Microelectronics Reliability 130 (2022) 114498

Contents lists available at ScienceDirect

Microelectronics Reliability

journal homepage: www.elsevier.com/locate/microrel

Reliability evaluation of FPGA based pruned neural networks☆

Zhen Gao [a,*], Yi Yao [a], Xiaohui Wei [a], Tong Yan [a], Shulin Zeng [b], Guangjun Ge [b], Yu Wang [b], Anees Ullah [c], Pedro Reviriego [d]

[a] Tianjin University, Tianjin 300072, China
[b] School of Electronic Engineering, Tsinghua University, Beijing 100084, China
[c] University of Engineering and Technology, Peshawar, Abbottabad 220101, Pakistan
[d] Universidad Carlos III de Madrid, 28911 Leganés, Spain

ARTICLE INFO

Keywords:
Convolutional Neural Networks (CNNs)
Pruning
Reliability
FPGAs
Fault injection

ABSTRACT

Convolutional Neural Networks (CNNs) are widely used for image classification. To fit the implementation of CNNs on resource-limited systems like FPGAs, pruning is a popular technique to reduce the complexity. In this paper, the robustness of the pruned CNNs against errors on weights and configuration memory of the FPGA accelerator is evaluated with VGG16 as a case study, and two popular pruning methods (magnitude-based and filter pruning) are considered. In particular, the accuracy loss of the original VGG16 and the ones with different pruning rates is tested based on fault injection experiments, and the results show that the effect of errors on weights and configuration memories are different for the two pruning methods. For errors on weights, the networks pruned using both methods demonstrate higher reliability with higher pruning rates, but the ones using filter pruning are relatively less reliable. For errors on configuration memory, errors on about 30% of the configuration bits will affect the CNN operation, and only 14% of them will introduce significant accuracy loss. However, the effect of the same critical bits is different for the two pruning methods. The pruned networks using magnitude-based method are less reliable than the original VGG16, but the ones using filter pruning are more reliable than the original VGG16. The different effects are explained based on the structure of the CNN accelerator and the properties of the two pruning methods. The impact of quantization on the CNN reliability is also evaluated for the magnitude-based pruning method.

1. Introduction

Modern Convolutional Neural Networks (CNNs) have achieved outstanding performance for image classification tasks, but the amount of parameters in the model is huge, which requires considerable storage and computational resources [1,2] and limits their use on embedded systems. To solve this problem, pruning methods are proposed to compress the network [3–6] which not only reduces the number of network parameters but also decreases the computational complexity. However, such compression might impact the reliability of the network [7].

To speed CNN systems, many researchers proposed to implement CNN based on Field Programmable Gate Arrays (FPGAs) because FPGA based accelerator exhibits unique advantages compared to the ASIC-based implementations such as intensive application-specific customization or convenient integration and rapid deployment [8–10]. Especially for applications that require changes in the functionality, SRAM based FPGA is preferred due to its re-configurability. However, SRAM-FPGAs are sensitive to soft errors on both user memory and configuration memory [11,12]. As reported in [14], robots were used to investigate and clean up the Fukushima Nuclear Power Plant in Japan, but the strong radiation caused the failure of the AI-based autonomous operation. A similar situation exists in the application of NNs in space environments [15,16]. Another scenario is the self-driving car. As reported

☆ This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant No. 62171313, in part by the NSFC Joint Foundation under Grant No. 20200509, in part by the ACHILLES project PID 2019-104207RB-I00 and the Go2Edge network RED2018-102585-T funded by the Spanish Ministry of Science and Innovation and in part by the Department of Research and Innovation of Madrid Regional Authority with the EMPATIA-CM Research Project (Reference Y2018/TCS-5046).

* Corresponding author.
E-mail addresses: zgao@tju.edu.cn (Z. Gao), yiyao@tju.edu.cn (Y. Yao), weixh2019@tju.edu.cn (X. Wei), Tyan@tju.edu.cn (T. Yan), zengsl18@mails.tinghua.edu.cn (S. Zeng), yu-wang@mail.tinghua.edu.cn (Y. Wang), aneesullah@uetpeshawar.edu.pk (A. Ullah), revirieg@it.uc3m.es (P. Reviriego).

https://doi.org/10.1016/j.microrel.2022.114498
Received 2 August 2021; Received in revised form 4 December 2021; Accepted 9 February 2022
Available online 18 February 2022

in [17], the complex electromagnetic interferences may cause the failure of the CNN based traffic sign recognition. Furthermore, in addition to degradation of the CNN performance, faults on the configuration memory of the SRAM-FPGA could also cause system exceptions, such as system stall and early termination [18]. In summary, errors on both parameters and configuration memories for FPGA based CNN accelerators pose severe reliability problems for the application of CNN in critical environments.

This paper aims to evaluate the reliability of pruned CNNs to errors on weights and configuration memory. Compared with related works about the reliability of pruned CNNs (discussed in Section 2.1), the contributions of this work include:

(1) A typical CNN and the pruned ones with two pruning methods are implemented on an advanced FPGA accelerator, and a hardware fault injection tool is implemented on the accelerator for reliability evaluation.
(2) The reliability of CNNs with different pruning rates to faults on weights is compared under different error rates for two popular pruning methods.
(3) The reliability of CNNs with different pruning rates to faults on weights of different layers is compared for two popular pruning methods.
(4) The effect of quantization on the reliability of the pruned network is evaluated.
(5) The reliability of CNNs with different pruning rates to faults on configuration memory is compared for two popular pruning methods.
(6) Fault injection experiments for the DSP array and adder tree in the accelerator are performed separately to explain the different effect of errors on the configuration bits for two pruning methods.

The rest of the paper is organized as follows. Section 2 introduces the preliminaries of this work, including the related work, the basics for FPGA based CNN accelerators, VGG 16, and two popular pruning methods. Section 3 discusses the implementation of the pruned networks on FPGA accelerators and evaluates the network performance. Section 4 evaluates the reliability of pruned networks to errors on weights and configuration memories based on the fault injection platform. Finally, the paper is concluded in Section 5.

2. Preliminaries

2.1. Faults on FPGA based CNN accelerators

Soft errors can corrupt both the configuration memory and the user memories (including flip-flops and Block RAMs) of SRAM-FPGAs [12], and the Single Event Upset (SEU) effect is typically the most frequent one in space environments [11,19]. While errors on user memory only corrupt the data stored, errors on the configuration memory can change the circuit implemented by the FPGA and can only be corrected by reconfiguration. For FPGA based CNN accelerators, the user memories are used to store the parameters (e.g. weights, bias, and so on) and feature maps, and all the processing modules are determined by the configuration memory.

Many works have studied the effect of errors on parameters and feature maps. In [20], the reliability of CNNs with 2 or 4 convolutional layers was evaluated when some weights are corrupted by SEUs. The results show that CNNs with large kernels are more robust. In [21], the reliability of LeNet-5 was evaluated when Gaussian noise is added to the weights, and found that the layers closer to the output layer are more vulnerable to errors. The study in [22] reveals that faults on weights and bias with an error rate lower than 10^{-4} would not degrade the classification accuracy, which is consistent with the results in [23] for AlexNet. Ref. [24] performed a similar study for VGG16 and ResNet50, and showed that faults on weights with an error rate lower than 10^{-7} will not degrade the accuracy, and the memories for feature maps are up to 50 times more tolerant to faults than those for the weights. In [25], the reliability of VGG 16, ResNet50, and InceptionV3 was evaluated by flipping some bits of the main parameters, including weights, biases, and the batch normalization (BN) parameters. The results show that ResNet50 and InceptionV3 are more robust to parameter errors than VGG16. The authors in [26] investigated the effects of faults on the data path and buffers in DNN accelerators for different data types, bit positions and layers, and found that high-order bits are vulnerable to faults and normalization layers are effective to reduce the impact of such faults. The authors of [27] also studied the reliability to faults on the weight memory and showed that the nonlinear activation operation is effective for fault tolerance by clipping the values of feature maps. Authors in [28] studied the effect of errors on weights for different layers in floating-point implemented LeNet-5 and Yolo networks based on fault injection experiments. Results show that the convolution layers in LeNet-5 are more reliable than the full connection layers, but such feature is not obvious for layers in Yolo. There are also several works discussing the impact of network compression on the resiliency of CNNs to faults on weights. Reference [29] evaluated the reliability of LeNet-5 and VGG16 with different data types and pruning methods. The testing results showed that there is no big difference in performance for irregular and structured pruning schemes, and quantization can effectively increase the fault resilience of DNNs. Since only a fixed pruning rate was evaluated, the impact of the pruning rate on the network reliability was not investigated. Similarly, reference [7] evaluated the impact of soft errors on AlexNet, RestNet 18 and ResNet50 with different data types and pruning rates, and concluded that quantization can bring a 27.4× reliability increase relative to the 32-bit floating-point baseline, and another 4× improvement could be achieved when combined with pruning. However, this work only considered pruning rates around 70%–80%. In our initial work [30], we investigated the impact of the pruning rates on the reliability of VGG16 to errors on different parameters and showed that the networks with higher pruning rates are more reliable. But this conclusion was obtained for 32-bit floating-point weights. As we will introduce in Section 2.5, pruning methods can be categorized into magnitude-based pruning and filter pruning. All above works are based on the magnitude-based pruning, and few works considered the more advanced filter pruning on the reliability of the CNN.

