To appear at the 23rd Design, Automation and Test in Europe (DATE 2020) in Grenoble, France.

FT-ClipAct: Resilience Analysis of Deep Neural Networks and Improving their Fault Tolerance using Clipped Activation

Le-Ha Hoang, Muhammad Abdullah Hanif, Muhammad Shafique
Technische Universität Wien (TU Wien), Vienna, Austria
{le-ha.hoang, muhammad.hanif, muhammad.shafique}@tuwien.ac.at

***Abstract*—Deep Neural Networks (DNNs) are widely being adopted for safety-critical applications, e.g., healthcare and autonomous driving. Inherently, they are considered to be highly error-tolerant. However, recent studies have shown that hardware faults that impact the parameters of a DNN (e.g., weights) can have drastic impacts on its classification accuracy. In this paper, we perform a comprehensive error resilience analysis of DNNs subjected to hardware faults (e.g., permanent faults) in the weight memory. The outcome of this analysis is leveraged to propose a novel error mitigation technique which squashes the high-intensity faulty activation values to alleviate their impact. We achieve this by replacing the unbounded activation functions with their clipped versions. We also present a method to systematically define the clipping values of the activation functions that result in increased resilience of the networks against faults. We evaluate our technique on the AlexNet and the VGG-16 DNNs trained for the CIFAR-10 dataset. The experimental results show that our mitigation technique significantly improves the resilience of the DNNs to faults. For example, the proposed technique offers on average 68.92% improvement in the classification accuracy of resilience-optimized VGG-16 model at 1×10^{-5} fault rate, when compared to the base network without any fault mitigation.**

***Index Terms*—DNN, Reliability, Resilience, Fault-Tolerance, System-Level Optimization, Error Mitigation, Machine Learning**

I. Introduction

Due to their state-of-the-art accuracy in various applications, Deep Neural Networks (DNNs) have become the primary choice for most of the machine learning-based applications [1], ranging from simpler ones like hand written digit recognition to complex safety-critical applications like autonomous driving. In general, DNNs require a significantly large number of parameters (as shown in Fig. 1a for prominent DNNs used for image classification) to generalize well for real-time scenarios and, therefore, are highly computation and memory intensive. To efficiently process data using these networks, specialized hardware accelerators are utilized which are built using smaller technology nodes, in order to achieve high power and performance efficiency [2], [3], [4]. Moreover, these accelerators make use of large on-chip and off-chip memories to store the parameters of the DNNs.

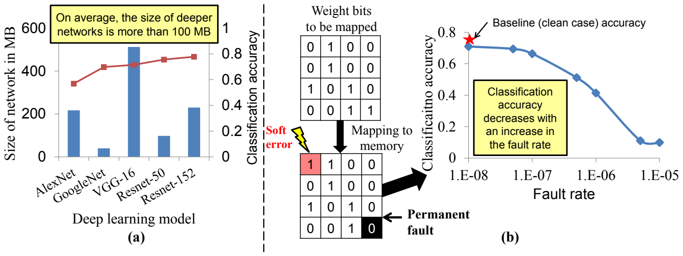


Figure 1: (a) Memory consumption of state-of-the-art DNN models. (b) The impact of hardware faults (bit flips in the weight memory) on the classification accuracy of AlexNet.

A major concern that DNN accelerators face in the nano-scale technologies is their reliability against faults, i.e, they suffer from faults due to soft errors, aging and manufacturing-induced defects [5], which can lead to catastrophic effects in case of their usage in safety-critical applications [6]. Fig. 1b illustrates our reliability analysis for the baseline AlexNet DNN (i.e. unprotected) [7] doing image classification on the CIFAR-10 dataset [8]. It can be noticed that the accuracy drops significantly with growing error rates.

