Imperceptible misclassification attack on deep learning accelerator by glitch injection

Liu, Wenye; Chang, Chip-Hong; Zhang, Fan; Lou, Xiaoxuan

2020

Liu, W., Chang, C.-H., Zhang, F., & Lou, X. (2020). Imperceptible misclassification attack on deep learning accelerator by glitch injection. Proceedings of the 2020 57th ACM/IEEE Design Automation Conference (DAC). doi:10.1109/DAC18072.2020.9218577

<https://hdl.handle.net/10356/145856>

<https://doi.org/10.1109/DAC18072.2020.9218577>

© 2020 Association for Computing Machinery (ACM). All rights reserved. This paper was published in 2020 57th ACM/IEEE Design Automation Conference (DAC) and is made available with permission of Association for Computing Machinery (ACM).

Downloaded on 26 Apr 2023 19:16:42 SGT

Imperceptible Misclassification Attack on Deep Learning Accelerator by Glitch Injection

Wenye Liu*, Chip-Hong Chang*, Fan Zhang[†] and Xiaoxuan Lou[†]

*School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore

[†]College of Computer Science and Technology, Zhejiang University, China

Email: echchang@ntu.edu.sg*, fanzhang@zju.edu.cn[†]

Abstract—The convergence of edge computing and deep learning empowers endpoint hardware or edge devices to perform inferences locally with the help of deep neural network (DNN) accelerator. This trend of edge intelligence invites new attack vectors, which are methodologically different from the well-known software oriented deep learning attacks like the input of adversarial examples. Current studies of threats on DNN hardware focus mainly on model parameters interpolation. Such kind of manipulation is not stealthy as it will leave non-erasable traces or create conspicuous output patterns. In this paper, we present and investigate an imperceptible misclassification attack on DNN hardware by introducing infrequent instantaneous glitches into the clock signal. Comparing with falsifying model parameters by permanent faults, corruption of targeted intermediate results of convolution layer(s) by disrupting associated computations intermittently leaves no trace. We demonstrated our attack on nine state-of-the-art ImageNet models running on Xilinx FPGA based deep learning accelerator. With no knowledge about the models, our attack can achieve over 98% misclassification on 8 out of 9 models with only 10% glitches launched into the computation clock cycles. Given the model details and inputs, all the test images applied to ResNet50 can be successfully misclassified with no more than 1.7% glitch injection.

I. INTRODUCTION

Deep learning empowers a great diversity of edge applications such as virtual personal assistant, video surveillance and autonomous vehicle. To unlock deep learning potential on endpoint devices, many efficient hardware deep neural network (DNN) accelerators have been developed [1]. Extensive use of deep learning accelerator raises concerns on its security and robustness especially for safety and life-critical applications. A conspicuous vulnerability of DNN has been exposed by adversarial examples [2]. An aggressor can subvert the output of a well-trained classifier by adding carefully crafted perturbations on the input [3] [4]. On the premise that any input pixel can be precisely altered to any arbitrary value in adversarial examples, the attack is unlikely to achieve the desired outcomes when the adversarial examples are presented to a DNN accelerator.

Fault injection attack is the primary method to fail the classifier of a DNN hardware accelerator. Various fault attacks such as laser injection [5], hardware Trojan [6], glitch disturbance [7] and rowhammering [8] can impact circuit operations within the DNN. Straightforward fault injection will cause a denial of service which alerts attention to trigger immediate damage control to limit the benefits that can be reaped from a successful attack. For instance, overheating the DNN hardware will not only affect classification but also suspend the system.

Most existing fault attacks on deep learning focus on model weight manipulations [9] [10] [11]. However, falsification of model parameters will leave footprints in memory regardless of how much data has been manipulated. In consequence, such persistent fault induction in the weights can be directly detected by model readback and bypassed by parameter reloading.

