Robust Computing for Machine Learning-Based Systems

Muhammad Abdullah Hanif, Faiq Khalid, Rachmad Vidya Wicaksana Putra, Mohammad Taghi Teimoori, Florian Kriebel, Jeff (Jun) Zhang, Kang Liu, Semeen Rehman, Theocharis Theocharides, Alessandro Artusi, Siddharth Garg, and Muhammad Shafique

1 Introduction

Machine learning (ML) has emerged as the principal tool for performing complex tasks which are impractical (if not impossible) to code by humans. ML techniques provide machines the capability to learn from experience and thereby learn to perform complex tasks without much (if any) human intervention. Over the past decades, many ML algorithms have been proposed. However, Deep Learning (DL), using Deep Neural Networks (DNNs), has shown state-of-the-art accuracy, even surpassing human-level accuracy in some cases, for many applications [31]. These applications include, but are not limited to, object detection and localization, speech recognition, language translation, and video processing [31].

The state-of-the-art performance of the DL-based methods has also led to the use of DNNs in complex safety-critical applications, for example, autonomous driving [11] and smart healthcare [10]. DNNs are intrinsically computationally

M. A. Hanif (✉) · F. Khalid · R. V. W. Putra · M. T. Teimoori · F. Kriebel · S. Rehman
M. Shafique
Technische Universität Wien (TU Wien), Vienna, Austria
e-mail: muhammad.hanif@tuwien.ac.at; faiq.khalid@tuwien.ac.at; rachmad.putra@tuwien.ac.at; florian.kriebel@tuwien.ac.at; semeen.rehman@tuwien.ac.at; muhammad.shafique@tuwien.ac.at

J. Zhang · K. Liu · S. Garg
New York University, New York, NY, USA
e-mail: jeffjunzhang@nyu.edu; kang.liu@nyu.edu; sg175@nyu.edu

T. Theocharides
University of Cyprus, Nicosia, Cyprus
e-mail: ttheocharides@ucy.ac.cy

A. Artusi
University of Cyprus, Nicosia, Cyprus

MRG DeepCamera RISE, Nicosia, Cyprus

J. Henkel, N. Dutt (eds.), *Dependable Embedded Systems*, Embedded Systems,
https://doi.org/10.1007/978-3-030-52017-5_20

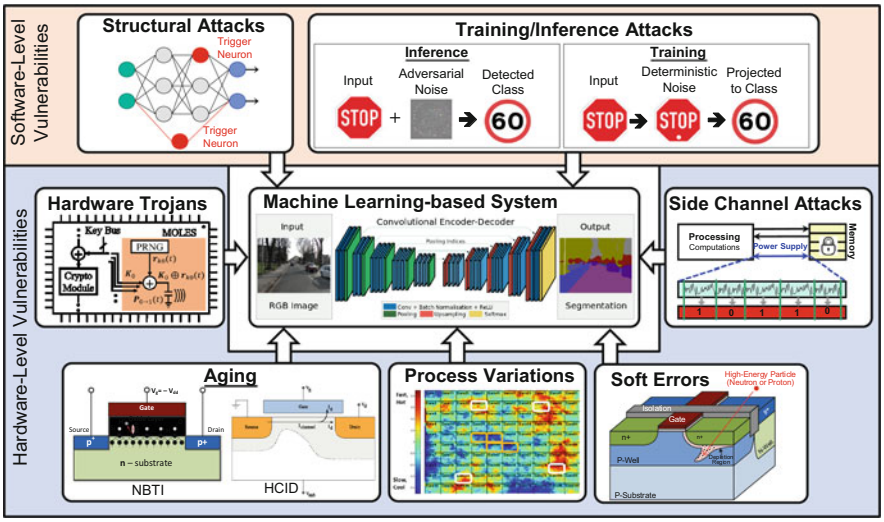

Fig. 1 Overview of different reliability and security vulnerabilities to machine learning-based systems. (Picture sources: [47, 49])

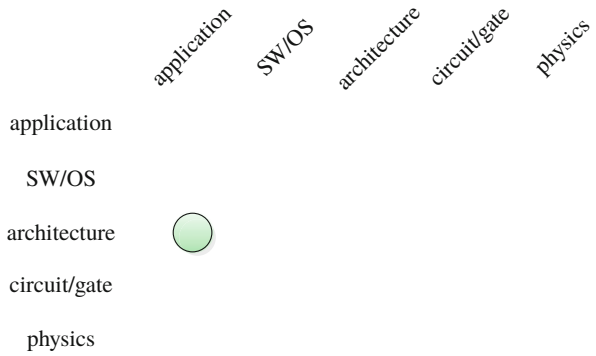

Fig. 2 Main abstraction layers of embedded systems and this chapter's major (green, solid) contributions

intensive and also require high memory resources [53]. Current research mainly focuses on the development of less computationally intensive and resource-efficient DNNs that can offer high accuracy, and energy and performance efficient DNN accelerators for ML-based applications [1, 18, 23, 29, 34, 36, 37, 44, 53]. *However, when considered for safety-critical applications, the robustness of these DNN-based systems to different reliability and security vulnerabilities also becomes one of the foremost objectives.* An overview of different types of vulnerabilities in ML-based systems is shown in Fig. 1, which are discussed from the architectural- and application-layer perspective in this chapter. Figure 2 shows the abstraction layers in the context of the SPP 1500 covered in this chapter.

Reliability Threats: In hardware design, reliability is the ability of the hardware to perform as intended for a specified duration, i.e., the lifetime of the hardware. There are a number of hardware related vulnerabilities that can disrupt the functionality of a digital system in its lifetime.

1. **Soft Errors** are transient faults caused by high energy particle strikes. These faults surface at hardware-layer as bit-flips and can propagate to the application layer resulting in incorrect output.
2. **Aging** is the gradual degradation of the hardware due to different physical phenomena like Hot carrier Injection (HCI), Negative-Bias Temperature Instability (NBTI), and Electromigration (EM). It leads to timing errors and eventually can also lead to permanent faults [56].
3. **Process variations** are the imperfections caused by the variations in the fabrication process of the chips. This can lead to variations in the timing and leakage power characteristics within a chip as well as across different chips [45].

Apart from the above-listed vulnerabilities, environmental conditions can also affect the reliability of a system. Such factors include temperature, altitude, high electric fields, etc.

A number of techniques have been proposed for improving the resilience of the systems against the reliability threats. However, most of these mitigation techniques are based on redundancy, for example, DMR: dual modular redundancy [58] and TMR: triple modular redundancy [35]. The redundancy based approaches, although considered to be very effective for other application domains [19], are highly inefficient for DNN-based systems because of the compute intensive nature of the DNNs [48], and may incur significant area, power/energy, and performance overheads. *Hence, a completely new set of resource-efficient reliability mechanisms is required for robust machine learning systems.* A list of techniques proposed for improving the reliability of DNN-based systems, which are later discussed in the following sections of the chapter, are mentioned in Fig. 3.

Security Threats: In system design, security is defined as the property of a system to ensure the confidentiality, integrity, and availability of the hardware and the data while performing the assigned tasks. There are several security vulnerabilities that can be exploited to perform security attacks.

1. **Data Manipulation:** The input data or data during inter-/intra-module communication in a system can be manipulated to perform several security attacks. For example, in DNNs, the training dataset and the inference data can be manipulated to perform misclassification or confidence reduction attacks [17, 24, 26, 27, 43, 51].
2. **Denial-of-Service:** A tiny piece of code/hardware or flooding the communication channels can be used to trigger the malfunctioning or failure of the system. For example, in DNNs, adding an extra neuron/set of neurons [17] or introducing the kill switch in DNN-based hardware can lead to system failure or malfunctioning, i.e., misclassification.

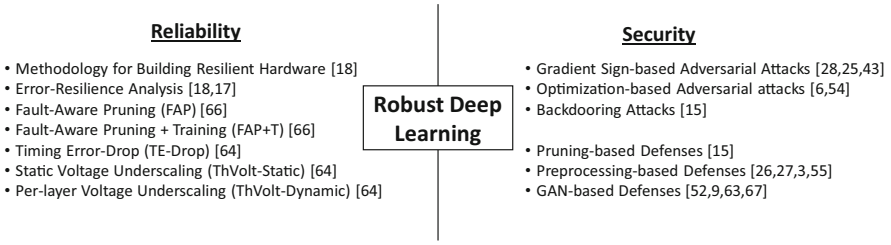

Fig. 3 Overview of the works discussed in this chapter for addressing reliability and security vulnerabilities of deep learning-based systems

3. **Data/IP Stealing:** The side-channel information (in hardware, power, timing, and loopholes or behavior leaking properties of the algorithms) can be exploited to steal the confidential information. For example, in DNNs, the gradient information can be used to steal trained model [50, 57, 60].