Anecdotally, researchers speculated that DNNs forgive hardware errors [9]. But, our analysis (and other studies like [10]) has revealed that the accuracy drops even at low/nominal fault rates. In this paper, through a comprehensive analysis, we will show that it highly depends upon which weights are corrupted and if they belong to the sensitive neurons or not. *In short, there is a dire need for improving the resilience of these networks to provide reliable functionality when used with unreliable hardware having nominal fault rates.*

State-of-the-art and their Limitations: Various techniques have been proposed to mitigate the effects of hardware-level faults in DNN-based systems. *At hardware-level*, redundancy-based fault-mitigation techniques are commonly used, e.g., Dual Modular Redundancy (DMR) and Triple Modular Redundancy (TMR) [11] for mitigating faults in computational units, and Error Correction Codes (ECC) [12] for error-detection and correction in memories. In fact, the machine learning hardware in Tesla's self-driving cars uses expensive DMR to mitigate the impact of faults [13]. Note that, although these approaches offer improved resilience against faults, they have high overheads and are not preferable for computation/memory intensive DNNs. Other techniques include selective node hardening to improve the reliability of standard logic cells [14] and hardened SRAM-cells [15], [16]. *At software-level*, fault-aware training has been introduced for mitigating the memory faults [17], [18]. However, there are two drawbacks of these approaches: (1) they require access to the training dataset, which in several real-world scenarios may not be available for designing Inference Engines [1]; and (2) retraining costs a lot of resources and it may not be feasible to do it for every single chip. Moreover, such solutions are only limited to design-time faults, and cannot cope with run-time faults.

Targeted Research Problem: How to improve the resilience of the DNNs to hardware-level faults with minimal energy/power and performance overhead and without the need of training dataset, redundancy, or any costly reliability feature.

Our Novel Contributions: We address the above challenge through the following novel contributions:

- We perform a comprehensive analysis (Section III) to study the impacts of hardware-level faults on the accuracy and the intermediate outputs of the DNNs. This allows us to understand the resilience of DNNs in a systematic way, which can enable an efficient reliability mechanism.
- Based on the analysis, we propose a clipped activation function (Section IV) for improving the resilience of the DNNs, which bounds the intermediate output (i.e., activation) values of the networks to a defined range.
- We propose a systematic methodology (Section IV) to define the output range of the activation functions for each layer of a DNN without the need of the training dataset and without modifying the weights and biases of the network.
- We present a comprehensive evaluation of the effectiveness of our mitigation technique on the AlexNet and the VGG-16 networks. The evaluation shows 18.19% and 69.49% improvement in the classification accuracy of the AlexNet and the VGG-16 networks, respectively, at 5×10^{-7} fault rate compared to their baseline (without error mitigation) variants.

II. Background: An Overview of DNNs

A prominent type of DNNs is Convolutional Neural Networks (CNNs), which is used for processing spatially correlated data, e.g., images and videos. A CNN is mainly composed of two types of computational layers, i.e., convolutional (CONV) layers and fully-connected (FC) layers, where each computational layer is followed by an activation layer and each CONV layer is (optionally) followed by a pooling layer. Note that the FC layers are used for classification tasks and, therefore, are used towards the end of the CNNs while the CONV layers are used for extracting features and, therefore, are placed at the start and feeds the extracted features to the FC layers. A high-level view of the LeNet-5 network is shown in Fig. 2. The outputs of these layers are generated by the dot product operations between parameters and input values, which are then passed through activation functions, e.g. ReLU, to add non-linearity in the computations. The outputs from the activation functions are usually referred to as activations. A more comprehensive overview of the neural networks can be found in [19].