Inconspicuous attack on dedicated deep learning accelerators is seldom studied. Interpolation on model parameters, including saturating the last layer's bias [9], minimizing the modification of model weights [10], rowhammering [11] and memory collisions [12], can induce the DNN into incorrect predictions. Unfortunately, even a single parameter change will leave some traces in the memory which can be detected by comparing with the primitive model. Hardware Trojan is another threat on DNN accelerators [6] [13] [14]. Besides, a laser beam based fault attack on embedded DNN has been reported recently [5]. To the best of our knowledge, existing hardware attacks on edge deep learning applications are constrained to DNN hardware on small scale (10 categories) classification [5] [12] [14] [15] or are based on simulated instead of physically induced faults [6] [10] on ImageNet [16] (1000 categories) classification.

This paper presents the first stealthy misclassification attack on deep learning accelerator for ImageNet applications. Well-trained ImageNet models are the backbone of advanced classification tasks such as object detection [17] and image segmentation [18]. Our attack aims at inducing temporal fault into intermediate results of the convolutional layer. These temporary perturbed data will propagate to the inference stage but they will be overwritten by the correct data after each prediction, leaving no trace for detection. To tamper specific computation output, low cost and yet effective clock glitches are injected infrequently [7] [19]. The attack is conducted on FPGA based deep learning accelerator. Both black-box and gray-box scenarios are investigated. For the black-box attack, no model knowledge is assumed. Surprisingly, by increasing the glitch intensity up to 10% of the total computation clock cycles, 8 out of 9 models can be misclassified with a success rate of 98%. Under the gray box scenario, the number of clock glitches can be minimized by targeting specific convolution layer. Our extensive experiments show that by perturbing at most 1.7% of clock cycles, which is 5.9x less than the black box setting, the outputs of ResNet50 [20] for all the test inputs are misclassified.

The rest of the paper is organized as follows. DNN accelerator, adversarial examples and existing hardware attacks on deep learning are briefly introduced in Section II. The threat model and attack approach are described in Section III. Section IV presents the experiment setup and results. Potential countermeasures are suggested before the paper is concluded in Section V.

Fig. 1. A General Architecture of DNN Accelerator

II. RELATED WORKS

This section reviews the hardware accelerator for DNN, and the adversarial and fault attacks.

A. Deep Learning Accelerator

Due to the high demand on computing power of DNN, efficient hardware accelerators have been developed to enable deep learning on resource and energy-constrained IoT endpoint devices [1]. Data quantization is the primary technique utilized by deep learning accelerator as a coarsely quantized numerical representation reduces memory requirement, and saves hardware and power consumption by reduced precision arithmetic operations [21]. The throughput can also be increased by utilizing massively parallel processing engines (PE) [22] [23].

Fig. 1 illustrates the general architecture of a deep learning accelerator. An ARM/RISC based host processor works in tandem with the highly parallel and energy-efficient PE array to adapt rapidly to fast developing DNN algorithms by offloading the computationally intensive tasks, such as convolutional layer and fully-connected layer, to the PEs. Each PE contains pipelined fixed-point multipliers and accumulators. The data controller and global buffer are optimized to maximize on-chip data reuse [21] [23], which saves considerable power by avoiding off-chip memory access. The PE array is separated from other domains such as clock [24] and supply voltage modules [25] to provide flexibility in voltage-frequency scaling. Throughput-oriented accelerators can use a high-frequency clock for computation array while power-oriented accelerator can easily apply voltage scaling on the PEs [25] [26].

B. Adversarial Attacks

Despite high classification accuracy has been achieved on DNN models, small adversarial perturbation added on input layer can result in wrong prediction. Images with carefully crafted disturbances are called adversarial examples [2]. Goodfellow et al. [3] explain adversarial examples from the linearity of DNN models and propose a fast gradient sign method (FGSM) to generate them. The adversarial example x' is generated by

$$x' = x - \epsilon \cdot \text{sign}(\nabla \text{loss}(x, y)) \quad (1)$$

where x is the clean input image, y is the correct label and ϵ is an adjustable parameter to constrain L_∞ norm (the maximum change of any input elements) on perturbations.

Carlini and Wagner [4] provide a powerful attack algorithm to find a small disturbance δ for a given x by

$$\text{minimize } \|\delta\|_p + c \cdot f(x + \delta), \quad x + \delta \in [0, 1]^n \quad (2)$$

where c is a constant larger than 0 and $f(\cdot)$ is an alternative objective function to that of [2]. The most efficient attack [4] is obtained by solving (2) with the L_2 norm (root-mean-square distance).