Several countermeasures have been developed to address these threats, but most of these defenses are either based on obfuscation or run-time monitoring [3, 22]. These techniques are very effective for traditional systems, however, DNN-based systems require different approaches because of their unique security vulnerabilities, i.e., training/inference data manipulation. Some of the techniques proposed for addressing the security of DNN-based systems are listed in Fig. 3 and are later discussed in the chapter.

In the following sections, we discuss:

1. A brief *overview of DNNs and the hardware accelerators* used for efficiently processing these networks.
2. In Sect. 3, we present our *methodology for building reliable systems* and discuss *techniques for mitigating permanent and timing errors*.
3. The *security vulnerabilities* in different types of DNNs are discussed in Sect. 4.
4. *Open challenges* and further *research opportunities* for building robust systems for ML-based safety-critical applications

2 Preliminaries

2.1 Deep Neural Networks

A neural network can be described as a network of interconnected neurons. *Neurons* are the fundamental computational units in a neural network where each neuron performs a weighted sum of inputs (dot-product operation), using the inputs and the weights associated with each input connection of the neuron. Each output is then (optionally) passed through an *activation function* which introduces non-linearity and thereby allows the network to learn complex classification boundaries.

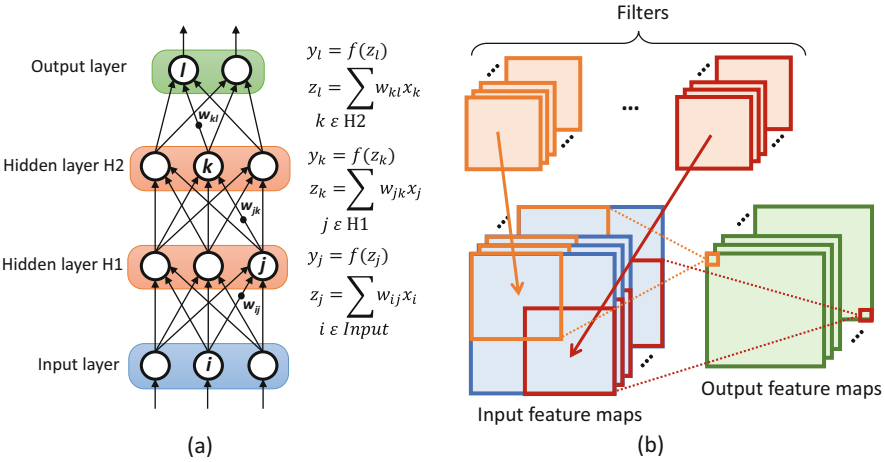

Fig. 4 Illustration of (**a**) a multi-layer perceptron and (**b**) a convolutional layer

In neural networks, neurons are arranged in the form of *layers*. There are several types of NNs, for instance, Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Multi-Layer Perceptrons (MLPs) [31]. Although the techniques discussed in the following sections are not limited to a specific type of NNs, in this chapter, we mainly focus on feed-forward neural networks (i.e., CNNs and MLPs) because of their widespread use in many artificial intelligence applications.

An MLP is a type of NN that is composed of multiple fully-connected layers. In a fully-connected layer, each neuron is connected to all the neurons in the neighboring layers. An example illustration of a three layer MLP is shown in Fig. 4a.

A CNN is a type of NN that is composed of several convolutional layers and the fully-connected layers. An example illustration of a *convolutional layer* is shown in Fig. 4b. The layer is composed of multiple filters which are convolved with the input feature maps to generate the output feature maps. The depth of the filters and the input feature maps is the same. Each filter results in one *output feature map* and, therefore, the number of output feature maps is equal to the number of filters in a convolutional layer. These input and output feature maps are also referred to as *activation maps*. A detailed description of CNNs can be found in [53].

2.2 *Hardware Accelerators for Deep Neural Networks*

To enable the use of DNNs in energy-/power-constraint scenarios as well as in high performance applications, several different hardware architectures for DNN acceleration have been proposed. While all the accelerators provide some unique

features and support some specific dataflows in a more efficient manner, *systolic array-based designs are considered among the promising ones* [18, 23, 37, 61].

A systolic array is a homogeneous network of processing elements (PEs), which are tightly coupled together. Each PE in the network receives data from its nearest neighbors, performs some function, and passes on the result and data to the neighboring PE/s. The systolic array-based architectures alleviate the memory bottleneck issue by *locally reusing the data*, without the need of expensive memory read and write operations. Moreover, the systolic arrays are intrinsically efficient at performing matrix multiplications, which is the core operation of neural networks. Therefore, many accelerators use these arrays at their core for accelerating the neural networks [18, 23, 37, 61]. The *Tensor Processing Unit (TPU)*, a DNN accelerator that is currently in use in the datacenters of Google, is a systolic array-based architecture that uses an array of 256 × 256 multiply-and-accumulate (MAC) units. The TPU provides 15 × −30× faster execution, and 30 × −80× more efficient (in terms of performance/Watt) performance than the K80 GPU and the Haswell CPU [23].

Figure 5 illustrates a design overview of an exemplar DNN accelerator which is based on the TPU architecture. The design is used as the basic architecture in the following section. The architecture is composed of a systolic array of MAC units, similar to that in the TPU. Prior to the computations, the weights are pre-loaded in the PEs from the weight memory in a manner that the weights from the same filter/neuron are loaded in the same column of the array. During processing, the

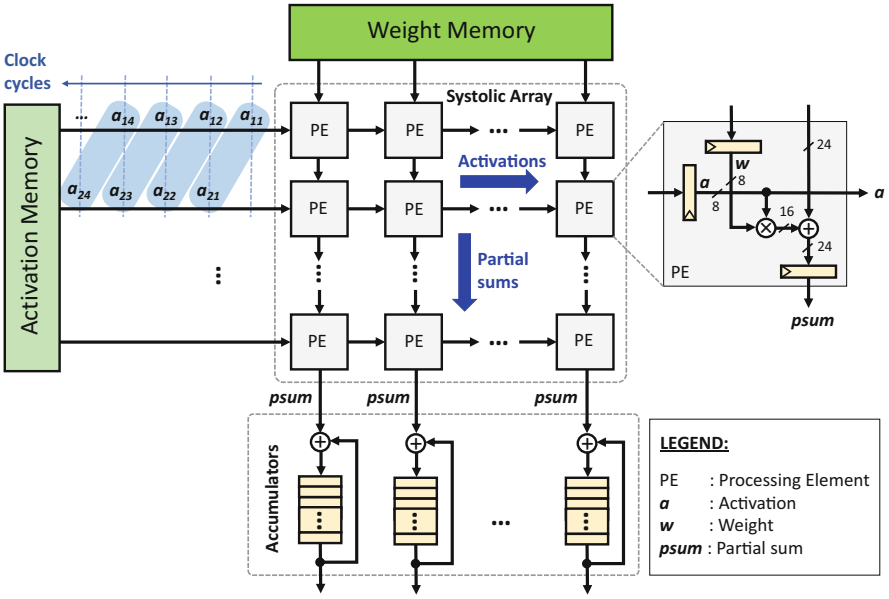

Fig. 5 A systolic array-based DNN accelerator architecture (adapted from [65])

weights are held *stationary* inside the PEs and the activations are streamed in from the activation memory. At each clock cycle, the activations are passed-on from left to right while the partial sums are moved downstream. The activations across rows are aligned such that the activations corresponding to a particular output reaches a particular PE at the same instance when its partial sum reaches that PE. In case, the size of a filter/neuron is larger than the number of rows in the array, each output computation related to the filter/neuron is divided into multiple portions and the accumulators at the bottom are used for temporarily holding the partial sums while rest of the corresponding partial sums are computed by the array. A more detailed explanation of the architecture can be found in [65].