Several recent works studied the reliability of FPGA based CNN to SEUs on the configuration memories based on fault injection experiments. For example, [31] implemented a simple NN with 3 layers based on SRAM-FPGA with separate resources for each layer, and the fault injection experiment results showed that most SEUs will not degrade classification performance, and faults on the last layer have larger impact on the result. Authors in [32] implemented a CNN for traffic sign recognition based on SRAM-FPGAs, and the experiment results showed that SEUs on about 20% of the configuration memories will cause wrong classifications. Similarly, reference [33] studied the reliability of FPGA-based CNN accelerators based on radiation experiments, and found that most errors are tolerable, and some layers are more reliable than others. Research in [34] proves that the sensitivity of the convolutional layers to radiation can be reduced by 39% by binary quantization, but the percentage of critical errors increases by 12%. Reference [17] performed a systematic evaluation of the FPGA-based CNN accelerator to permanent faults based on fault injection experiments. Results showed that the system exceptions caused by faults on memory access logic and control unit dominates the reliability of the system. As far as the authors know, there is no research studying the impact of pruning on the CNN reliability to SEUs on configuration memory, especially with modern accelerator architectures.

2.2. FPGA accelerators for CNNs

FPGA-based CNN accelerators can be divided into two types [35]:

Streaming Architectures (SAs) and Single Computation Engines (SCEs). SAs directly map layers to different resources of an FPGA to perform computations easily and efficiently. The reliability evaluations in [29,32] are based on such kind of accelerator. However, with the rapid development of CNN algorithms, SCEs have become more popular [8,10]. Different from SAs, SCEs apply instruction-set architectures (ISAs) to reuse the same modules for processing of different layers. The CNN operations are transformed into instructions that can be deployed on the hardware accelerator by the software compiler [35]. In this way, SCEs can easily handle different neural network architectures and parameters without reconfiguring the FPGA. The work of [17] is based on such accelerator. We will also use an ISA-based FPGA CNN accelerator to implement the VGG16 and the pruned networks in this paper.

As shown in Fig. 1, the basic structure for the ISA-based FPGA CNN accelerator is composed of the instruction dispatch module, data mover, compute module, and on-chip memory pool. The instruction dispatch module is responsible for instruction parsing, scheduling, and dispatching. The LOAD and SAVE modules inside the data mover are responsible for data transfer between DDR and on-chip memory pool. Among these modules, the CONV module is the key component that performs most of the computations and consumes most the of FPGA resources. As shown in Fig. 2, the CONV module utilizes an array of processing elements (PEs) to realize high parallelism for the convolution operation. In this work, the Xilinx Zynq SoC (XC7Z020) is used for the CNN implementation. On this platform, the DSP48E2 block and adder tree based on Look Up Tables (LUTs) are used for the multiplications and accumulations, respectively.

2.3. Structure of VGG16

VGG is a deep CNN for image recognition proposed by the visual geometry group of Oxford University [36]. This network achieved a significant improvement by pushing the depth to 16–19 layers; and won the 1st and 2nd places in the localization and classification tracks in ILSVRC2014, respectively. Since then, VGG16 has been used as the baseline for many works about performance improvement or network simplification.

As shown in Fig. 3, the VGG16 network is composed of 13 convolution layers, 5 pooling layers, and 3 full connection layers. The size of the kernel is fixed to 3 × 3 for all the convolution layers, and the Rectified Linear Unit (ReLU) is used as the non-linear activation function after each convolution layer. The maximum pooling is used with a

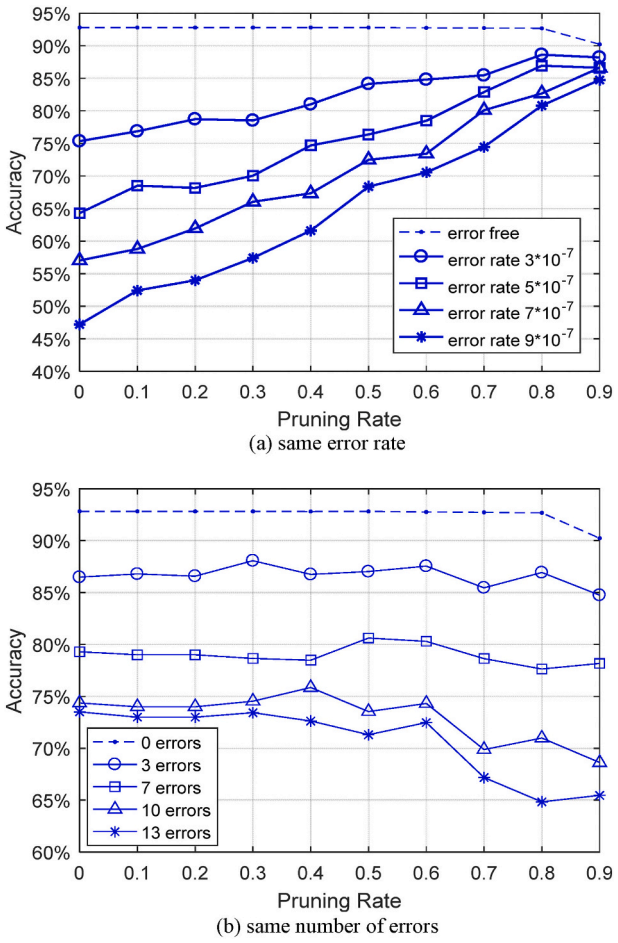


Fig. 2. Structure of the CONV module.

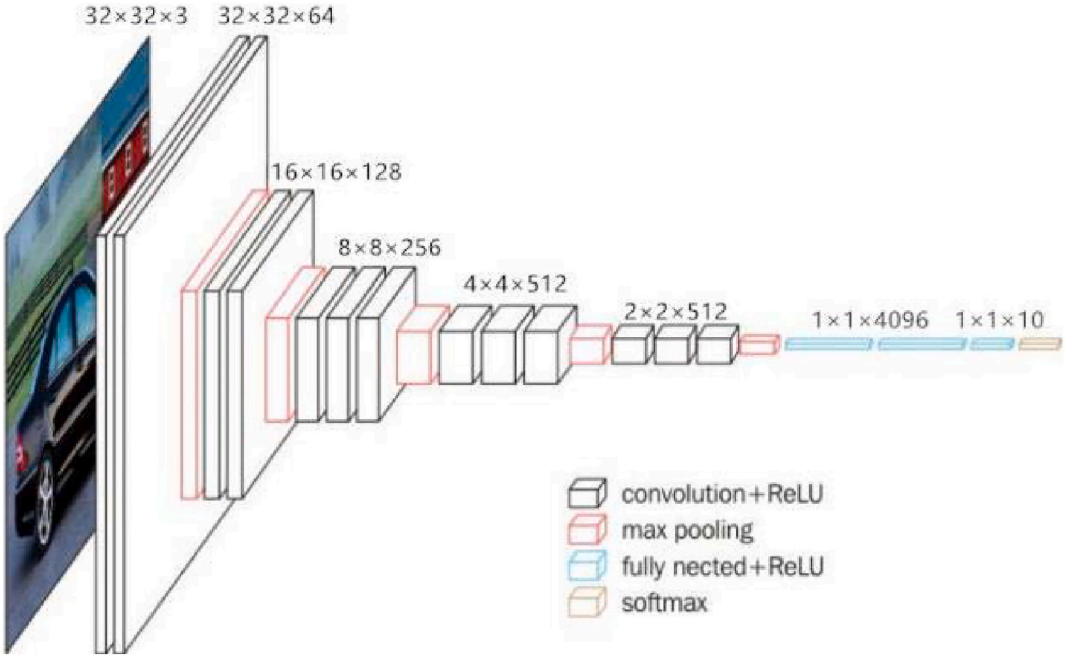


Fig. 1. Structure of the ISA-based FPGA CNN accelerator.