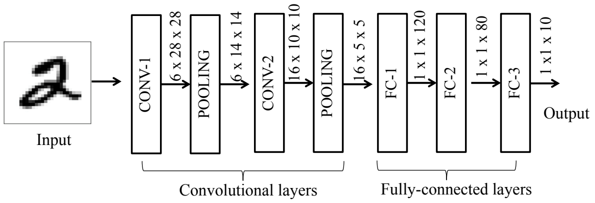


Figure 2: A high-level view of the LeNet-5 network

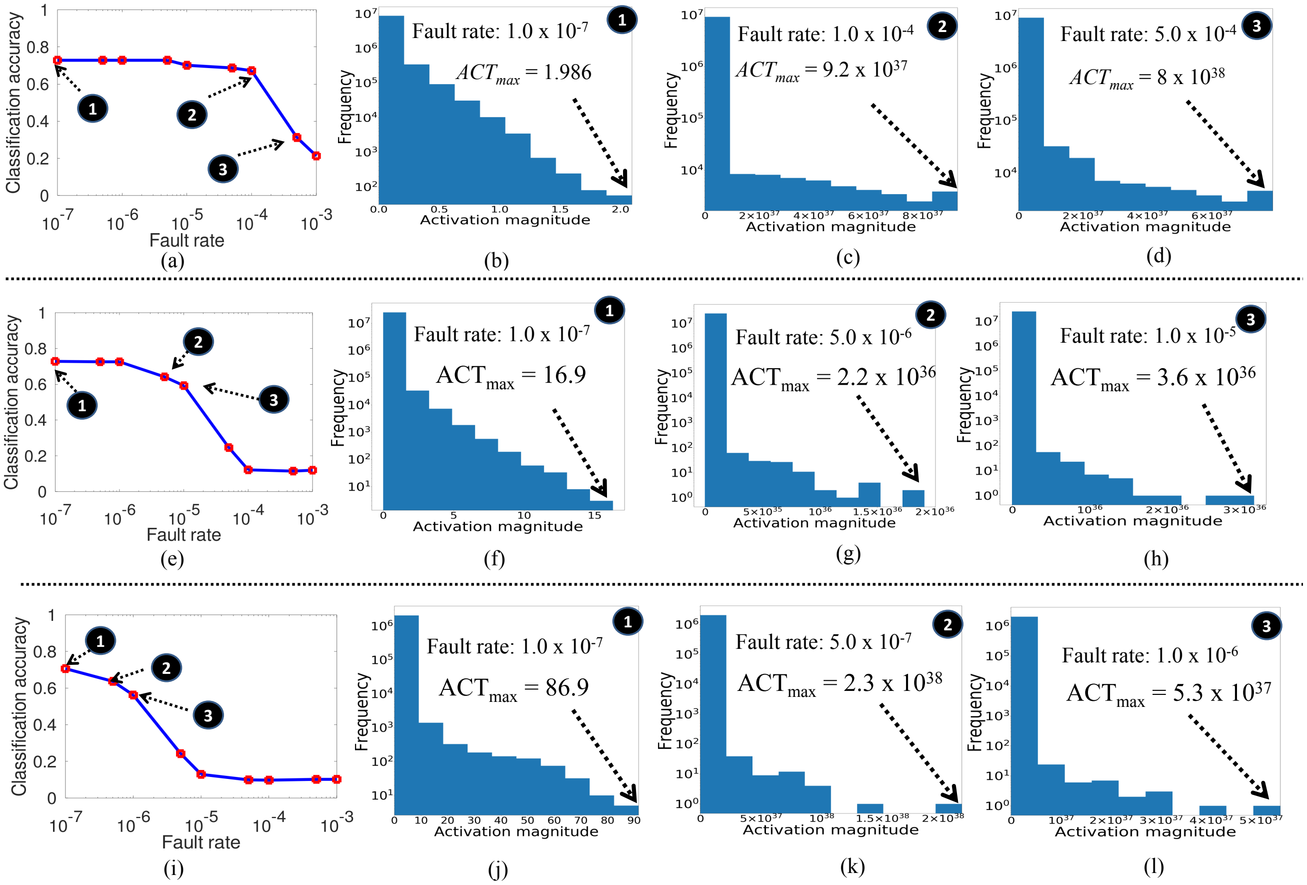
III. Error Resilience Analysis of Deep Neural Networks

To analyze the error resilience of a DNN against memory faults, we developed a fault-injection framework, where random bit-flips are injected in the memory blocks storing the parameters of the DNN model. We perform per-layer fault injection to study the sensitivity of individual layers and the effects of the faults on the output activations. Fig. 3 illustrates the resilience of CONV-1 layer (first computational layer), CONV-5 layer (fifth computational layer), and FC-1 layer (sixth computational layer) of the AlexNet. The figure also shows the distributions of the output activations of the respective layers.

From the analysis of Fig. 3, we draw the following key observations:

- In general, the classification accuracy of the network decreases with an increase in the fault rate, as shown in Figs. 3a, 3e, and 3i. Moreover, the decrease in the accuracy is monotonic, which is mainly because, at higher fault rates, the probability of a fault occurring at a critical location is significantly higher.
- At lower fault rates, the accuracy of the network stays close to the baseline accuracy before dropping drastically, as there is a significant chance that the faults do not occur at critical bit locations or are masked within the network. Also, the fault rate till which the accuracy stays close to the baseline accuracy is different for each layer. This is because each layer has different number of parameters and has different number of layers between the output and itself.
- The distribution of the output activations at higher fault rates have values of higher-intensities as well, as can be observed from Figs. 3c and 3d. This trend is consistent across layers, as can also be seen in Figs. 3g, 3h, 3k, and 3m. This is mainly because of the fact that the weights are distributed close to zero value and bit-flips from 0 to 1 at Most Significant Bit (MSB) locations of the weights can result in them having higher magnitudes and, thereby, resulting in high-intensity activations during inference.