3 Reliable Deep Learning

In this section, we present our methodology for building reliable hardware for DNN-based applications. We also highlight a few case studies, targeting different types of reliability threats, for building reliable yet efficient hardware for DNN-based applications.

3.1 Our Methodology for Designing Reliable DNN Systems

Figure 6 presents our design flow for developing reliable hardware for DNN-based applications [17]. The methodology is composed of two parts: (1) Design-time steps; and (2) Run-time steps.

The **design-time** steps focus on proposing a hardware architecture which is capable of mitigating different types of reliability faults that arise due to process variations and aging, as well as aggressive voltage scaling (i.e., permanent faults

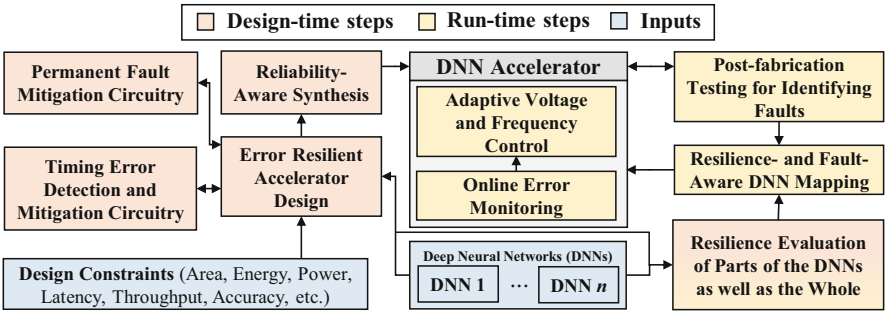

Fig. 6 Our methodology for designing reliable hardware for DNN-based applications (adapted from [17])

and timing errors). Provided a set of design constraints, representative DNN models, and resilience of the DNNs to different types of reliability threats and errors, a baseline hardware architecture is designed. We then reinforce it with different architectural enhancements for mitigating permanent faults (see Sect. 3.3) and handling timing errors (see Sect. 3.4). *The architectural enhancements are performed in a manner that they do not significantly affect the resource efficiency of the baseline architecture.* Once the architecture is finalized, the hardware is synthesized using reliability-aware synthesis techniques, for example, by using standard cells to selectively harden vulnerable nodes in the hardware [33], to harden the more vulnerable parts of the hardware design.

The **run-time** steps focus on proposing *mapping policies* for mapping DNN computations to the synthesized hardware. The mapping policies are decided based on the fault maps generated using post-fabrication and testing, and the error resilience of the DNNs. *Techniques like error injection can be used for the resilience analysis* [16, 46]. *Fault-aware training of DNNs* can also be used for designing/modifying network architecture/parameters (see Sect. 3.3). Moreover, adaptive voltage scaling can be employed for trading off reliability with energy efficiency based on the error resilience of the DNNs. If required, software-level redundancy can also be employed to further improve the reliability by performing the computations related to critical neurons/filters multiple times.

3.2 Resilience of DNNs to Reliability Threats

Neural Networks are assumed to be inherently error resilient [12]. However, different types of errors can have different impact on the output of a DNN. This section presents the accuracy analysis of DNNs in the presence of different types of reliability faults.

3.2.1 Resilience of DNNs to Permanent Faults

This section highlights the resilience of DNNs to permanent faults by empirically analyzing the effects of *stuck-at* permanent faults in the TPU-based accelerator (presented in Fig. 5) on the classification accuracy of different DNNs. The datasets (i.e., *MNIST* and *TIMIT*) and the corresponding network architectures used for this analysis are listed in Table 1. To study the resilience, the TPU with a systolic array of 256×256 MAC units is synthesized using 45 nm OSU PDK to generate a gate-level netlist and then stuck-at faults are inserted at internal nodes in the netlist. For this analysis, faults only in the data-path were considered as the faults in the memory components can be mitigated using Error Correction Codes (ECC) and faults in control-path can lead to undesirable results.

Figure 7a shows the impact of using a faulty TPU for two different classification tasks, i.e., *image classification using the MNIST dataset* and *speech recognition*

Table 1 Datasets and the corresponding 8-bit DNNs used for evaluation (adapted from [65])

Dataset	Network architecture	Accuracy(%)
MNIST [30]	Fully-connected (L1–L4): 784×256×256×256×10	98.15
TIMIT [4]	Fully-connected (L1–L4): 1845×2000×2000×2000×183	73.91
ImageNet [7]	Convolutional (L1–L2): (224, 224, 3)×(27, 27, 64)×(13, 13, 192) Convolutional (L3–L5): (13, 13, 384)×(13, 13, 256)×(6, 6, 256) Fully-connected (L6–L8): 4096×4096×1000	76.33 (Top-5)

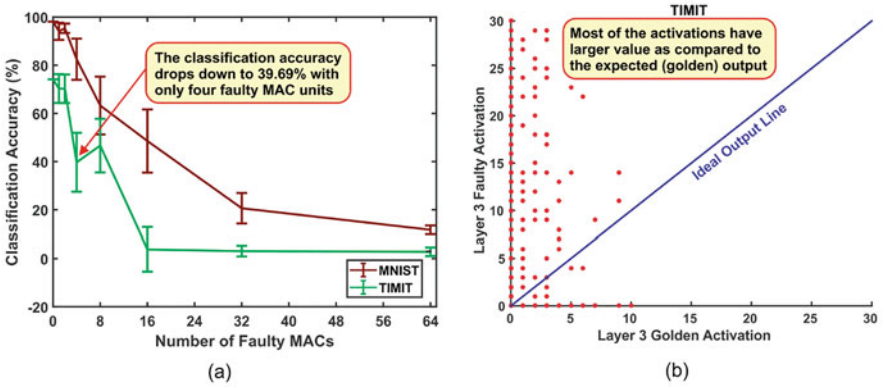

Fig. 7 Impact of stuck-at-faults in the baseline TPU-based architecture on DNN applications. (**a**) Classification accuracy drop due to stuck-at-fault MACs. (**b**) Impact of TPU stuck-at-faults on DNN applications (adapted from [66])

using the TIMIT dataset. It can be seen in the figure that the classification accuracy of both the tasks decreases significantly with the increase in the number of faulty PEs in the hardware. For example, the classification accuracy for the *TIMIT* dataset drops from 74.13 to 39.69% when only four (out of 256 × 256) MAC units are faulty and is almost 0% when the number of faulty MACs increases to 16 or more.

The reason for the significant drop in accuracy can be understood by comparing the golden (fault-free) output of the neurons of a particular layer with the outputs computed by the faulty TPU. Figure 7b shows that the computed output of the final layer of the network used for the *TIMIT* dataset in most of the cases has higher activation value as compared to the expected. This is mainly because of the fact that *stuck-at faults, in some of the cases, affect the higher order bits of the MACs output*. This highlights the need for permanent fault mitigation in the hardware to increase the yield as hardware with permanent faults cannot be used for ML-based applications, specifically for the safety-critical applications.

3.2.2 Resilience of DNNs to Timing Faults

Timing failures in high performance nanometer technology-based digital circuits are a major reliability concern and are caused by various mechanisms, e.g., power supply disturbance, crosstalk, process variations, as well as aging. Moreover, the operating conditions, which play a vital role in defining the performance and energy efficiency of the hardware, also have a significant impact on the frequency of the timing errors. Although it is assumed that the critical paths, which are more vulnerable to timing errors, are rarely exercised, the timing errors can significantly affect the functionality of an application. Here, we highlight this for DNN-based applications by analyzing the energy-quality trade-off achieved using voltage underscaling. We show the analysis for two widely accepted types of timing error mitigation techniques: (1) *timing error detection and recovery* (TED) [9]; and (2) *timing error propagation* (TEP) [41, 62]. The TED makes use of additional components (e.g., using Razor flip-flops [9]) for detecting timing errors, and recovers by reliably re-executing the function in case of errors. On the other hand, TEP allows errors to propagate through to the application layer in the hope that the application is error resilient.

For this analysis, the TPU-based hardware architecture discussed in Sect. 2.2 is considered. The architecture is assumed to be composed of a 256×256 MAC array. The terms *Local Timing Error* and *Global Timing Error* are used to characterize the resilience. The local timing error is used to denote the error in a single MAC unit. The global timing error defines the error in the complete systolic array. Figure 8b shows the impact on the classification accuracy for the *MNIST* dataset with voltage underscaling when the timing errors are allowed to propagate through to the application layer. It can be seen from the figure that as soon as the timing errors start occurring, i.e., below the voltage underscaling ratio of $r = 0.9$ (as shown in Fig. 8b), the classification accuracy of the DNN for TEP drops sharply.