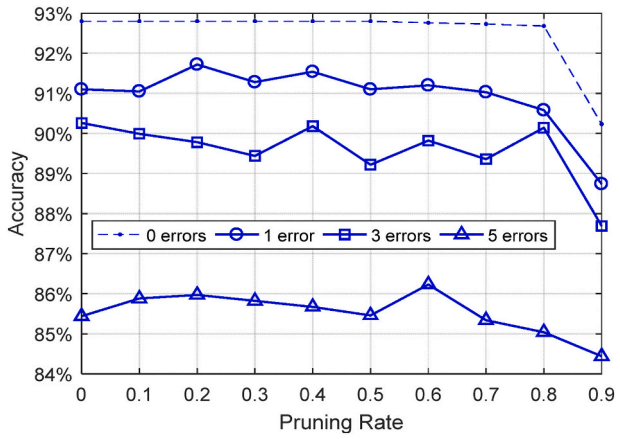


Fig. 3. Structure of VGG16 for CIFAR-10.

window size of 2 × 2 and stride of 2 to reduce the size of the feature maps by half in both the vertical and horizontal dimensions. The original VGG16 network is designed for the classification of images in the ImageNet dataset. There are 1000 categories in ImageNet and the size of each image is 224 × 224 × 3, so the size of the input layer is 224 × 224 × 3, and there are 1000 nodes in the last full connection layer. In this paper, VGG16 is used for image classification on CIFAR-10, which includes pictures with a size of 32 × 32 × 3 from 10 categories, so the size of the input layer is changed to 32 × 32 × 3. Then the width and height for the following layers are adjusted accordingly, and the final fully connected layer only contains 10 nodes. The outputs of the 10 nodes (e_i, $i = 1, 2, \ldots, 10$) are transformed by softmax to a score between [0,1]. Finally, the label corresponding to the maximum score is output as the classification result.

2.4. Parameters in VGG16

A pixel value of one output feature map in a convolution layer or full connection layer could be calculated as

$$f_o(\bar{x}_i) = \overline{W}_o \bar{x}_i + b_o = \sum_{n=1}^{N} W_n^o x_n^i + b_o \quad (1)$$

where i and o are the indices for the input feature maps (or channels) and the output feature maps (or kernels), respectively, $\overline{W}_o = [W_1^o, W_2^o, \ldots, W_N^o]$ and $\bar{x}_i = [x_1^i, x_2^i, \ldots, x_N^i]$ are the weights of the o-th convolution kernel and a data block of the i-th input feature map, respectively, and b_o is the corresponding bias. $\overline{W}_o$ and b_o are trainable parameters. For the convolution layers, $N = 3 \times 3 \times$ I where I is the number of input feature maps. For the three fully connected layers, the values of N are 4096, 4096 and 10, respectively.

To accelerate the network training, the Batch Normalization (BN) technique is usually applied between the convolution and ReLU in each convolution layer [37]. In this case, Eq. (1) for convolution layer is changed to

$$F_o(\overline{X}_o) = \gamma_o \frac{\overline{X}_o - u_o}{\sqrt{\sigma_o^2 + \varepsilon}} + \beta_o \quad (2)$$

where $\overline{X}_o$ is the result of Eq. (1), u_o and ${\sigma_o}^2$ are the mean value and variance of $\overline{X}_o$, γ_o and β_o are a scaler and an offset, and ε is a small number preventing the divisor from being 0. In the implementation, u_o and ${\sigma_o}^2$ are obtained statistically during the training process for each kernel and are stored for use in the inference period. γ_o and β_o are trainable parameters. Since the subtraction of the mean value u_o would remove the effect of the bias for convolutional layers, b_o is only used in the fully connected layers in the implementation.

In summary, the parameters of VGG16 include the weights ($\overline{W}$), bias for fully connected layers (b), and BN factors for convolution layers (u, σ^2, γ and β). The number of each type of parameters in VGG16 for CIFAR10 is listed in Table 1. As we can see, the weights account for 99.9% of all the parameters, so pruning is only performed over weights.

2.5. Pruning of neural networks

As introduced above, CNNs are both computationally intensive and memory intensive. But according to the analysis in [3], there exists redundancy in neural networks, so it is possible to remove part of the nodes and connections with negligible performance degradation. The pruning technique is proposed for this purpose. In this paper, two popular pruning methods are considered for evaluation.

2.5.1. Magnitude-based pruning

The magnitude-based method is popular for CNNs for image classification tasks, in which the connections with small weights would be removed since they are prone to have less contribution to the final result [3]. Such methods usually perform three steps: 1) train a network to learn which connections are important; 2) prune the unimportant connections; 3) retrain the network to finetune the weights of remaining connections. On this basis, two methods are compared in [4]. One is to remove the same portion of connections in each layer ('Class-uniform pruning'), and the other is to sort the weights from all layers according to their magnitude and remove the part of the connections with the smallest weights regardless of which layer the weights belong to ('Class-blind pruning'). Performance comparisons show that Class-blind pruning outperforms Class-uniform pruning. In this paper, the Class-blind scheme is applied.

2.5.2. Filter pruning (structured pruning)

Although the magnitude-based pruning reduces a significant number of weights, it may not adequately reduce the computation costs due to irregular sparsity in the networks. To solve this problem, several works proposed filter pruning or structured pruning, where the whole filters are removed together with their connecting feature maps [5,6]. This approach can significantly reduce the computation costs, especially for the FPGA or GPU based accelerators using parallel computing structures. In this paper, the popular scheme proposed in [5] is used for evaluation based on an open source implementation [38], and the basic operation includes six steps: 1) determine the target layers by evaluating the sensitivity of each layer; 2) calculate the sum of magnitude of all weights in each filter belongs to the target layers; 3) sort all filters by the magnitude sum; 4) remove part of the filters with the smallest magnitude sum; 5) remove the filters that correspond to the feature maps generated by the removed filters in the previous layer; 6) retrain the network to finetune the weights of remaining connections. Since the removed filters in step 4) will cause additional filter to be removed in step 5), the pruning rate cannot be exactly controlled as in the magnitude-based pruning.

3. Implementation and performance evaluation of pruned VGG16 networks

3.1. Implementation of VGG16 and pruned networks

The original VGG16 and the pruned networks were implemented on

Table 1
Number of parameters in VGG16 network.

Parameters	Weights	Bias	BN	
			$\gamma + \beta$	$u + \sigma^2$
Numbers	15,239,872	1034	4224 + 4224	4224 + 4224

Pynq-Z2 board with Xilinx Zynq SoC (XC7Z020) based on the ISA-based accelerator structure. Parallelism of 512 was applied, where 4 PEs run in parallel and 32 DSPs are used in each PE to produce results for 8 output channels simultaneously based on 8 input channels. The resource usage of the accelerator is shown in Table 2. About half of the main resources in the FPGA are used, and all DSPs are used in the CONV module. It should be noted that the original VGG16 and the pruned networks are implemented using the same PEs and BRAMs, and the only difference is that fewer weights will be loaded in each cycle for the network with higher pruning rate. For fixed fault rate on unit memory, fewer weights loaded means fewer errors on the weights.

Figs. 4 and 5 show the distribution of the weights over the different layers for the original VGG16 and the networks with different pruning rates with magnitude-based pruning and the filter pruning, respectively. As we can see, most of the weights are located on convolution layers 9–13, and the pruned weights are also from these layers. For the filter pruning method, the 1st and the 8th–13th convolutional layers are identified as non-sensitive layers based on the sensitivity evaluation algorithm in [5], so only filters from these layers are removed based on the procedures introduced in Section 2.5. The pruning rates in Fig. 4 for the magnitude-based pruning are accurate, but those for filter pruning are actually 19.43% (for VGG-20%), 50.79% (for VGG-50%) and 79.83% (for VGG-80%). In this study, the original VGG16 and the pruned ones are implemented using the same accelerator shown in Fig. 2. For the pruned network with magnitude-based method, the only difference with the original VGG16 is that some weights are fixed to be 0. While for the pruned network with filter pruning, the only difference with the original VGG16 is that the CONV module is less used due to fewer number of kernels.

3.2. Performance evaluation of pruned networks

The classification accuracy of the original VGG16 and the networks with different pruning rates was evaluated on CIFAR-10. The data set includes 60,000 pictures from 10 categories, among which 50,000 pictures are used for training (training set) and the other 10,000 are used for testing (test set). In the implementation, the weight decay and momentum are set to be 0.0001 and 0.9, respectively. The learning rate starts from 0.1 and is divided by 10 at 30, 60 and 90 iterations and the training ends at 100 iterations. The input image data and the intermediate feature maps are quantified to 8-bit integers.