Both FGSM [3] and C&W [4] attacks create human-imperceptible adversarial examples. As the magnitude of the perturbations is constrained by either L_∞ or L_2 , these adversarial examples are generated by changing all the pixels from the original image with small values.

C. Fault Injection Attacks

Fault injection can disturb the circuit functionality, thus creating malicious output. Several studies of fault attack on deep learning reveal the vulnerabilities of DNN hardware. Effect of tampering model parameters, such as saturating the last layer's bias, was studied in [9]. Albeit effective, it generates conspicuous behaviour where all the outputs are converged to a single specific class. An improved algorithm [10] has been proposed to minimize the modification of parameter and only selective image will be misclassified. However, the simulations performed in both [9] and [10] assumed that the attacker can modify any parameter to an arbitrary value. In [11], Hong et al. utilize rowhammering to manipulate model weights in high performance computing cluster. It is worth noting that the attack can be mitigated by low-precision numeral representation, as suggested in [11], which happens to be a common practice of existing deep learning accelerator for edge applications.

Attacks on edge deep learning hardware have been attempted recently. RAM-Jam [12] exploits memory collision by writing opposite patterns to the same address of dual-port RAMs. Such malicious attack will create voltage drop and excessive heat, which can flip the bits of model weights stored in the memory. We find two potential issues of RAM-Jam attack. First, model parameters interpolation, as well as perturbation induced by [9] [10] [11], will leave a non-erasable signature in the memory. Second, voltage underfeeding affects the power distribution network. It not only disrupts the DNN processing but can also result in a denial of service. Either the memory trace or a system hangup can be easily detected and blocked by the device owner. A physical fault attack on DNN was demonstrated in [5] to disturb its hardware operation by pointing focused laser beam on silicon die. By skipping operation of activation functions, input images can be misclassified. Instruction skip fault [5] is feasible to general propose processor, such as microcontroller, with 32-bit full precision data but is hard to be induced on dedicated parallel and bitwidth optimized DNN accelerator. The threats of hardware trojan and varying layer activations on DNN have also been studied. The attack mainly restricts to small scale classification problem [13] [14] or is emulated only on software model [6] [15].

III. ATTACK METHODOLOGY

A. Threat Model

We assume that the attacker is aiming at deluding a physically deployed deep learning accelerator into a wrong decision

without leaving any trace or faults in the circuit immediately after the attack. The stealthy requirement has ruled out invasive and tamper-evident probing attacks into the circuits internals. We assume that the attacker knows the general architecture of the hardware but not the low level implementation details, such as the locations of the DNN parameters. Both black-box and gray-box penetrations are considered. For black-box attacks, the adversary has no knowledge of DNN model running on the hardware. For the gray-box attack, the adversary is assumed to have complete knowledge about the model structure, weight parameters and input images. This will allow the fault to be induced on target internal layer of the DNN instead of treating the deep learning model as an end-to-end system for adversarial attacks.

A successful black-box attack will produce misclassified output with high probability regardless of the input images and models. The gray-box scenario provides the adversary with an extra degree of freedom to attack selective layer of the DNN. Our objective is to minimize the number and duration of injected faults to reduce the attack cost and increase its stealth.

B. Imperceptible Misclassification Attack

1) *Attack model*: We leverage on the timing violation in computation to produce imperceptible misclassification of input image on deep learning accelerator. This attack is inconspicuous, as the corrupted intermediate results will be superseded by correct intermediate results once the perturbation subsides after the prediction of the current input. No trace or distortion will be left in the memory or permanently stuck at a computing node to affect the next input. Under normal operating condition, the clock signal of an accelerator should fulfill the following timing requirement:

$$T_{clk} \geq t_{pcq} + t_{pd} + t_{setup} + t_{skew} \quad (3)$$

where t_{pcq} is the worst-case propagation delay from the clock rising edge to the flip-flop's output, t_{pd} is the maximum combinational logic delay between two flip-flops, t_{setup} is the setup time of the flip-flop and t_{skew} is the clock uncertainty. Reducing the clock period T_{clk} below the minimum or increasing the clock skew until there is no setup time margin will lead to timing violation. The fault intensity due to timing violation is not evenly distributed across all bits of an affected result. As higher order bits in most arithmetic operations usually take longer time to settle, they are more likely to be flipped by the setup time violation.