[1] For example, consider a DNN IP provided by a service provider, which has to be deployed on a particular embedded hardware. The IP provider has not made the training dataset available (as training dataset is a key IP), and one of the system requirements is to have a defined-level of fault-tolerance which the network (when deployed on the embedded hardware) does not meet.



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k) (l)

Figure 3: Error resilience analysis of CONV-1 layer (a-d), CONV-5 layer (e-h), and FC-1 layer (i-l) of the AlexNet on the CIFAR-10 dataset

IV. Our Mitigation Technique for Improving Fault Tolerance of Deep Neural Networks

Fig. 4 shows an overview of our methodology for improving the fault tolerance of DNNs using *clipped activation functions*. The methodology is based on the observation made in Section III that higher fault rates result in faulty activations with higher magnitudes, which dominate the result and may lead to misclassification. The proposed methodology is independent of the training dataset and only requires a small subset of the validation set for tuning the clipping thresholds of the clipped activation functions. Our methodology operates in three main steps, as discussed below.

Step-1: We perform profiling for computing the statistical properties of the activations of all the layers using a subset of the validation dataset. The statistics extracted from this step are the maximum value of the activations (ACT_{max}) observed at the output of each layer.

Step-2: We replace the unbounded activation functions in the DNN with their clipped variants (explained in Section IV-A) and initialize their thresholds with their corresponding ACT_{max}.

Step-3: We perform fine-tuning of the clipping thresholds using an efficient method explained in Section IV-C. The metric used for resilience evaluation is presented in Section IV-B. Note that Step 3 is repeated for each layer of the network, using the network generated from Step 2, to find suitable clipping thresholds for all the layers. *The final outcome from the methodology is a fault-tolerant DNN with optimized thresholds for the clipped activation functions.*

A. The Clipped Activation Function

Based on the observations made in Section III and following an inspiration from the pruning [20] and the dropout [21] techniques, we introduce a novel *clipped version of the ReLU activation function* for mapping high-intensity (possibly faulty) activation values to zero. We formulate this function as:

$$f(x) = \begin{cases} x, \ if \ 0 \ \leq x \leq T \\ 0, \ otherwise \end{cases}$$

Where, $f(x)$ is the output activation, x is the input (i.e., output after dot-product operation), and T is the clipping threshold beyond which all the values are considered faulty and are mapped to zero. Although we present the clipped version of only the ReLU function, clipped versions of other

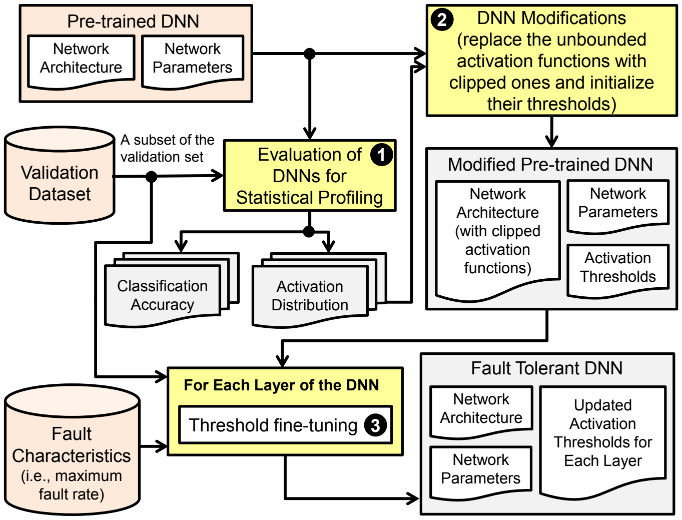

Pre-trained DNN
Network Architecture
Network Parameters
2 DNN Modifications (replace the unbounded activation functions with clipped ones and initialize their thresholds)
A subset of the validation set
Validation Dataset
Evaluation of 1 DNNs for Statistical Profiling
Modified Pre-trained DNN
Network Architecture (with clipped activation functions)
Network Parameters
Activation Thresholds
Classification Accuracy
Activation Distribution
Fault Tolerant DNN
For Each Layer of the DNN
Threshold fine-tuning 3
Fault Characteristics (i.e., maximum fault rate)
Network Architecture
Network Parameters
Updated Activation Thresholds for Each Layer