For the magnitude-based pruning method, the weights are quantified with different word lengths for the evaluation of the effect of quantization on the network reliability, including 6-bit, 8-bit, 12-bit and 16-bit. The performance of pruned networks under different quantization sizes is listed in Table 3, in which 'VGG-x%' means x% of the weights (connections) are pruned. As we can see from the table, the accuracy for weights with 8-bit, 12-bit and 16-bit quantization schemes are very close, and that for 6-bit quantified weights is about 2.5% lower. In addition, the accuracy is almost the same for pruning rates between 0% (original) and 70%, and drops dramatically for a pruning rate of 80%. Based on these results, the reliability of the pruned networks in the following sections is only evaluated for pruning rates of 10%, 30%, 50% and 70%.

For the filter pruning method, the implementation with 8-bit quantized weights is used for evaluation, and the accuracy for pruning rates of 20%, 50% and 80% are shown in Table 4. As we can see, the accuracy for VGG-80% (92.61%) is much higher than that using the magnitude-based pruning method (85.58%). This is because the magnitude-based pruning removes weights only based on the amplitude regardless of the layer. But based on the analysis in [5], the weights with the same amplitude may have different importance for the network. By comparing Figs. 4 and 5 for pruning rate of 50%, we can see that the filter pruning method removes more weights from the 9th – 13th layers (non-sensitive) than the magnitude-based method, so that the weights in the sensitive layers are kept to maintain the performance of the network.

Table 2
Resource consumption of the CNN accelerator.

	LUT	DSP	Register	BRAM
Resources for CNN	26,605	128	26,901	88
Resources in XC7Z020	53,200	220	106,400	140
Resource Utilization	50%	58%	25%	63%

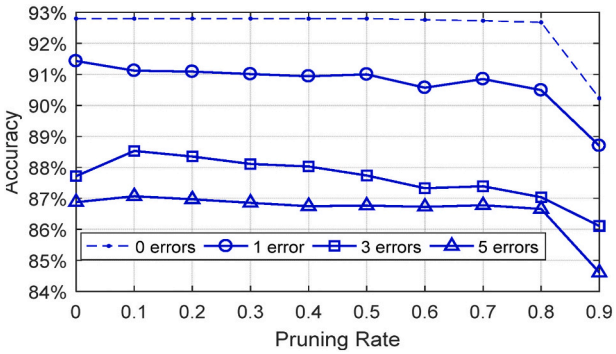


Fig. 4. Distribution of weights for different pruning rates (magnitude-based pruning).

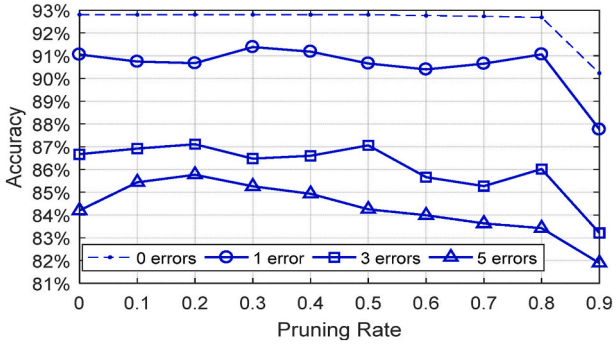


Fig. 5. Distribution of weights for different pruning rates (filter pruning).

Table 3
Accuracy of VGG16 and pruned networks (magnitude-based).

	0%	10%	30%	50%	70%	80%
16 bits	92.85%	92.85%	92.83%	92.82%	92.73%	85.88%
12 bits	92.84%	92.82%	92.84%	92.81%	92.72%	85.90%
8 bits	92.67%	92.67%	92.66%	92.75%	92.56%	85.58%
6 bits	90.11%	90.10%	90.10%	90.17%	89.94%	84.36%

Table 4
Accuracy of VGG16 and pruned networks (filter pruning).

	0%	20%	50%	80%
8 bits	92.67%	92.67%	92.74%	92.61%

4. Reliability evaluation of pruned networks

4.1. Fault injection platform

To evaluate the reliability of the pruned networks, faults have been injected using an adapted version of the fault injection tool in [39] and implemented on a Xilinx Zynq 7000 SoC (XC7Z020). The experimental setup of the fault injection platform is shown in Fig. 6. The injection

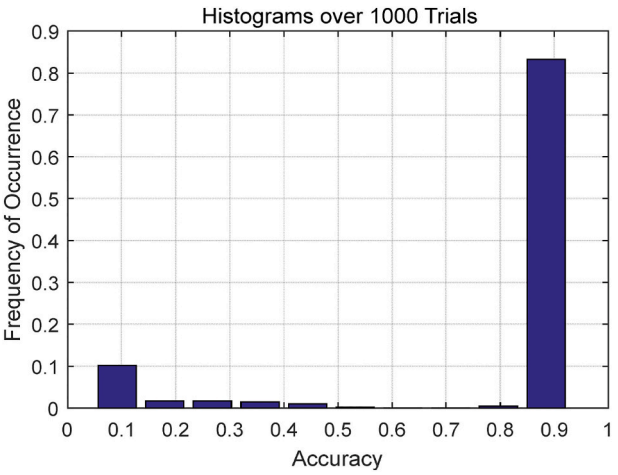


Fig. 6. Fault injection platform for CNN reliability evaluation on one Pynq-Z2 board.

platform consists of the Processing System (PS) and the Programmable Logic (PL). The PS consists of the ARM Cortex-A9 processor and dedicated controllers for different peripherals e.g. DDR memory controller, SD controller, UART controller etc. The DDR controller is responsible for storing the weights file and the fault lists, including a list of bits in the user memory and a list of essential bits in the configuration memory. These lists are generated during the compile time in Vivado and it is used by the injection algorithm for reliability evaluation of the Design Under Test (DUT), which is the CNN system in our case. The SD controller is responsible for reading configuration files and images from the SD card, and the UART module is used to log the evaluation results to the PC. The PL part consists of the DUT and a synchronizer block. The latter is responsible for controlling the clock to the DUT. Furthermore, the DUT receives inputs (image data in our case) from the ARM processor in the PS part, and returns the processing results to the ARM. Another important module housed by the PL part is the Internal Configuration Access Port (ICAP) module, which allows the ARM processor to access the user memory (weights in our case) and the configuration memory related to the DUT (mainly the PE array), and modify it in runtime for error injection or removal. The modules in the PL region are connected to the PS region through AXI buses.

The ARM processor runs the software that controls the fault injection process. The fault injection starts by freezing the clock to the DUT in the PL part through the synchronizer module. This is followed by reading back the target bit from a frame of the configuration memory through the ICAP port. The address of the configuration memory bits and the user memory bits (weights) for fault injection is extracted from the fault list stored in the DDR memory. The read back values are corrupted by inserting a bit flip for SEU emulation and written back. This is followed by resuming the design clock. After each fault injection, the network will process 10,000 images in the data set; and reports the softmax outputs for each image to PC through UART interface. The PC will compare the received softmax outputs with those from a fault-free accelerator. With errors on weights, the PC will calculate the accuracy of the network by comparing the labels corresponding to the maximum score with the expected ones. For configuration memory, since fault on some bits may not affect the network processing, the PC will first identify the critical bits on which faults will change the softmax outputs, and further calculate the accuracy of the network for each critical bit.

As we will see later, the most time-consuming task is to identify critical bits from the hundreds of thousands configuration memory bits and to test the classification accuracy of each network with faults on each critical bit. This process may take more than three years to finish the experiments with a single Pynq-Z2 board. To speed up the experiments, a large-scale fault injection platform with 56 boards was built. The same bitstream runs on all the boards for a CNN, and each board is responsible for the fault injection experiments for part of the fault bits in the list. Then the total testing time is decreased to one month. The whole fault injection system is shown in Fig. 7, where 56 Pynq-Z2 boards are connected to the PC through three 20-port UART hubs.

4.2. Reliability evaluation for faults on weights

The user memory in the CNN accelerator is used for feature maps and parameters, including weights, bias, and BN factors. Based on [24], feature maps would be 50× more tolerant to faults than the weights. In addition, our initial work [30] showed that the effect of faults on bias and normalization parameters does not change for different pruning rates. Therefore, we mainly considered faults on weights in the user memory in this work.

In harsh environments like space, the bit error rate (BER) for unit memory size is usually fixed, so more bits would be corrupted for a larger number of weights. Following the same approach in [28], the total number of faulty bits on weights (N_{wb}) is calculated as the product of the number of weights (N_w) and the quantization size (N_b), and the number of bit upsets for each injection is calculated as $N_{fi} = N_{wb} * \text{BER}$. To make a fair comparison, the reliability of VGGs with different pruning rates is evaluated under the same BER. Since the effect of quantization schemes is independent of the pruning method, it is only evaluated for the magnitude based pruning method. Then reliability comparison of networks with different pruning rates under the same BER and the effect of faults on weights for different layers is evaluated for both pruning methods.