As mentioned above, the TED-based approaches work on the principle of error detection and recovery. The recovery phase in TED defines its limitation for huge systolic array-based systems as, for synchronization of the data flow, the complete systolic array has to be stalled to recover the error in a single PE. This limitation of the TED-based approach can be highlighted using Fig. 8a which shows the impact of voltage underscaling on the overall energy consumption of the TPU-based hardware architecture for generating accurate outputs. It can be noted from the figure that *the overall energy consumption for a recovery based technique starts increasing as soon as errors start appearing*, which is the case for even the most naive type of error recovery mechanism, i.e., single cycle recovery.

3.2.3 Resilience of DNNs to Memory Faults

To illustrate the importance of memory faults, we presented an analysis in [17] where we injected random faults at bit-level in the weight memory (i.e., the memory storing the network parameters) and studied the impact of those faults on the

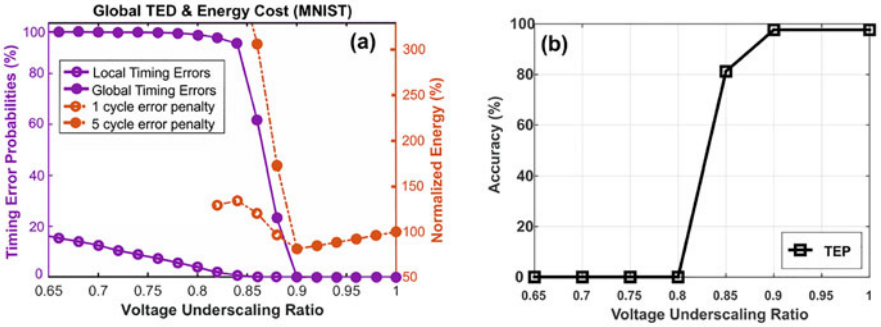

Fig. 8 (**a**) Timing error probabilities versus voltage underscaling ratio, and the corresponding energy cost for global TED. (**b**) DNN accuracy on the MNIST versus voltage underscaling for TEP. (Adapted from [65])

accuracy of a DNN. The analysis concluded that, for the higher significance bits of the weights, the accuracy of the DNNs drop sharply with the increase in error rate. We also studied the impact of different types of bit-flips, i.e., from 0 to 1 bit-flips and from 1 to 0 bit-flips, and found that the 0 to 1 bit-flips result in erroneous output while the 1 to 0 bit-flips do not impact the accuracy much. This is inline with the concept of dropout [20] and dropconnect [59] in the sense that in case of 1 to 0 bit-flips the erroneous output is leaned towards 0 value, whereas in case of 0 to 1 bit-flips the error can increase significantly if the bit-flip occurs in any of the higher significance bits. This analysis was performed on the AlexNet network using the ImageNet dataset. Similar, fault injection methods, e.g., [16] and [46], can also be used for analyzing the resilience of DNNs, as a whole as well as of individual layers/neurons of the networks.

3.3 Permanent Fault Mitigation

To mitigate permanent faults in the computing units of the hardware, two different methods have been proposed: (1) Fault-Aware Pruning (FAP); and (2) Fault-Aware Pruning + Training (FAP+T).

The **Fault-Aware Pruning (FAP)** works on the principle of pruning the weights (i.e., setting them to zero) that have to be mapped on faulty MAC units. *The principle is inline with the concepts of dropout [20] and dropconnect [59] which are commonly used for regularization and avoiding over-fitting*. For this work, the TPU architecture shown in Fig. 5 with static mapping policy is assumed. The static mapping policy means that each weight is mapped to a specific PE while multiple weights can be mapped to the same PE at different time instances. Moreover, it is also assumed that post-fabrication tests are performed on each TPU chip to extract the fault map which indicates the faulty PEs.

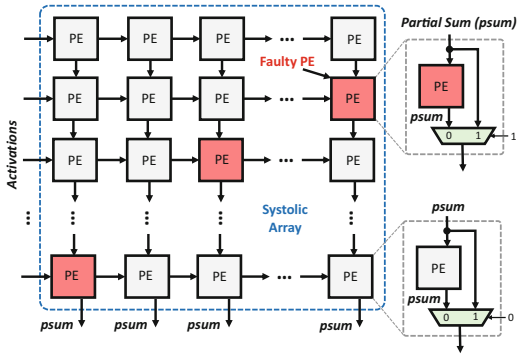

Fig. 9 Systolic array-based architecture for permanent fault mitigation (adapted from [66])

Figure 9 shows an implementation that can be used to realize the concept where a bypass path is provided for each MAC unit [66]. *The bypass path enables to skip the contribution of a specific partial sum in case the specific PE is faulty, which is equivalent to setting the weight to zero.* The area overhead of the modified design is only around 9% [66].

The **Fault-Aware Pruning + Training (FAP+T)** technique starts with the FAP approach, however, it additionally retrains the unpruned weights while forcing the pruned weights to zero to optimize the network parameters. One drawback of this approach is that the fault map of each chip can be different which means that a network has to be retained for each chip based on its own fault map.

Figure 10 shows the impact on the classification accuracy versus the percentage of faulty MAC units for three different classification problems mentioned in Table 1. The results show that both the techniques show significant resilience to the permanent faults. Moreover, the FAP+T technique outperforms FAP because of the involved optimization of the network parameters and allows the DNN-based system to run with negligible accuracy loss even when 50% of its MAC units are faulty. However, in cases where FAP+T is impractical FAP can also provide reasonable accuracy, specifically in cases where the number of faulty units is less.

3.4 Timing Fault Mitigation

As mentioned in Sect. 3.2.2, the conventional TED approaches have significant overheads when used for DNN accelerators. Here, we discuss the new architectural innovations proposed in Thundervolt [65] for mitigating timing errors in DNN accelerators in a performance efficient manner.

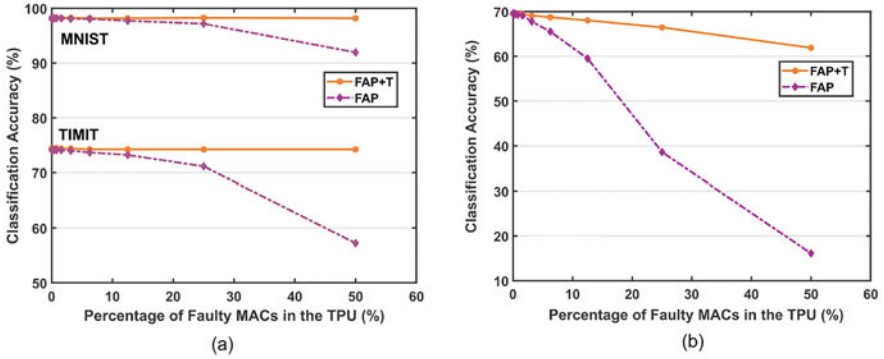

Fig. 10 Classification accuracy versus percentage of faulty MACs using FAP and FAP+T for the networks used corresponding to (**a**) MNIST and TIMIT; and (**b**) ImageNet datasets (adapted from [66])

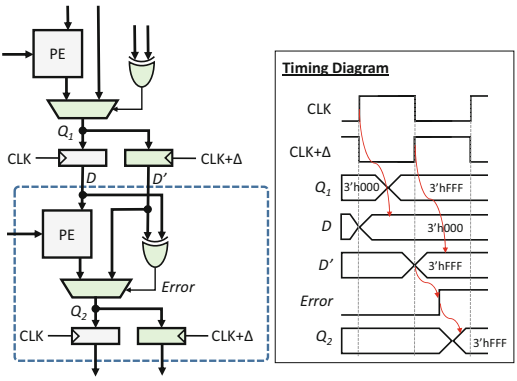

Fig. 11 A block-level diagram illustrating the architectural modifications for TE-Drop and the impact of timing errors on the computation of a neuron (adapted from [65])

3.4.1 TE-Drop

Thundervolt [65] proposed a novel technique to deal with timing errors in a systolic array-based DNN accelerator, i.e., TE-Drop. *TE-Drop utilizes the Razor flip-flops to detect timing errors, however, it does not re-execute erroneous MAC operations.* Similar to the FAP techniques, TE-Drop also works on the principle that the contribution of each individual MAC output to the output of a neuron in DNNs is small. Hence, a few MAC operations can be ignored without significantly affecting the overall accuracy of the network. In case of a timing error, TE-Drop allows the MAC unit to sample the correctly computed output to an alternate register operating on a delayed clock. The succeeding PE is then bypassed and the correctly computed output is provided instead. The architectural modifications required to realize the concept are shown in Fig. 11.