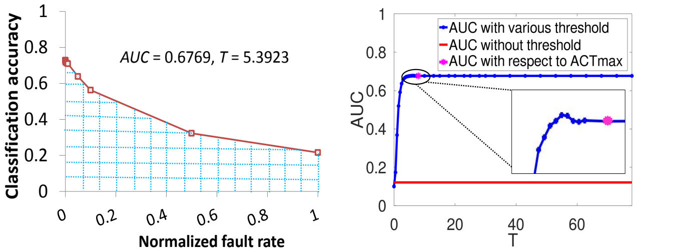
Figure 4: Our methodology to improve the resilience of a pre-trained DNN model

activation functions (e.g., Leaky-ReLU) can also be designed similarly.

B. Resilience Evaluation Metric and the Corresponding Analysis for Finding Suitable Clipping Thresholds

Evaluation metric: Hardware fault-rates can vary in a defined range in real scenarios. Therefore, to capture the resilience characteristics of a network across different fault rates in a single metric, we introduce the area under the accuracy vs. normalized fault rate curve (AUC) as a metric, where the area is computed using the Trapezoidal rule. An illustration of this is shown in Fig. 5a, where the area of the region marked with blue grid represents the AUC. Note that both the axes are normalized such that the ideal scenario, i.e., the case where the network provides 100% accuracy at all the considered fault rates, has an AUC of 1.

Resilience sweep across thresholds: To study the impact of threshold value of the clipped activation function of a layer of a network on the resilience of the network, let us consider the AUC vs. T curve of CONV-4 layer of the AlexNet network trained on the CIFAR-10 dataset. The plot is shown in Fig. 5b.



(a) AUC is the grid pattern showing the area under the classification accuracy curve

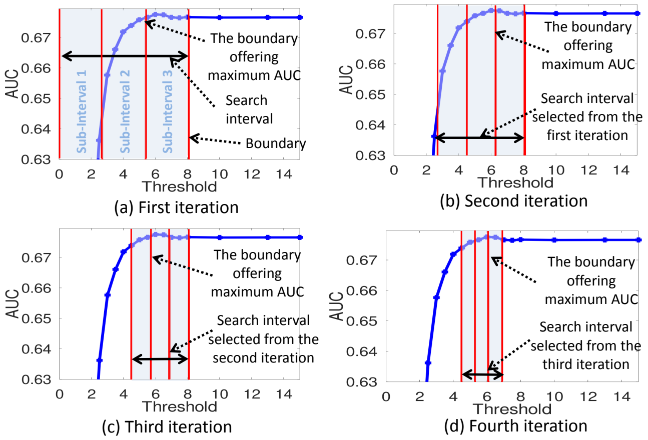
(b) The AUC vs. T curve of CONV-4 of the AlexNet trained on the CIFAR-10 dataset

Figure 5: (a) Illustration of the AUC calculation of a the AlexNet with clipped activation in the presence of faults in CONV-4 layer; (b) The AUC curve with various threshold (blue line) and AUC without threshold (red line)

As can be seen from the figure, moving from higher to lower threshold values, the AUC rises to a peak value at a particular location before decreasing drastically. Ideally, we should select the threshold at this location as the optimal clipping threshold ($T_{optimum}$) of the activation functions of the layer, as it offers the highest resilience within a pre-defined fault range. Note that, although the blue curve in the figure seems to have a fixed value at higher T values (i.e., at $T > ACT_{max}$), the AUC of the network with unbounded activations is significantly low, as shown with the help of red line in the figure. This also reaffirms the fact that clipping high-intensity activation values can significantly improve the overall resilience of a DNN.