4.2.1. Reliability of pruned VGGs with different quantization schemes (magnitude-based pruning)

Considering the BERs used in [22–24,29], we choose BERs between 10^{-7} and 10^{-4} for the evaluation in this part. In the test, faults are randomly injected on weights from all layers. For each BER, 500 random fault injections are performed, and the accuracy of each injection is averaged as the final accuracy measure. The results for weight quantization of 6-bit, 8-bit, 12-bit and 16-bit are shown in Figs. 8–11, respectively. As we can see, the networks with higher pruning rates achieve higher accuracy for all quantization sizes, which is consistent with the results for floating-point implementation [30]. In addition, the reliability is generally improved for larger quantization sizes. In particular, the reliability of all networks is obviously improved when the quantization size is increased from 6 to 8, and the improvement for 12-bit and 16-bit quantization is different for networks with different pruning rates. The reliability improvement for the original VGG16 and VGG-70% is very limited when the quantization bit number is larger

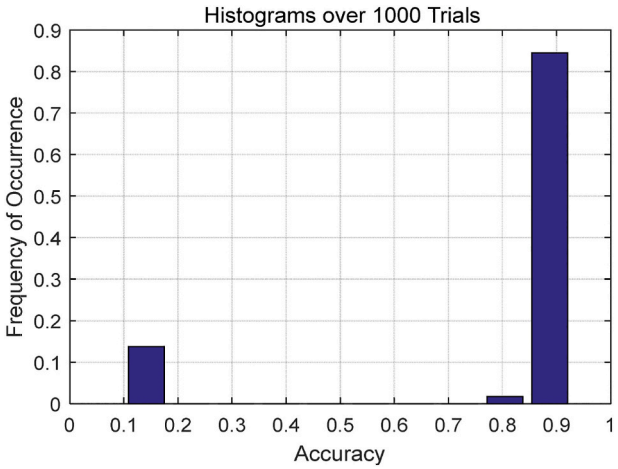


Fig. 7. Fault injection platform with 56 Pynq-Z2 boards.

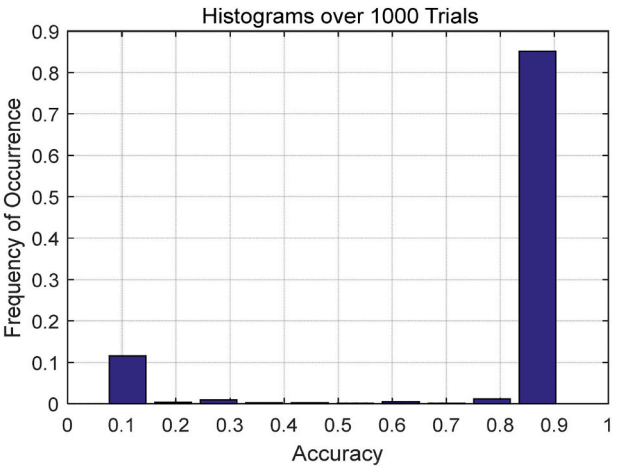


Fig. 8. Average accuracy with errors on weights (6-bit).

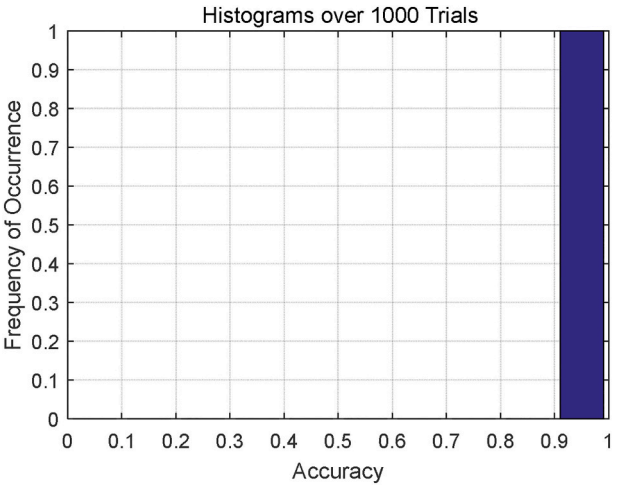


Fig. 9. Average accuracy with errors on weights (8-bit).

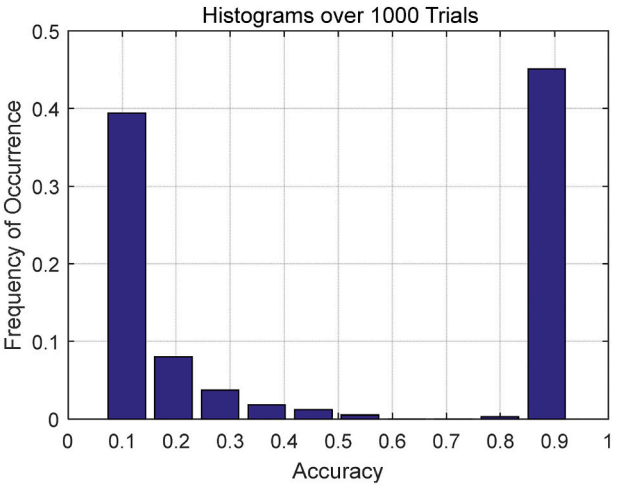


Fig. 10. Average accuracy with errors on weights (12-bit).

than 8, but that for less pruned network is still obvious. This phenomenon can be explained based on the conclusion from [26,28] that the performance degradation of the network is mainly caused by the faults on the significant bits of weights. Although the total number of bits for

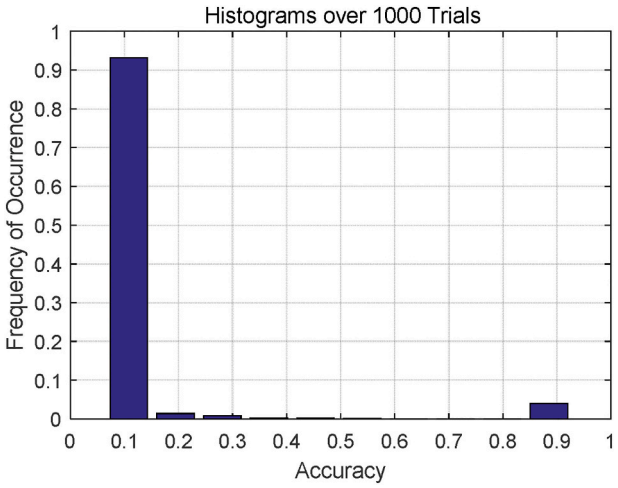


Fig. 11. Average accuracy with errors on weights (16-bit).

quantization size of 16 is twice of that for quantization size of 8, the number of significant bits is not increased so much. In other words, although more errors are injected for larger quantization sizes than for smaller quantization sizes, the portion of corrupted significant bits actually decreases. The testing results imply that the reliability improvement caused by a smaller portion of significant bits is slightly larger than the reliability degradation by a larger number of faults.

4.2.2. Reliability of VGGs with different pruning rates for filter pruning methods

For the filter pruning method, the accuracy of the network with pruning rates of 20%, 50% and 80% are compared with BER between 10^{-6} and 10^{-4}, and the results are shown in Fig. 12. Same to that for the magnitude-based pruning, the network with higher pruning rate achieves higher accuracy. In addition, by comparing the performance of the 8-bit implementations for two pruning methods (Figs. 9 and 12), we find that the networks using filter pruning method are less reliable than that using magnitude-based pruning. This is because the removed filters are all from the non-sensitive layers for the filter pruning method. In other words, more weights from sensitive layers are kept by the filter pruning, which are also sensitive to SEUs.

4.2.3. Effect of errors on weights for different layers

For this test, the 8-bit implementations for two pruning methods are evaluated under BER of $3 * 10^{-4}$. We inject the same portion of errors to

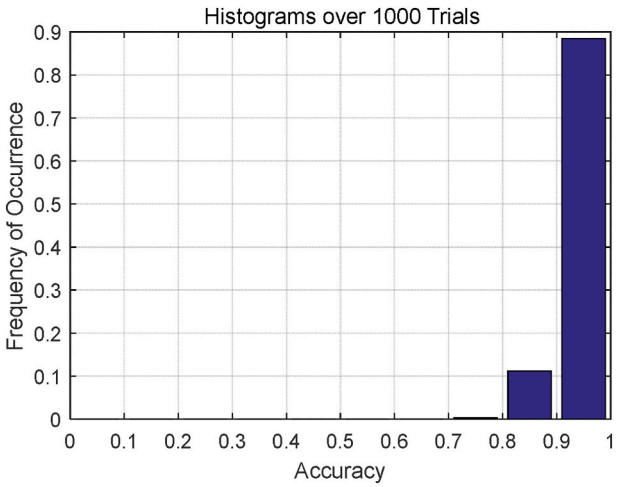


Fig. 12. Average accuracy with errors on weights (filter pruning).

different layers of each network and compare the average classification accuracy over 500 injections. The results for the two pruning methods are shown in Figs. 13 and 14, respectively. Two common conclusions could be obtained from the results. One is that the network with higher pruning rates is more reliable. The other is that the 8th–13th convolution layers are more vulnerable in general. For the magnitude-based pruning method, VGG-70% and VGG-50% do not degrade by the weights errors. But when the pruning rate decreases to 10%, the same portion of errors would cause more severe performance degradation if they occur on the 8th–13th convolution layers. This trend also holds for the filter pruning method, but it is obvious that the performance degradation by errors on the 8th–13th convolution layers is more severe than that for the magnitude-based pruning method. This is because only important weights are kept in these non-sensitive layers for the filter pruning method, so errors on them are more likely to change the classification results. This is consistent with the results in the previous subsection.