Figure 11 illustrates the functionality of the TE-Drop with the help of a timing diagram. Here, it is assumed that the shadow clock is delayed by 50% of the clock

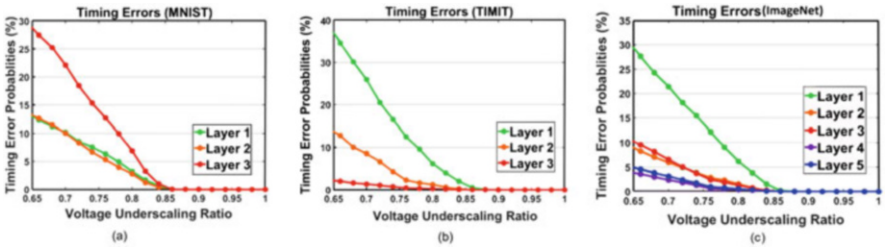

Fig. 12 Timing error probabilities for each layer of the networks used corresponding to (**a**) MNIST. (**b**) TIMIT, and (**c**) ImageNet datasets (adapted from [65])

period. It is assumed that the clock frequency is defined such that the error signal and correct partial sum from the erroneous MAC become available after this much duration. Note that the error signal is obtained by OR-ing the bitwise XOR of all the individual Razor flip-flop at the output of the MAC unit.

3.4.2 Per-Layer Voltage Underscaling

In most of the accelerators, it is assumed that the layers of a DNN are executed in a serial fashion (i.e., one after the other), where processing of each layer can take thousands of clock cycles, depending on the size of the layer. Figure 12 shows the timing error rate versus voltage underscaling ratio plots for each individual layer of three DNN architectures mentioned in Table 1. It can be seen from the figures that the error rate varies significantly across layers. Based on this observation, a per-layer voltage underscaling scheme was proposed in Thundervolt [65] that distributes the total timing error budget equally among the layers of a network to ensure that the more sensitive layers should not consume a significant part of the budget and limits the achievable efficiency gains.

Figure 13 compares two versions of Thundervolt:

1. **ThVolt-Static** where each voltage underscaling ratio is kept the same throughout a DNN execution.
2. **ThVolt-Dynamic** that utilizes per-layer voltage underscaling based on the sensitivity of each layer.

For the baseline, the results of the TEP scheme are also shown. The plot for ThVolt-Static is obtained by sweeping voltage underscaling ratios, and that of ThVolt-Dynamic is obtained by sweeping the total timing error budget. The figures show that for each case Thundervolt outperforms TEP scheme, and for complex tasks (e.g., image classification on the ImageNet dataset) the **ThVolt-Dynamic** outperforms the **ThVolt-Static** approach.

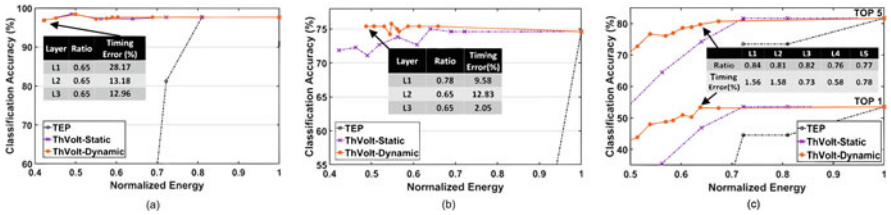

Fig. 13 Accuracy versus energy trade-off using Thundervolt [65] on validation data. (**a**) MNIST. (**b**) TIMIT (**c**) ImageNet (adapted from [65])

4 Secure Deep Learning

In this section, we present different security attacks on DNNs and potential countermeasures.

4.1 Security Attacks on DNNs

Several security attacks have been proposed by exploiting the security vulnerabilities, especially data dependency and unpredicted behavior of intermediate layers of DNN-algorithms during training as well as inference. However, *adversarial and backdooring attacks are some of the most effective and popular attacks for DNNs*. Therefore, in the following subsections, we analyze the state-of-the-art adversarial attacks and proposed backdoor attacks.

4.1.1 Adversarial Perturbation Attacks

It can be defined as the crafted imperceptible noise to perform *targeted* or *untargeted misclassification* in a DNN-based system. In these attacks, an attacker's objective can be summarized as follows: given an image x with a classification label $y = \text{classifier}(x)$, where classifier is the function of the neural network. The attacker aims to find an image x' whose classification label is y', such that $y' = \text{classifier}(x') \neq y$, and $\|x' - x\| \leq \delta$, where δ is an upper bound of the distortion from x to x'. For example, some input adversarial attacks are shown in Fig. 14.

Several attacks have proposed to exploit the adversarial vulnerabilities in DNN-based systems. However, based on the attack methodology, these attacks can broadly be categorized into Gradient Sign Methods and Optimization-based approaches.

1. **Gradient Sign Methods:** These attacks exploit the derivatives and backpropagation algorithm to generate the attack images with imperceptible crafted noise. The main goal of these attacks is to minimize the prediction probability of the true label so as to mislead the network to output a different label (can be targeted or untargeted) other than the ground truth. Some of the most commonly proposed

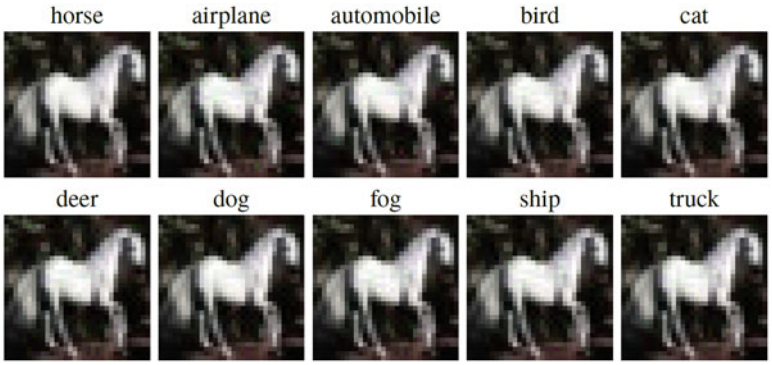

Fig. 14 Clean and adversarial images with different prediction labels, where the clean image of a horse and its adversarial images remain extremely similar, however, their prediction labels are quite distinct and each targets a totally different class

attacks are Fast Gradient Sign (FGS), Iterative Fast Gradient Sign (IFGS), and Jacobian-based saliency map attack (JSMA) methods [42]. Based on the similar principle, there are following attacks which do not require training data and also have less convergence time (in terms of queries):

- *TrISec:* This attack exploits the backpropagation algorithm to identify the small change (attack noise) in input pixels with respect to misclassification at the output, while ensuring the imperceptibility [25].
- *RED-Attack:* Most of the state-of-the-art attacks require a large number of queries to generate an imperceptible attack. However, in resource-constraint scenarios, these attacks may fail, therefore, we proposed a methodology that generates an attack image with imperceptible noise while requiring a very less number of queries [26].

2. **Optimization-based Approaches:** Unlike the gradient-based approaches, these attacks redefine the loss function (i.e., the cost function used for optimization) by adding extra constraints with respect to targeted or untargeted misclassification, and then propose different optimization algorithms to generate adversarial images. For example, Limited Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) [54] and Carlini and Wagner (CW) [5] attacks use the box-constrained L-BFGS algorithm with single and multi-objective optimization, respectively.

Other types of neural networks, i.e., Capsule Networks and Spiking Neural Networks, are emerging as an alternative because of their robustness to affine transformations and potential for offering higher energy efficiency, respectively. However, recent works showed that these networks are also vulnerable to adversarial attacks [38, 39].