C. Threshold Fine-Tuning Algorithm

The threshold fine-tuning algorithm is based on the observation made in the previous subsection that the AUC vs. T curves always have a bell shaped curve, as also shown in Fig. 5b. Another key observation which helped us in designing an efficient algorithm is that the peak of the curve always lie below the ACT_{max} value determined in Step 1 of the methodology. The algorithm starts by initializing search interval, i.e., $S = [0,\ ACT_{max}]$, and dividing it into three equally-sized sub-intervals, which is illustrated in Fig. 6a. The $AUC_i^{T_i}$ corresponding to the i^{th} boundary, at the threshold T_i in the current search interval, is computed for each $i \in \{1, 2, 3, 4\}$. The region ($\bar{S}$) covering the sub-interval/s around the boundary offering maximum AUC is selected while the rest are discarded. The search interval S is updated with $\bar{S}$ and then again divided into three equally spaced sub-intervals in the next iteration and the same process is repeated, as shown in Figs. 6b, 6c, and 6d. This process is applied until the number of iterations ($counter$) reaches a defined number (N), or the maximum difference between the adjacent $AUC_i^{T_i}$s (Δ_j, $1 \leq$ j$\leq$ 3) is less than a predefined limit (δ) and $counter \geq M$ ($M < N$). The detailed algorithm is shown in Algo. 1.



(a) First iteration

(b) Second iteration

(c) Third iteration

(d) Fourth iteration

Figure 6: Threshold Fine-Tuning Algorithm applied on CONV-4 layer of the AlexNet on CIFAR-10 dataset

Algorithm 1: Threshold Fine-Tunning

```
input : Modified pre-trained DNN model from Step 2 of the proposed
        methodology
output: $T$
BEGIN ALGORITHM
$counter \leftarrow 1$ ;
while $counter \leq N$ do
    if $i$ ==1 then
        S ← [0 , $ACT_{max}$] ;
        $AUC_1^{T_1}, AUC_2^{T_2}, AUC_3^{T_3}, AUC_4^{T_4}$ =
          $AUC\_Calculation(S)$ ;
    else
        S, T = Interval_Search( $T_1, T_2, T_3, T_4, AUC_1^{T_1}, AUC_2^{T_2},$
          $AUC_3^{T_3}, AUC_4^{T_4}$) ;
        $AUC_1^{T_1}, AUC_2^{T_2}, AUC_3^{T_3}, AUC_4^{T_4}$ = AUC_Calculation(S)
          ;
    counter ← counter+1 ;
    for $i$ = 1 to 3 do
        $\Delta_i \leftarrow | AUC_{i+1}^{T_{i+1}} - AUC_i^{T_i} |$;
    if maximum($\Delta_1$, $\Delta_2$, $\Delta_3$) $\leq \delta$ and counter $\geq M$ then
        Return T ;
Return T ;
END ALGORITHM
Function Interval_Search($T_1, T_2, T_3, T_4, AUC_1^{T_1}, AUC_2^{T_2},$
  $AUC_3^{T_3}, AUC_4^{T_4}$):
    Index ← index of T with the highest AUC
    if Index == 4 then
        $\bar{S} \leftarrow [T_3, T_4]$ ;
    else if Index == 1 then
        $\bar{S} \leftarrow [T_1 , T_2]$ ;
    else
        $\bar{S} \leftarrow [T_{Index-1}, T_{Index+1}]$ ;
    T ← $T_{Index}$ ;
    Return $\bar{S}$, T ;
Function AUC_Calculation($S$):
    $T_1$ ← minimum(S) ;
    $T_2$ ← $T_1$ + (maximum(S) - minimum(S))/3 ;
    $T_3$ ← $T_2$ + (maximum(S) - minimum(S))/3 ;
    $T_4$ ← maximum(S);
    for $i$ = 1 to 4 do
        Evaluating model using $T_i$;
        Calculating $AUC_i^{T_i}$;
    Return $AUC_1^{T_1}, AUC_2^{T_2}, AUC_3^{T_3}, AUC_4^{T_4}$;
```

V. Results And Discussion

A. Experimental setup

We evaluated our proposed mitigation technique on two DNNs models, i.e., the AlexNet and the VGG-16 [22]. Both the models are modified to take the CIFAR-10 dataset images as inputs. The AlexNet contains 5 CONV layer and 3 FC layer while the base VGG-16 contains 13 CONV layer and 1 FC layer. The AlexNet and the VGG-16 offer baseline classification accuracies of 72.8% and 82.8%, respectively.

We developed our fault injection framework in Python using the Pytorch framework [23]. The developed framework is in-line with other fault injection frameworks proposed in state-of-the-art works, e.g., Ares in [24]. All experiments are performed on an Intel Core i7@3.2 GHz processor with two NVIDIA GeForce GTX 1080 Ti GPUs.