4.3. Reliability evaluation for faults on configuration memories

4.3.1. Percentage of critical bits

Based on the investigation of [20], faults on the configuration memory of CNN accelerators can cause both accuracy loss and system exceptions. The former is mainly caused by faults on PE array and latter is mainly caused by faults on control and instructions logic. In our case, since the original VGG16 and pruned ones are implemented using the same accelerator with the same instructions, system exceptions should be the same for all the networks, so we focus on the accuracy loss due to fault on the configuration memories of the PE array that account for the majority of the bits. We applied the same approach used in [35–37] that only a single bit of the configuration memory is corrupted for each injection (SEU). Quantization of weights is 8-bit for the testing in this subsection.

Based on the essential bit file (.ebd) generated during the compilation in Vivado, the number of essential bits for each PE is 365,349. Based on the fault injection experiments, SEUs on about 30% of the essential bits will affect the CNN processing (critical bits), and there are slight differences among the number of critical bits for different pruning rates and methods (115,438 ±0.1%). This is expected because all the networks are implemented based on the same hardware shown in Fig. 1. However, the effect of SEUs on the critical bits is different for different pruning rates and different pruning methods.

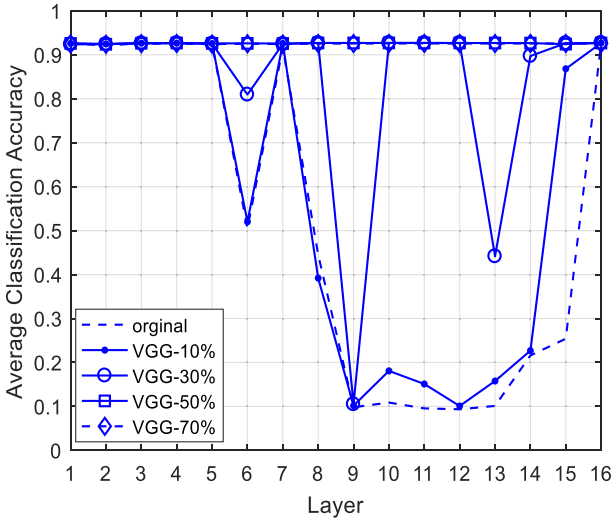


Fig. 13. Average accuracy with errors on weights for different layers (magnitude-based pruning).

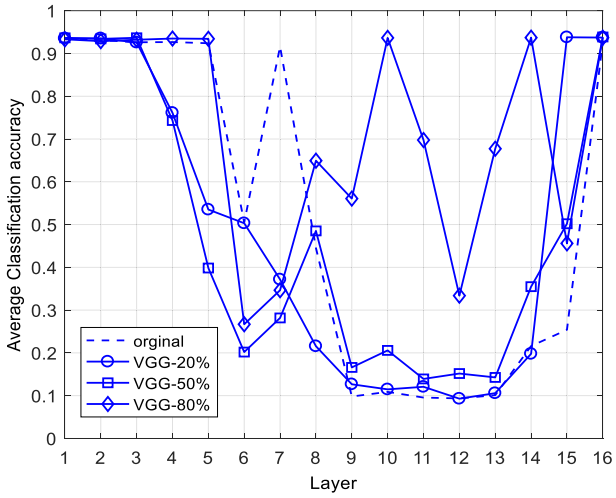


Fig. 14. Average accuracy with errors on weights for different layers (filter pruning).

4.3.2. Effect of SEU on critical bits for magnitude-based pruning

For the magnitude-based pruning method, the Probability Density Function (PDF) curves for the accuracy with SEUs on each of the critical bits for different pruning rates is shown in Fig. 15, based on which three conclusions can be obtained. First, over 70% of the critical bits will not degrade the network accuracy much (accuracy $>$ 90%, robust bits), and only less than 14% of the critical bits would cause poor results (accuracy $<$ 15%, vulnerable bits). Second, the percentage of vulnerable bits does not change for different pruning rates. This expected because the vulnerable bits are those related to the configuration logic or the common data access logic for the DSP arrays and adder tree, which has nothing to do with pruning. Third, the percentage of robust bits of the pruned networks is smaller than that of the original VGG16, and decreases for higher pruning rates, *which means the network with more weights pruned is less reliable*.

To further reveal the reason for the third conclusion, we evaluate the network accuracy with SEUs on the critical bits for the DSP array and adder trees separately, and the PDF curves of the accuracy for pruning rates of 30% and 70% are compared in Fig. 16. As expected, the portion of vulnerable bits for the adder tree and DSPs for VGG-30% are about 16.1% and 9%, respectively, and these portions do not change for VGG-

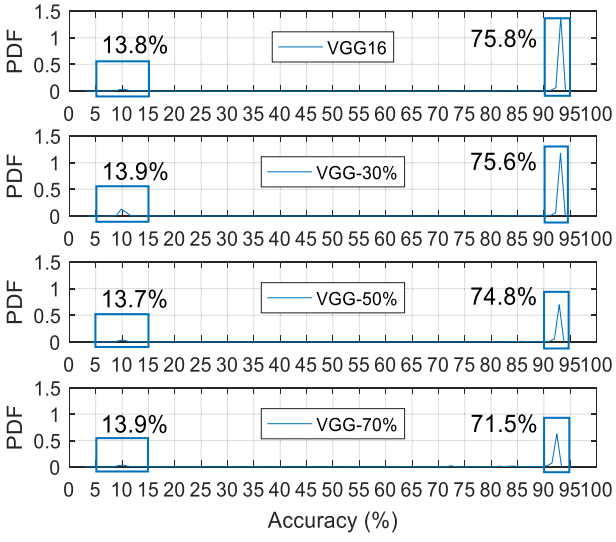


Fig. 15. PDF of accuracy for pruned networks with faults on critical bits (magnitude-based pruning).

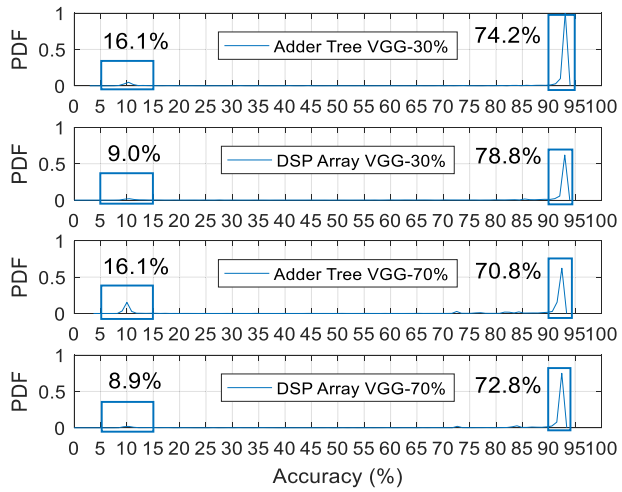


Fig. 16. PDF of accuracy for VGG-30%/70% with faults on critical bits for Adder Tree and DSP Array (magnitude-based pruning).

70%. However, compared with VGG-30%, the portion of robust bits for adder tree and DSPs in VGG-70% decrease by 3.4% and 6%, respectively. This can be explained as follows. Since the values of the pruned weights are set to be 0, more inputs of the adder tree would be 0 for higher pruning rates. Then a change on the 0 input by SEUs in the adder would have larger influence on the accumulation results when less non-zero inputs exists (more 0 inputs for higher pruning rates). This effect would be amplified by the multiplication of a faulty DSP, so the portion of robust bits decreases more by SEUs on DSP for higher pruning rates.

