci neuronowe (DNN) są centralnym punktem tego badania, ponieważ nadal przynoszą znakomite wyniki w różnych zastosowaniach, tj. medycznych, motoryzacyjnych, lotniczych, obronnych itp. Tradycyjne techniki niezawodności implementacji w przypadku DNN nie zawsze są praktyczne, ponieważ nie wykorzystują unikalnych cech DNN. Co więcej, istotne jest również zrozumienie docelowego sprzętu brzegowego, ponieważ wpływ usterek może być różny w układach ASIC i FPGA. Dlatego też w niniejszym przeglądzie najpierw zbadaliśmy wpływ usterek w układach ASIC i FPGA, a następnie staramy się zapewnić wgląd w ostatnie postępy poczynione w kierunku DNN odpornych na błędy. Omówiliśmy kilka czynników, które mogą wpływać na niezawodność sieci DNN. Ponadto rozszerzyliśmy tę dyskusję, aby rzucić światło na wiele najnowocześniejszych technik ograniczania błędów w sieciach DNN.

Słowa kluczowe: odporność na błędy, niezawodność, układy FPGA, układy ASIC, sieci neuronowe

Rizwan Tariq Syed, MSc, Eng.
syed@ihp-microelectronics.com
ORCID: 0000-0001-9232-734X

He completed his Bachelor's in Industrial Electronics from the Institute of Industrial Electronics Engineering (affiliated with NED UET), Karachi, Pakistan. He has further received his Master's Degree in Masters of Science in Communication Engineering (MSCE) from Technical University Munich, Germany. Since 2017, he has been with IHP, Frankfurt (Oder), Germany, where he works as a Research Scientist in the ‚System Architectures' department. For the last few years, his work has been mainly focused on fault-tolerant and reconfigurable AI Accelerators on FPGAs. He has been involved in many research and development projects at IHP (EMPHASE, Space Region, Open6G Hub).

Markus Ulbricht, PhD, Eng.
ulbricht@ihp-microelectronics.com
ORCID: ORCID: 0000-0001-9230-640X

He received his doctorate degree from Brandenburg University Cottbus-Senftenberg in 2014 on the topic of „Systematic lifetime-optimization of highly integrated systems on the basis of nano-structures by means of stress optimization and self-repair". In the following two years, he collected thorough experience as a test engineer at Intel Communications GmbH in Munich. To have a stronger focus on his scientific career, he transferred to IHP in 2016, where his first projects involved backend HDL design and the design and implementation of a fault-tolerant radar platform for distance measurements for automated driving. As of 2020, he is leading the fault-tolerant computing group with a strong focus on sensory platforms, open-source hardware, reliable computing, and neuromorphic processing systems.

Krzysztof Piotrowski, PhD, Eng.
piotrowski@ihp-microelectronics.com
ORCID: 0000-0002-7231-6704

He received his Master degree in Computer Science from the University of Zielona Gora in Poland in 2004. Since then, he is with the IHP in Frankfurt (Oder) in Germany, where he currently leading the Sensor Networks and Middleware Platforms (SMP) group in the Wireless Systems department. In 2011 he received his PhD in Computer Science from the Brandenburg University of Technology Cottbus-Senftenberg, in Germany. Since 2019 he is leading the joint lab on Distributed Measurement Systems and Wireless Sensor Networks with the University of Zielona Gora. His research interests include distributed data handling and storage with the focus on privacy and security issues and resource-constrained devices - wireless sensor networks. He was coordinating the EU FP7 project e-balance and the INTERREG project SmartGrid Plattform and is responsible for SmartGrid and SmartCity related research within the SMP group at the IHP. He is also coordinating the INTERREG project SmartRiver with the focus on wireless sensor networks and distributed data storage and processing. He published more than 80 refereed technical articles and is/was active in several European and national (German) research projects.

Prof. Milos Krstic, PhD, Eng.
krstic@ihp-microelectronics.com
ORCID: 0000-0003-0267-0203

He received the Dr-Ing. degree in electronics from Brandenburg University of Technology, Cottbus, Germany in 2006. Since 2001 he has been with IHP, Frankfurt (Oder), Germany, where he leads the department System Architectures. From 2016 he is also professor for "Design and Test Methodology" at the University of Potsdam. For the last few years, his work was mainly focused on fault tolerant architectures and design methodologies for digital systems integration. Prof. Krstic has been managing many international and national R & D projects. He is also leading and coordinating space activities at IHP. He has published more than 250 journal and conference papers, and registered 12 patents.