B. Comparison with the unprotected DNNs

To show the effectiveness of the proposed methodology, we compared the accuracy of the resilient DNNs, developed using the proposed method, with unprotected DNNs. Fig. 7a shows the classification accuracies of the resilient and the unprotected AlexNet. The figure clearly illustrates that the network with clipped activation functions shows significant improvements in the fault-resilience of the DNN at fault rates around 1×10^{-7} and 1×10^{-6}. For example, the classification accuracy of the resilient AlexNet with clipped activations at 5×10^{-7} fault rate is 69.36% compared to 51.16% observed for the unprotected DNN. Overall, the proposed method shows 173.32% improvement in the AUC of the AlexNet considering the fault range from 0 to 1×10^{-5}. Note that the accuracies reported in Fig. 7a are mean values computed using 50 experiments, which is already large considering highly compute-intensive nature of DNNs and their multiple execution runs and parameter settings.

Figs. 7b and 7c show the variations across multiple experiments using box plot. Note that at fault rates 1×10^{-8} and 5×10^{-8} the worst-case accuracy of the resilient network, generated using the proposed methodology, is close to the baseline accuracy (i.e., 72.8%) while the worst-case accuracy of the unprotected network for the same fault rates is 41.93% and 13.66%, respectively, i.e., significantly lower than the baseline.

Similar trend is observed in case of the VGG-16 network, as shown in Fig. 8. However, the proposed technique shows significant improvements in the resilience of the network, e.g., 654.91% at fault rate 5×10^{-7} in AUC as can be observed from Fig. 8a, even better than the case of the AlexNet network.

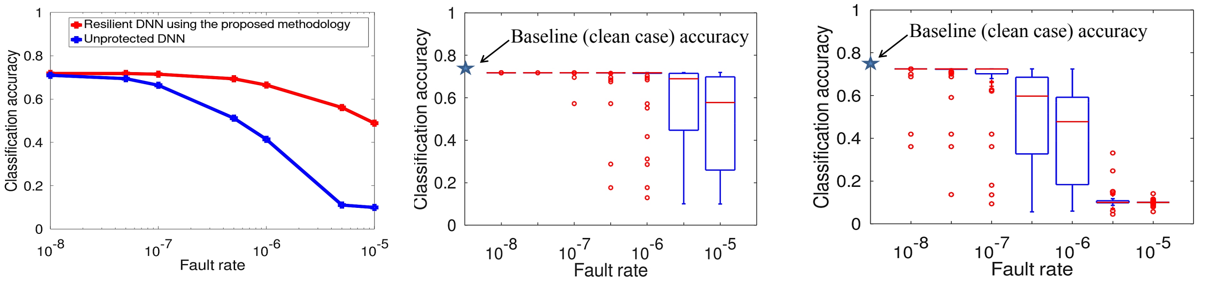
Note that for all the results reported in Figs. 7 and 8, we employed the CIFAR-10 test set in order to avoid any overlap between the data used for testing and the data used for computing the thresholds.

VI. Conclusion

In this work, we presented an analysis to study the impact of hardware faults on the accuracy and the intermediate outputs of the DNNs. We analyzed how high-intensity activations, generated due to the parameter corruption, result in the degradation of the accuracy of DNN models. To mitigate the effects of faults, we proposed a technique based on clipped activation functions, which blocks the high-intensity (potentially faulty) activations and maps them to zero. We also proposed an efficient algorithm for defining the range of the clipped activation functions. The proposed technique offers a significant improvement in the resilience of the DNNs. For example, the proposed technique provides 68.92% improvement at 10^{-5} fault rate for the VGG-16 network trained on the CIFAR-10 dataset, when compared to the unprotected network.