4.3.3. *Effect of SEU on critical bits for filter pruning*

For the filter pruning method, the PDF curves for the accuracy with SEUs on each of the critical bits for different pruning rates is shown in Fig. 17. Similar to the case for magnitude-based method, robust bits dominate the critical bits, and the portion of vulnerable bits does not change for different pruning rates. But a key difference is that the portion of robust bits is even higher than that for the original VGG16, and almost does not change for higher pruning rates (slight decreasing for VGG-80%). In other words, *the pruned networks using filter pruning are more reliable than the original VGG16 and the reliability does not decrease when more weights are pruned*. To reveal the reason, the accuracy with SEUs on the critical bits for the DSP array and adder trees are evaluated separately, and the PDF curves of the accuracy for pruning rate of 20% and 80% are compared in Fig. 18. As expected, the portion of vulnerable bits does not change for different pruning rates. However, two key differences can be obtained by comparing with the results in Fig. 16. One is that both the portion of robust bits for the adder tree and DSP array are obviously higher than those for the case of magnitude-base pruning. The other is that, when the pruning rates increases from 20% to 80%, the portion of robust bits for DSP array almost keeps unchanged, and that for adder tree only drops 1%. These phenomena can be explained as follows. Compared with the original VGG16, the key difference for pruned networks is that less kernels are used for convolution in each layer, so the same hardware is less used, and fewer output feature maps are produced for each layer. This means that the same configuration error caused by SEU will have less impact on the final result, which explains the reliability improvement over the original VGG16. Based on the work schedule of the accelerator, the portion of convolution workload for single DSP is the same for different number kernels in a layer, so an SEU on it will introduce errors to the same portion of output feature maps pixel regardless of the number of output feature maps. That's why the reliability of DSP array does not change for different pruning rates. Finally, the adder tree is implemented based on LUTs, and an SEU on it will introduce errors to one value in the LUT, which will change the output feature map pixel if the corrupted value is hit. For higher pruning rates, the hit probability decreases due to fewer use times, but the effect would be more obvious once hit because the output changes is relatively more important due to the fewer number of output feature maps. This explains why the reliability of adder tree increases first when the pruning rates increases from 0% (the original VGG16) to 50%, and decreases slightly for VGG-80%.

5. Conclusions

In this paper, we implemented the original VGG16 and pruned versions with different pruning rates using two pruning methods on an ISA-based FPGA accelerator and studied the impact of pruning rate on the network reliability to errors on weights and configuration memories for the processing element array based on hardware fault injection experiments. Some conclusions are obtained based on the experimental results. First, networks with higher pruning rates are more robust to errors on weights, but the reliability improvement for pruned networks using filter pruning is relatively smaller because the weights kept are more vulnerable to errors. Second, errors on about 30% of the configuration bits will affect the CNN operation, and only 14% of them will introduce significant accuracy loss. These percentages are the same for the original

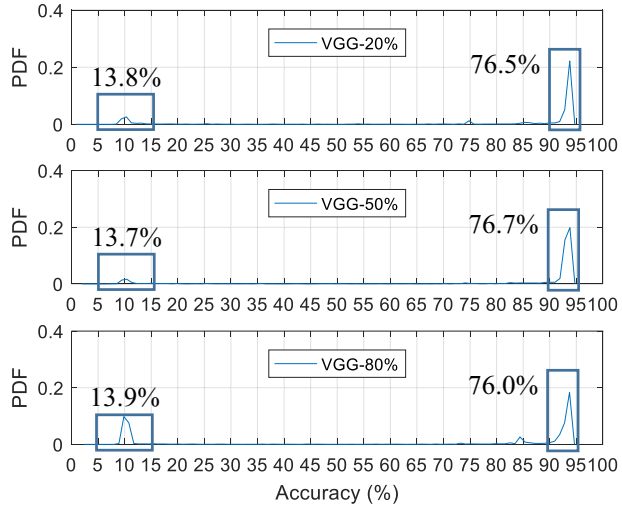


Fig. 17. PDF of accuracy for pruned networks with faults on critical bits (filter pruning).

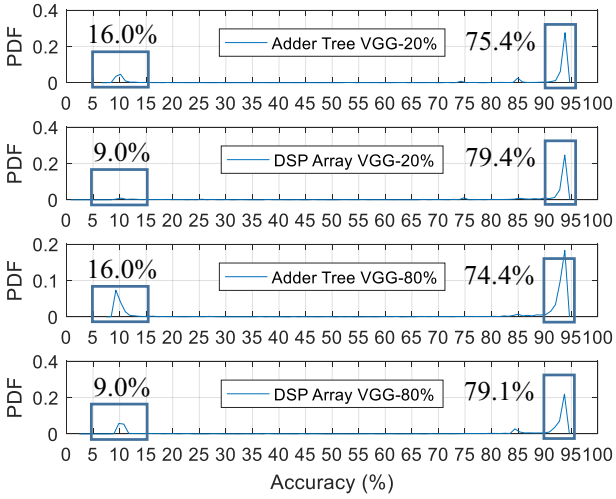


Fig. 18. PDF of accuracy for VGG-20%/80% with faults on critical bits for Adder Tree and DSP Array (filter pruning).

VGG16 and the pruned ones using two pruning methods. Third, for the magnitude-based pruning, higher pruning rates decrease the reliability of the network to errors on configuration bits because errors on the operation with 0 inputs has larger impact to the result. Fourth, for the filter pruning, pruned network is more reliable to errors on configuration bits than the original VGG16 because the same hardware is less used due to the fewer number of kernels so that the same error has less impact on the final results. In addition, the evaluation on the pruned networks using magnitude-based pruning also shows that larger quantization size improves the network reliability, but the improvement for networks with large pruning rates is limited for quantization sizes larger than 8.

CRediT authorship contribution statement

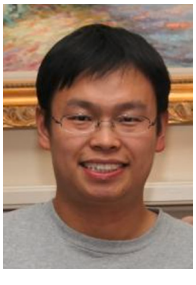
Zhen Gao: Methodology, analysis, draft
Yi Yao: Fault injection experiments for errors on configuration memory
Xiaohui Wei: Fault injection experiments for errors on weight
Tong Yan: Setup of FPGA based fault injection platform
Shulin Zeng: Support for FPGA accelerators
Guangjun Ge: Support for FPGA accelerators
Yu Wang: Support for FPGA accelerators, approach
Anees Ullah: Support for FPGA based fault injection platform
Pedro Reviriego: Writing and analysis.

Declaration of competing interest

The authors declared that they have no conflicts of interest to this work.