4.1.2 Backdoor Attacks

Due to expensive and computationally intensive training, the training (or fine-tuning) of DNNs is usually outsourced which opens the new frontiers of security threats, e.g., *backdoored neural networks (BadNets [14])*. These threats arise due to the involvement of untrusted third party service providers that can insert backdoors by training the ML models on compromised training data, or by altering the DNN structure. The untrusted third party also ensures the required accuracy of the backdoored model on most validation and testing inputs, but cause targeted misclassification or confidence reduction based on *backdoor trigger*. For example, in case of autonomous driving use case, an attacker can introduce the backdoor in a street sign detector while ensuring the required accuracy for classifying street signs in most of the cases, however, it can perform either targeted or untargeted misclassification, i.e., classifies stop signs with a particular sticker as speed limit signs or any other sign different from stop sign. This kind of misclassification can lead to catastrophic effects, e.g., in case of misclassification of a stop sign, autonomous vehicle does not stop at the intersection which can result in an accident.

4.2 Defences Against Security Attacks on DNNs

Several countermeasures have been proposed to defend against the adversarial attacks, i.e., *DNN masking*, *gradient masking*, *training for known adversarial attacks*, and *pre-processing of the CNN inputs* [6]. For examples, Fig. 15 shows

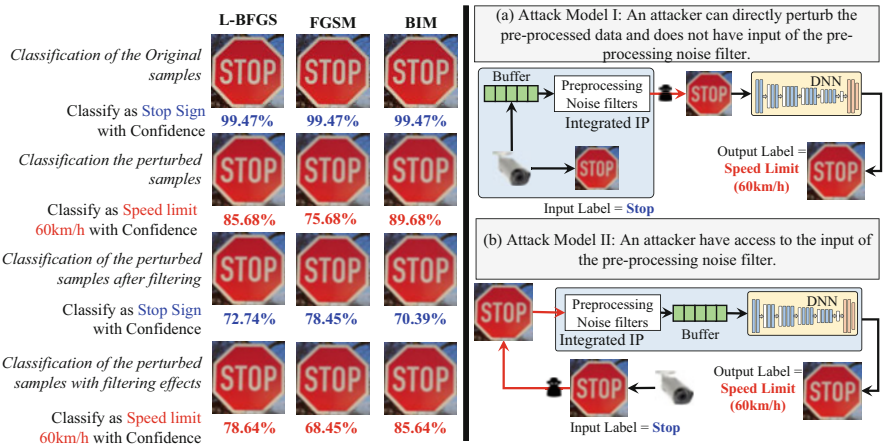

Fig. 15 Impact of the pre-processing filtering on the state-of-the-art adversarial attacks with different attack models with and without the access of filters. (**a**) Attack model I: an attacker can directly perturb the pre-processed data and does not have input of the pre-processing noise filter. (**b**) Attack model II: an attacker have access to the input of the pre-processing noise filter (adapted from [24, 27])

that *low-pass pre-processing filters that can nullify the adversarial attacks if they are not known to the attacker* [24, 27]. Therefore, based on this analysis, we have proposed to utilize the pre-processing quantization to improve the perceptibility of the attack noise [2]. Similarly, Sobel-filers can also be used to decrease the attack strength [55].

However, these defences are not applicable to backdoor-based attacks because the backdoor attacks intrude the networks and are activated through a specific trigger. Therefore, to address these attacks, we propose to use *pruning* as a natural defense because it eliminates the neurons that are dormant on clean inputs, consequently disabling backdoor behavior [14]. Although these defenses are effective, most of them provide defense against known adversarial and backdoor attacks. Therefore, one of the most important problems in designing secure machine learning systems is *the ability to define threats, and model them sufficiently so that any learning system can be trained to be able to identify such threats*.

4.2.1 Generative Adversarial Networks

To address the above-mentioned challenge, Generative Adversarial Networks (GANs) have emerged as one of the prime solutions because of their ability to generate the model by learning to mimic actual models [13]. In particular, GANs is a framework to estimate generative models where simultaneously two models are trained, *generator* (G) and *discriminator* (D) (see Fig. 16). This is achieved through an adversarial process where the two models are competing with each other for achieving two opposite goals. Simply speaking, D is trying to distinguish real images from fake images and G is trying to create images as close as possible to real images so as D will not be able to distinguish them, as illustrated in Fig. 16. When dealing with inference scenarios, the challenge is to provide a training set which includes attack-generated data patterns labeled of course correctly as attacks. For example, an autonomous system may rely on visual information to orient and steer itself or to undertake significant decisions. However, white or patterned noise can be maliciously inserted into a camera feed that may fool the system, and thus results in potentially catastrophic scenarios. The problem with modeling these types of attacks is that the attack models are hard to mathematically formulate, and thus

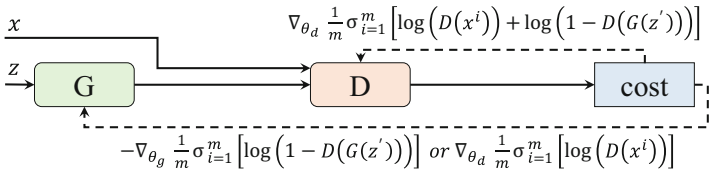

Fig. 16 GANs framework: an illustration of how G and D are trained, adapted from [21]

hard, if not impossible, to replicate and therefore, train the system to recognize them as attacks. Hence, GANs provide us with this capability, as we can utilize the G model to generate-and-evaluate threat models and train the D model to differentiate between what we consider an attack or not. However, GAN-based threat comes with the following challenges [21, 40]:

1. **Collapsing:** In this case G produces only a single sample or set of similar samples, regardless the type of input given to it.
2. **Convergence:** Since G and D models are competing towards achieving two opposite goals, this may make the model parameters to oscillate, destabilizing the training process.
3. **Gradient Vanish:** If one of the two models becomes more powerful than the other, the learning signal is becoming useless, making the system incapable to learn.
4. **Over-Fitting:** This is typically due to the unbalance optimization of G and the D models, e.g., if too much time is spent on minimizing G, then D will most likely collapse to a few states.
5. **Sensitive:** It is characterized by being highly sensitive to the selection of the hyperparameters, i.e., learning rate, momentum, etc.; making the training process much more tedious.

4.2.2 Case Study: Noisy Visual Data, and How GANs Can be Used to Remove Noise and Provide Robustness

To illustrate how a GAN-based framework can be used to define threats and subsequently to provide robustness in a DNN-based system, we use computer vision as an example because security threats in computer vision applications may arise from either physical attacks, cyber attacks or a combination of both. We use the hazing in images to model such threats. To remove this threat, we use the GANs because of their capability in preserving fine details in images and producing results that look perceptually convincing [28]. The goal is to translate the input image with haze, into a haze-free output image. In this case, the noise distribution z is the noisy image, and it is given as input to the GANs, i.e., haze input image. Afterwards, a new sample image F is generated by G. D will receive as input the generated image F and the ground truth haze-free image, to be trained to distinguish between real and artificially generated images (see Fig. 17).

This approach has recently been used for haze removal [8, 52, 63]. These methods mainly differ from each other, based on the utilized deep learning structure for G and D, i.e., using three types of G to solve the optimization of the haze removal problem [63], using the concept of cycle GAN introduced in [67]. Also they may differ for the type of loss function used for the training process, where the overall objective functions is constrained to preserve certain features or priors. However, they provide a solution that, in most of the cases, is capable to improve the quality performances of the state-of-the-art haze removal methods for single image, so as making this quite a promising area.

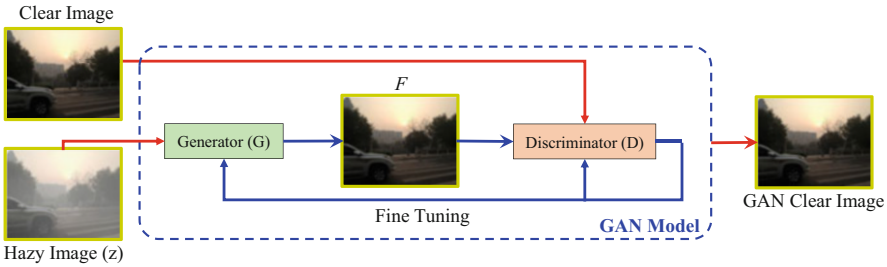

Fig. 17 Example of GANs used for removing haze noise from a single image. The haze image is input to G that generate an output image F and D receives as input both the generated image F and the free-haze image. Input images taken from [32]

5 Open Research Challenges

Machine learning has paved its way to a majority of the fields that involve data processing. However, regardless of all the work which has been carried out in interpreting the neural networks and making the ML-based systems reliable, there are still quite some challenges which are to be addressed before ML algorithms (specifically, DNNs) can be widely accepted for complex safety-critical applications. Following is a list of a few of the main challenges in this direction.