Acknowledgement

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)

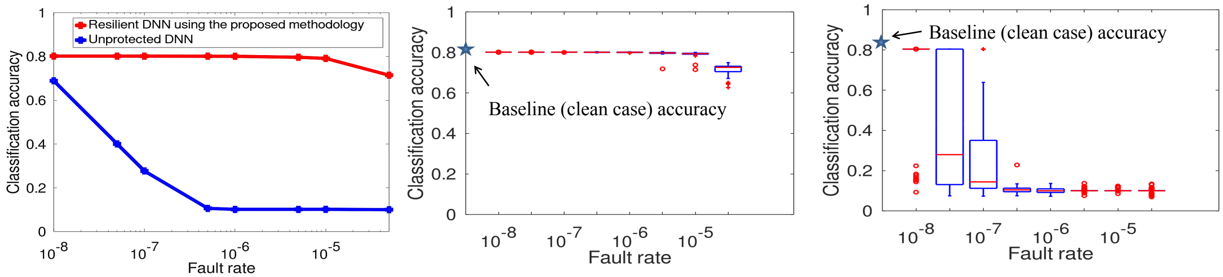


(a) Average classification accuracy at various fault rats of the DNN using clipped activation functions and unprotected DNN

(b) Classification accuracy distribution at various fault rates of the DNN using clipped activation functions

(c) Classification accuracy distribution at various fault rates of the unprotected DNN

Figure 7: Error resilience evaluation of the AlexNet with and without clipped activation functions



(a) Average classification accuracy at various fault rates of the DNN using clipped activation functions and the unprotected DNN

(b) Classification accuracy distribution at various fault rates of the DNN using clipped activation functions

(c) Classification accuracy distribution at various fault rates of the unprotected DNN

Figure 8: Error resilience evaluation of the VGG-16 with and without clipped activation functions

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[2] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *ISCA*, 2017.

[3] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3. IEEE Press, 2016, pp. 367–379.

[4] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: Efficient inference engine on compressed deep neural network," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, June 2016, pp. 243–254.

[5] C. Constantinescu, "Trends and challenges in vlsi circuit reliability," *IEEE Micro*, vol. 23, no. 4, pp. 14–19, July 2003.

[6] "Road vehicles — functional safety," Tech. Rep., 2016.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[8] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[9] Q. Zhang, T. Wang, Y. Tian, F. Yuan, and Q. Xu, "Approxann: An approximate computing framework for artificial neural network," in *Proceedings of the 2015 DATE*. EDA Consortium, 2015, pp. 701–706.

[10] J. J. Zhang, K. Liu, F. Khalid, M. A. Hanif, S. Rehman, T. Theocharides, A. Artussi, M. Shafique, and S. Garg, "Building robust machine learning systems: Current progress, research challenges, and opportunities," in *Proceedings of the 56th DAC*, 2019.

[11] R. E. Lyons and W. Vanderkulk, "The use of triple-modular redundancy to improve computer reliability," *IBM journal of research and development*, vol. 6, no. 2, pp. 200–209, 1962.

[12] K. Furutani, K. Arimoto, H. Miyamoto, T. Kobayashi, K. Yasuda, and K. Mashiko, "A built-in hamming code ecc circuit for drams," *IEEE Journal of Solid-State Circuits*, vol. 24, no. 1, pp. 50–56, Feb 1989.

[13] "meet tesla's self-driving car computer and its two ai brains research," Tech. Rep., 2019.

[14] D. B. Limbrick, N. N. Mahatme, W. H. Robinson, and B. L. Bhuva, "Reliability-aware synthesis of combinational logic with minimal performance penalty," *IEEE Transactions on nuclear science*, vol. 60, no. 4, pp. 2776–2781, 2013.

[15] A. Azizimazreah, Y. Gu, X. Gu, and L. Chen, "Tolerating soft errors in deep learning accelerators with reliable on-chip memory designs," in *2018 IEEE International Conference on NAS*. IEEE, 2018, pp. 1–10.

[16] J. Guo, L. Xiao, and Z. Mao, "Novel low-power and highly reliable radiation hardened memory cell for 65 nm cmos technology," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 7, pp. 1994–2001, 2014.

[17] S. Kim, P. Howe, T. Moreau, A. Alaghi, L. Ceze, and V. Sathe, "Matic: Learning around errors for efficient low-voltage neural network accelerators," in *DATE*, March 2018, pp. 1–6.

[18] L. Xia, Mengyun Liu, Xuefei Ning, K. Chakrabarty, and Yu Wang, "Fault-tolerant training with on-line fault detection for rram-based neural computing systems," in *54th DAC*, June 2017.

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

[20] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[23] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[24] B. Reagen, U. Gupta, L. Pentecost, P. Whatmough, S. K. Lee, N. Mulholland, D. Brooks, and G.-Y. Wei, "Ares: A framework for quantifying the resilience of deep neural networks," in *Proceedings of the 55th DAC*, 2018.