References

[1] A. Krizhevsky, I. Sutskever, G. Hinton, ImageNet classification with deep convolutional neural networks, in: International Conference on Neural Information Processing Systems (NIPS), 2012, pp. 1097–1105.
[2] Y. Taigman, M. Yang, M. Ranzato, Web-scale training for face identification, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 2746–2754.
[3] S. Han, H. Mao, W.J. Dally, Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding, in: International Conference on Learning Representations, 2016.
[4] A. See, M.T. Luong, C.D. Manning, Compression of neural machine translation models via pruning, in: SIGNLL Conference on Computational Natural Language Learning (CoNLL), 2016.
[5] H. Li, A. Kadav, I. Durdanovic, Pruning Filters for Efficient ConvNets, ICLR, 2017.
[6] Y. He, X. Dong, G. Kang, et al., Asymptotic soft filter pruning for deep convolutional neural networks, IEEE Trans. Cybern. 50 (8) (Aug. 2020) 3594–3604.
[7] B. F. Goldstein S. Srinivasan D. Das , et al., "Reliability evaluation of compressed deep learning models," 2020 IEEE 11th Latin American Symposium on Circuits & Systems (LASCAS).
[8] K. Guo, L. Sui, J. Qiu, et al., Angel-eye: a complete design flow for mapping CNN onto embedded FPGA, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 37 (1) (January 2018).
[9] X. Lian, Z. Liu, Z. Song, et al., High-performance FPGA-based CNN accelerator with block-floating-point arithmetic, IEEE Trans. Very Large Scale Integr. Syst. 27 (8) (August 2019).
[10] Y. Yu, C. Wu, T. Zhao, OPU: an FPGA-based overlay processor for convolutional neural networks, IEEE Trans. Very Large Scale Integr. Syst. 28 (1) (Oct. 2019).
[11] F.L. Kastensmidt, L. Carro, R. Reis, Fault-tolerance Techniques for SRAM-based FPGAs, Springer, New Haven, 2006.
[12] P.S. Ostler, M.P. Caffrey, D.S. Gibelyou, SRAM FPGA reliability analysis for harsh radiation environments, IEEE Trans. Nucl. Sci. 56 (6) (Dec. 2009) 3519–3526.
[14] L. Kolodny, Japanese Authorities Decry Ongoing Robot Failures at Fukushima, Mar. 2017.
[15] D. Binder, E.C. Smith, A.B. Holman, Satellite anomalies from galactic cosmic rays, IEEE Trans. Nucl. Sci. 22 (6) (Dec. 1975) 2675–2680.
[16] P.E. Dodd, M.R. Shaneyfelt, J.R. Schwank, et al., Current and future challenges in radiation effects on CMOS electronics, IEEE Trans. Nucl. Sci. 57 (4) (2010) 1747–1763.
[17] I. C. Lopes F. Benevenuti F. L. Kastensmidt , et al., "Reliability analysis on case-study traffic sign convolutional neural network on APSoC," IEEE LATS 2018, Sao Paulo.
[18] D. Xu, Z. Zhu, C. Liu, Persistent fault analysis of neural networks on FPGA-based acceleration system, in: IEEE 31st International Conference on Application-specific Systems, Architectures and Processors (ASAP), 2020.
[19] N. Kanekawa, E.H. Ibe, Y.T. Suga, Dependability in Electronic Systems: Mitigation of Hardware Failures, Soft Errors, and Electro-magnetic Disturbances, Springer Verlag, New York, USA, 2010.
[20] A.P. Arechiga, J.A. Michaels, The effect of weight errors on neural networks, in: 8th IEEE Annual Computing and Communication Workshop and Conference, 2018.
[21] S. Kwon, K. Lee, Y. Kim, Measuring error-tolerance in SRAM architecture on hardware accelerated neural network, in: IEEE International Conference on Consumer Electronics Asia, 2016.
[22] B. Reagen, P. Whatmough, R. Adolf, Minerva: enabling low-power, highly-accurate deep neural network accelerators, in: 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), June 2016, pp. 267–278.
[23] E. Ozen, A. Orailoglu, Sanity-check: boosting the reliability of safety-critical deep neural network applications, in: 2019 IEEE 28th Asian Test Symposium (ATS), 2019, pp. 7–75.
[24] B. Reagen, U. Gupta, L. Pentecost, Ares: a framework for quantifying the resilience of deep neural networks, in: 55th ACM/ ESDA/IEEE Design Automation Conference (DAC), 2018, pp. 1–6.
[25] G. Li, H. Siva, S. Michael, Understanding error propagation in deep learning neural network (DNN) accelerators and applications, in: International Conference for High-Performance Computing, Networking, Storage and Analysis (SC), Denver. Colorado, 2017.
[26] A.P. Arechiga, J.A. Michaels, The robustness of modern deep learning architectures against single event upset errors, in: IEEE High Performance Extreme Computing Conference, 2018.
[27] L. Hoang, M.A. Hanif, M. Shafique, FT-ClipAct: resilience analysis of deep neural networks and improving their fault tolerance using clipped activation, in: Design, Automation and Test in Europe, 2020, pp. 1241–1246.
[28] A. Bosio, P. Bernardi, A. Ruospo, A reliability analysis of a deep neural network, in: 2019 IEEE Latin American Test Symposium (LATS), Santiago, Chile, March 2019.
[29] M. Sabbagh, C. Gongye, Y. Fei, Evaluating fault resiliency of compressed deep neural networks, in: IEEE International Conference on Embedded Software and Systems, 2019.
[30] Z. Gao, X. Wei, H. Zhang, Reliability evaluation of pruned neural networks against errors on parameters, in: 33rd IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, Oct. 2020.
[31] I.C. Lopes, F.L. Kastensmidt, A.A. Susin, SEU susceptibility analysis of a feed forward neural network implemented in a SRAM-based FPGA, in: 2017 18th IEEE Latin American Test Symposium (LATS), 2017.
[32] C. Israel, B. Fabio, L.K. Fernanda, Reliability analysis on case-study traffic sign convolutional neural network on APSoC, in: 2018 IEEE 19th Latin-American Test Symposium (LATS), 2018, pp. 1–6.
[33] F. Libano, B. Wilson, J. Anderson, et al., Selective hardening for neural networks in FPGAs, IEEE Trans. Nucl. Sci. 66 (1) (2019) 216–222.
[34] F. Libano, B. Wilson, M. Wirthlin, et al., Understanding the impact of quantization, accuracy, and radiation on the reliability of convolutional neural networks on FPGAs, IEEE Trans. Nucl. Sci. 67 (7) (July 2020) 1478–1484.
[35] Y. Xing, S. Liang, L. Sui, et al., DNNVM: end-to-end compiler leveraging heterogeneous optimizations on FPGA-based CNN accelerators, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 39 (10) (Oct. 2020) 2668–2681.
[36] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: The Computing Research Repository, 2014 abs/1409.1556.
[37] S. Ioffe, C. Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, International on Machine Learning. JMLR.org, 2015.
[38] https://github.com/AlumLuther/PruningFilters.
[39] A. Ullah, P. Reviriego, J.A. Maestro, An efficient methodology for on-Chip SEU injection in Flip-flops for xilinx FPGAs, IEEE Trans. Nucl. Sci. 65 (4) (April 2018) 989–996.



Zhen Gao (M'11), received the BS, MS and PhD degree in Electrical and Information Engineering from Tianjin University, China, in 2005, 2007 and 2011, respectively. During 2008.10–2010.11, he was a visiting scholar in GeorgiaTech, working on the design and implementation for OFDM based cooperative communications. During 2011.7–2014.12, he was a Postdoc researcher in the Wireless and Mobile Communication Research Center in Tsinghua University, China, working on mobile satellite communication and fault tolerant design for DSPs. Since 2014.12, he is an Associate Professor in Tianjin University. His focus now includes fault-tolerant DSPs design and Blockchain technologies.

Yi Yao received a BS from Tianjin University and a BEng from University of Edinburgh in Jul. 2020, is now a MS student in Tianjin University since Sep. 2020. His research involves SRAM-FPGA fault tolerance system design and analysis.

Xiaohui Wei received the BS degree from Northeast Agriculture University in July 2019, and is a Master student in Electrical and Information Engineering from Tianjin University since Sep. 2019. Her research focus now is reliability evaluation of neural networks on radiation effects.

Tong Yan received the BS degree from Changchun University of Science and Technology in July 2018, and is a Master student in Tianjin University since Sep. 2018. His research focus now is design and implementation of SRAM-FPGA based fault injection platform and radiation effects study for digital signal processing.

Shulin Zeng received his B.S. degree in electronic engineering department of Tsinghua University, Beijing, China, in 2014. He is currently pursuing his PhD degree in electronic engineering department of Tsinghua University. His research mainly focuses on software-hardware co-design for deep learning and virtualization in the cloud.

Guangjun Ge is currently working as a post doctor in the Department of Electronic Engineering, Tsinghua University, Beijing, China. He received his BS and PHD degrees from Beihang University (BUAA) in 2012 and Tsinghua University in 2018, respectively. His research interests include information theory, fault tolerance coding and NN FPGA accelerator.

Yu Wang (Senior Member, IEEE), received the BS degree and PhD degree (with honor) from Tsinghua University, Beijing, China, in 2002 and 2007, respectively. He is currently a tenured professor and chair with the Department of Electronic Engineering, Tsinghua University, China. His research interests include application specific hardware computing, parallel circuit analysis, and power/reliability aware system design methodology. He has authored and coauthored more than 250 papers in refereed journals and conferences. He has received Best Paper Award in ASPDAC 2019, FPGA 2017, NVMSA17, ISVLSI 2012, and Best Poster Award in HEART 2012 with nine Best Paper Nominations. He is a recipient of DAC Under-40 Innovator Award, in 2018. He served as TPC chair for ICFPT 2019, ISVLSI 2018, ICFPT 2011 and finance chair of ISLPED 2012–2016, and served as the program commit- tee member for leading conferences in EDA/FPGA area.

Anees Ullah received BSc and MSc Degrees in Electrical Engineering for University of Engineering and Technology, Peshawar in 2009 and 2011, respectively. He received a PhD degree in Computer Engineering from Politecnico di Torino, Italy in 2015. During 2016–2017, he worked as a Post-doc researcher in Universidad Antonio de Nebrija, Spain. Currently, he is working as Assistant Professor in department of Electronics Engineering, University of Engineering and Technology, Peshawar. His research interests include fault-tolerant digital systems design, fault injection in FPGAs, FPGA-based Ternary Content Addressable Memories (TCAMs) and Approximate Computing.

Pedro Reviriego (SM'15), received the MSc and PhD degrees in telecommunications engineering from the Technical University of Madrid, Madrid, Spain, in 1994 and 1997, respectively. From 1997 to 2000, he was an R&D Engineer with Teldat, Madrid, working on router implementation. In 2000, he joined Massana to work on the development of 1000BaseT transceivers. From 2004 to 2007, he was a Distinguished Member of Technical Staff with the LSI Corpo-ration, working on the development of Ethernet transceivers. From 2007 to 2018 he was with Universidad Antonio de Nebrija. He is currently with the Departamento de Ingeniería Telemática, Universidad Carlos III de Madrid.