- **Error-Resilience Evaluation Frameworks:** One approach towards this for timing error estimation is proposed in [64]. However, more sophisticated frameworks are required to study the impact of multiple types of reliability threats and their interdependence in a time efficient manner.
- **Methodologies for Designing Robust and Resource-Efficient DNNs:** Retraining a DNN in the presence of hardware-induced faults [15] can improve their resilience. However, there is a need to investigate the types of DNN architectures which are inherently resilient to most (if not all) of the reliability threats. Furthermore, there is a need to investigate frameworks to develop robust ML systems by synergistically investigating reliability and security vulnerabilities.
- **Reliable and Resource-Efficient Hardware Architectures:** With all the security and reliability challenges highlighted in the chapter, there is a dire need to re-think the way current DNN hardware is designed, such that the vulnerabilities that cannot be addressed at the software-level have to be addressed through a robust DNN hardware.
- **Interpretability of Deep Neural Networks:** Developing interpretable DNNs is a challenge, however, it has to be addressed in order to better understand the functionality of the DNNs. This will help us in improving the learning capabilities of the DNNs, as well as in uncovering their true vulnerabilities and thereby will help is developing more efficient and robust network architectures.
- **Practicality of the Attacks:** With the ongoing pace of the research in ML, new methods and types of network architectures are surfacing, e.g., CapsuleNets.

Also, the focus of the community is shifting more towards semi-/un-supervised learning methods as they overcome the need for large labeled datasets. Therefore, there is a dire need to align the focus with the current trends in the ML community. Also, the attacks should be designed considering the constraints of the real systems, i.e., without making unrealistic assumptions about the number of queries and the energy/power resources available to generate an attack. An early work in this direction by our group can be found at [26].

Acknowledgments This work was supported in parts by the German Research Foundation (DFG) as part of the priority program "Dependable Embedded Systems" (SPP 1500—spp1500.itec.kit.edu) and in parts by the National Science Foundation under Grant 1801495.

References

1. Ahmad, H., Tanvir, M., Abdullah, M., Javed, M.U., Hafiz, R., Shafique, M.: Systimator: a design space exploration methodology for systolic array based CNNs acceleration on the FPGA-based edge nodes (2018). arXiv:1901.04986
2. Ali, H., Tariq, H., Hanif, M.A., Khalid, F., Rehman, S., Ahmed, R., Shafique, M.: QuSecNets: quantization-based defense mechanism for securing deep neural network against adversarial attacks (2018). arXiv:1811.01437
3. Athalye, A., Carlini, N., Wagner, D.: Obfuscated gradients give a false sense of security: circumventing defenses to adversarial examples (2018). arXiv:1802.00420
4. Ba, J., Caruana, R.: Do deep nets really need to be deep? In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems, vol. 27, pp. 2654–2662. Curran Associates, New York (2014). http://papers.nips.cc/paper/5484-do-deep-nets-really-need-to-be-deep.pdf
5. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks (2016). arXiv:1608.04644
6. Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., Mukhopadhyay, D.: Adversarial attacks and defences: a survey (2018). arXiv:1810.00069
7. Deng, J., Dong, W., Socher, R., Li, L.: ImageNet: a large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255 (2009). https://doi.org/10.1109/CVPR.2009.5206848
8. Engin, D., Genç, A., Ekenel, H.K.: Cycle-Dehaze: enhanced cycleGAN for single image dehazing. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2018, Salt Lake City, June 18–22, 2018, pp. 825–833 (2018). https://doi.org/10.1109/CVPRW.2018.00127. http://openaccess.thecvf.com/content_cvpr_2018_workshops/w13/html/Engin_Cycle-Dehaze_Enhanced_CycleGAN_CVPR_2018_paper.html
9. Ernst, D., Das, S., Lee, S., Blaauw, D., Austin, T., Mudge, T., Kim, N.S., Flautner, K.: Razor: circuit-level correction of timing errors for low-power operation. IEEE Micro **24**(6), 10–20 (2004)
10. Esteva, A., Robicquet, A., Ramsundar, B., Kuleshov, V., DePristo, M., Chou, K., Cui, C., Corrado, G., Thrun, S., Dean, J.: A guide to deep learning in healthcare. Nat. Med. **25**(1), 24 (2019)
11. Fink, M., Liu, Y., Engstle, A., Schneider, S.A.: Deep learning-based multi-scale multi-object detection and classification for autonomous driving. In: Fahrerassistenzsysteme 2018, pp. 233–242. Springer, Berlin (2019)

12. Gebregiorgis, A., Kiamehr, S., Tahoori, M.B.: Error propagation aware timing relaxation for approximate near threshold computing. In: Proceedings of the 54th Annual Design Automation Conference 2017, p. 77. ACM, New York (2017)
13. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, NIPS'14, pp. 2672–2680. MIT Press, Cambridge (2014). http://dl.acm.org/citation.cfm?id=2969033.2969125
14. Gu, T., Dolan-Gavitt, B., Garg, S.: BadNets: identifying vulnerabilities in the machine learning model supply chain (2017). arXiv:1708.06733
15. Hacene, G.B., Leduc-Primeau, F., Soussia, A.B., Gripon, V., Gagnon, F.: Training modern deep neural networks for memory-fault robustness. In: 2019 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–5. IEEE, Piscataway (2019)
16. Hanif, M.A., Hafiz, R., Shafique, M.: Error resilience analysis for systematically employing approximate computing in convolutional neural networks. In: 2018 Design, Automation and Test in Europe Conference and Exhibition (DATE), pp. 913–916. IEEE, Piscataway (2018)
17. Hanif, M.A., Khalid, F., Putra, R.V.W., Rehman, S., Shafique, M.: Robust machine learning systems: reliability and security for deep neural networks. In: 2018 IEEE 24th International Symposium on On-Line Testing and Robust System Design (IOLTS), pp. 257–260. IEEE, Piscataway (2018)
18. Hanif, M.A., Putra, R.V.W., Tanvir, M., Hafiz, R., Rehman, S., Shafique, M.: MPNA: a massively-parallel neural array accelerator with dataflow optimization for convolutional neural networks (2018). arXiv:1810.12910
19. Henkel, J., Bauer, L., Dutt, N., Gupta, P., Nassif, S., Shafique, M., Tahoori, M., Wehn, N.: Reliable on-chip systems in the nano-era: lessons learnt and future trends. In: Proceedings of the 50th Annual Design Automation Conference, p. 99. ACM, New York (2013)
20. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors (2012). arXiv:1207.0580
21. Hui, J.: Gan why it is so hard to train generative adversarial networks! Elsevier, Amsterdam (2018). https://medium.com/@jonathan_hui/gan-why-it-is-so-hard-to-train-generative-advisory-networks-819a86b3750b
22. Jia, J., Gong, N.Z.: Attriguard: a practical defense against attribute inference attacks via adversarial machine learning. In: 27th {USENIX} Security Symposium ({USENIX} Security 18), pp. 513–529 (2018)
23. Jouppi, N.P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., et al.: In-datacenter performance analysis of a tensor processing unit. In: 2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA), pp. 1–12. IEEE, Piscataway (2017)
24. Khalid, F., Hanif, M.A., Rehman, S., Qadir, J., Shafique, M.: Fademl: understanding the impact of pre-processing noise filtering on adversarial machine learning (2018). arXiv:1811.01444
25. Khalid, F., Hanif, M.A., Rehman, S., Shafique, M.: ISA4ML: training data-unaware imperceptible security attacks on machine learning modules of autonomous vehicles (2018). arXiv:1811.01031
26. Khalid, F., Ali, H., Hanif, M.A., Rehman, S., Ahmed, R., Shafique, M.: Red-attack: resource efficient decision based attack for machine learning (2019). arXiv:1901.10258
27. Khalid, F., Hanif, M.A., Rehman, S., Qadir, J., Shafique, M.: FAdeML: understanding the impact of pre-processing noise filtering on adversarial machine learning. In: Design, Automation and Test in Europe. IEEE, Piscataway (2019)
28. Kupyn, O., Budzan, V., Mykhailych, M., Mishkin, D., Matas, J.: Deblurgan: blind motion deblurring using conditional adversarial networks. In: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
29. Kwon, H., Samajdar, A., Krishna, T.: MAERI: enabling flexible dataflow mapping over DNN accelerators via reconfigurable interconnects. In: Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 461–475. ACM, New York (2018)

30. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998). https://doi.org/10.1109/5.726791
31. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436 (2015)
32. Li, B., Ren, W., Fu, D., Tao, D., Feng, D., Zeng, W., Wang, Z.: Benchmarking single-image dehazing and beyond. IEEE Trans. Image Process. **28**(1), 492–505 (2019)
33. Limbrick, D.B., Mahatme, N.N., Robinson, W.H., Bhuva, B.L.: Reliability-aware synthesis of combinational logic with minimal performance penalty. IEEE Trans. Nuclear Sci. **60**(4), 2776–2781 (2013)
34. Lu, W., Yan, G., Li, J., Gong, S., Han, Y., Li, X.: FlexFlow: a flexible dataflow accelerator architecture for convolutional neural networks. In: 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA), pp. 553–564. IEEE, Piscataway (2017)
35. Lyons, R.E., Vanderkulk, W.: The use of triple-modular redundancy to improve computer reliability. IBM J. Res. Development **6**(2), 200–209 (1962)
36. Marchisio, A., Shafique, M.: Capstore: energy-efficient design and management of the on-chip memory for CapsuleNet inference accelerators (2019). arXiv:1902.01151
37. Marchisio, A., Hanif, M.A., Shafique, M.: CapsAcc: an efficient hardware accelerator for CapsuleNets with data reuse (2018). arXiv:1811.08932
38. Marchisio, A., Nanfa, G., Khalid, F., Hanif, M.A., Martina, M., Shafique, M.: Capsattacks: robust and imperceptible adversarial attacks on capsule networks (2019). arXiv:1901.09878
39. Marchisio, A., Nanfa, G., Khalid, F., Hanif, M.A., Martina, M., Shafique, M.: SNN under attack: are spiking deep belief networks vulnerable to adversarial examples? (2019). arXiv:1902.01147
40. Metz, L., Poole, B., Pfau, D., Sohl-Dickstein, J.: Unrolled generative adversarial networks. In: Proceedings of the International Conference on Learning Representations (ICLR'17) (2017)
41. Nakhaee, F., Kamal, M., Afzali-Kusha, A., Pedram, M., Fakhraie, S.M. Dorosti, H.: Lifetime improvement by exploiting aggressive voltage scaling during runtime of error-resilient applications. Integr. VLSI J. **61**, 29–38 (2018)
42. Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A.: The limitations of deep learning in adversarial settings. In: 2016 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 372–387. IEEE, Piscataway (2016)
43. Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z.B., Swami, A.: Practical black-box attacks against machine learning. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, pp. 506–519. ACM, New York (2017)
44. Putra, R.V.W., Hanif, M.A., Shafique, M.: ROMANet: fine-grained reuse-driven data organization and off-chip memory access management for deep neural network accelerators (2019). arXiv:1902.10222
45. Raghunathan, B., Turakhia, Y., Garg, S., Marculescu, D.: Cherry-picking: exploiting process variations in dark-silicon homogeneous chip multi-processors. In: 2013 Design, Automation and Test in Europe Conference and Exhibition (DATE), pp. 39–44. IEEE, Piscataway (2013)
46. Reagen, B., Gupta, U., Pentecost, L., Whatmough, P., Lee, S.K., Mulholland, N., Brooks, D., Wei, G.Y.: Ares: a framework for quantifying the resilience of deep neural networks. In: 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC), pp. 1–6. IEEE, Piscataway (2018)
47. Rehman, S., Shafique, M., Henkel, J.: Reliable Software for Unreliable Hardware: A Cross Layer Perspective. Springer, Berlin (2016)
48. Schorn, C., Guntoro, A., Ascheid, G.: Efficient on-line error detection and mitigation for deep neural network accelerators. In: International Conference on Computer Safety, Reliability, and Security, pp. 205–219. Springer, Berlin (2018)
49. Shafique, M., Garg, S., Henkel, J., Marculescu, D.: The EDA challenges in the dark silicon era: temperature, reliability, and variability perspectives. In: Proceedings of the 51st Annual Design Automation Conference, pp. 1–6. ACM, New York (2014)
50. Shokri, R., Stronati, M., Song, C., Shmatikov, V.: Membership inference attacks against machine learning models. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 3–18. IEEE, Piscataway (2017)

51. Suciu, O., Marginean, R., Kaya, Y., Daume III, H., Dumitras, T.: When does machine learning {FAIL}? Generalized transferability for evasion and poisoning attacks. In: 27th {USENIX} Security Symposium ({USENIX} Security'18), pp. 1299–1316 (2018)
52. Swami, K., Das, S.K.: Candy: conditional adversarial networks based fully end-to-end system for single image haze removal (2018). https://arxiv.org/abs/1801.02892v2
53. Sze, V., Chen, Y.H., Yang, T.J., Emer, J.S.: Efficient processing of deep neural networks: a tutorial and survey. Proc. IEEE **105**(12), 2295–2329 (2017)
54. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks (2013). arXiv:1312.6199
55. Tariq, H., Ali, H., Hanif, M.A., Khalid, F., Rehman, S., Ahmed, R., Shafique, M.: SSCNets: a selective sobel convolution-based technique to enhance the robustness of deep neural networks against security attacks (2018). arXiv:1811.01443
56. Tiwari, A., Torrellas, J.: Facelift: hiding and slowing down aging in multicores. In: Proceedings of the 41st Annual IEEE/ACM International Symposium on Microarchitecture, pp. 129–140. IEEE Computer Society, Washington (2008)
57. Tramèr, F., Zhang, F., Juels, A., Reiter, M.K., Ristenpart, T.: Stealing machine learning models via prediction APIs. In: 25th {USENIX} Security Symposium ({USENIX} Security'16), pp. 601–618 (2016)
58. Vadlamani, R., Zhao, J., Burleson, W., Tessier, R.: Multicore soft error rate stabilization using adaptive dual modular redundancy. In: Proceedings of the Conference on Design, Automation and Test in Europe, pp. 27–32. European Design and Automation Association, Leuven (2010)
59. Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., Fergus, R.: Regularization of neural networks using dropconnect. In: International Conference on Machine Learning, pp. 1058–1066 (2013)
60. Wang, B., Gong, N.Z.: Stealing hyperparameters in machine learning. In: 2018 IEEE Symposium on Security and Privacy (SP), pp. 36–52. IEEE, Piscataway (2018)
61. Wei, X., Yu, C.H., Zhang, P., Chen, Y., Wang, Y., Hu, H., Liang, Y., Cong, J.: Automated systolic array architecture synthesis for high throughput CNN inference on FPGAs. In: Proceedings of the 54th Annual Design Automation Conference 2017, p. 29. ACM, New York (2017)
62. Whatmough, P.N., Das, S., Bull, D.M., Darwazeh, I.: Circuit-level timing error tolerance for low-power DSP filters and transforms. IEEE Trans. Very Large Scale Integration Syst. **21**(6), 989–999 (2013)
63. Yang, X., Xu, Z., Luo, J.: Towards perceptual image dehazing by physics-based disentanglement and adversarial training. In: Thirty-Second AAAI Conference on Artificial Intelligence (AAAI) (2018)
64. Zhang, J.J., Garg, S.: Fate: fast and accurate timing error prediction framework for low power DNN accelerator design. In: Proceedings of the International Conference on Computer-Aided Design, p. 24. ACM, New York (2018)
65. Zhang, J., Rangineni, K., Ghodsi, Z., Garg, S.: ThUnderVolt: enabling aggressive voltage underscaling and timing error resilience for energy efficient deep neural network accelerators (2018). arXiv:1802.03806
66. 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: 2018 IEEE 36th VLSI Test Symposium (VTS), pp. 1–6. IEEE, Piscataway (2018)
67. Zhu, J., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 2242–2251 (2017). https://doi.org/10.1109/ICCV.2017.244