As depicted in Fig. 1, the PE array makes use of separate clock/voltage domains for throughput and power optimization. This can be an exploitable source of vulnerability for imperceptible attack, as disturbing the local clock signal to the PE array will not notably impact the functionality of other circuits. Existing fault injection techniques can induce glitches into hardware by equipment [7] or through malicious attack on host processor [27]. It has been demonstrated that the clock signal and supply voltage of an ARM based SoC can be successfully manipulated [27]. In this work, we show that the DNN accelerator can be fooled by momentary disturbing the clock signal of the PE array.

An overview of the timing diagrams of relevant signals in our attack on deep learning accelerator is provided by Fig. 2. clk_s is a slow operational clock of the peripheral circuits such as data controller and configuration unit. Let clk_f be the original fast reference clock signal derived from clk_s to the PE. clk_f

Fig. 2. Timing Diagrams of Glitch Injection into Deep Learning Accelerator

is shifted 90 degree to generate clk_f_{90} . To inject a glitch, a monostable pulse *trigger* is used to multiplex clk_f_{90} into clk_f at specific cycles of clk_f . The multiplexed output is clk_{mux} . When *trigger* is asserted, clk_f_{90} will be selected to clk_{mux} to produce a clock cycle of period T_{glitch} . Otherwise, the original clock period T_{clk} of the PE will be preserved in clk_{mux} by selecting clk_f . By operating the PE array under clk_{mux} instead of clk_f , erroneous computations will occur on those T_{glitch} cycles as due to violation of (3) with the reduced clock period. The errors (marked by gray data blocks in Fig. 2) will diffuse or accumulate as they propagate to subsequent stages of computation before the final inference stage.

As shown in Fig. 2, the number of glitches in clk_{mux} is determined by the frequency of *trigger*. The number of glitches induced over a complete forward-pass to the inference stage is referred to as the glitch intensity. Glitch intensity can be controlled by the activation frequency of the *trigger* pulse.

2) *Attack procedure*: A black-box attack can be mounted by a straightforward application of clk_{mux} to the PE array with different patterns of *trigger*. This is an image and model independent attack. The success rate of this kind of attack is evaluated in Section IV.

For gray-box scenario, the attacker can predict the timing of the pipelined operations and induce faults on any internal layer of the target model. Besides the model internals and input images, circuit characteristics on the target model is also necessary to increase the prediction accuracy. Computation workloads of different layers can vary depending on the DNN algorithm. The number of clock cycles required for the computation of a specific layer of a hardware accelerator is not solely dictated by the algorithm. Theoretically, the PE array can achieve 100% efficiency with proper data streaming. Due to bandwidth limitation and throughput optimization, the computation latency of all layers may not be equalized. To obtain the number of clock cycles consumed by different layers, it is necessary to profile the execution time of the target deep learning accelerator.

Algorithm 1 describes our gray-box attack. With the given model details (J), a list of model layers (L), a set of glitch intensity (G), the latency of each layer (T) and the input image (I), the stealthiest attack pattern (p_s) of *trigger* can be determined. This pattern can successfully misclassify a specific glitch intensity injected into a single selected layer.

Algorithm 1 Gray-box attack procedure

Require: Model(J), Layer list(L), Glitch Intensity(G), Image(I), Computation clocks(T)

Ensure: Stealthiest attack pattern (p_s), Efficiency (e)

// initialize $P \leftarrow \{\}$, $E \leftarrow \{\}$

// correct label of the input image

1: $c \leftarrow J(I, 0, 0)$

// layer-wise glitch injection with different intensity

2: **for all** l such that $1 \leq l \leq L$ **do**

3: **for all** g such that $1 \leq g \leq G$ **do**

4: $c_p \leftarrow J(I, l, g)$

5: **if** $c_p \neq c$ **then**

6: $P \leftarrow P \cup (l, g)$

7: **end if**

8: **end for**

9: **end for**

// find the least glitch pattern

10: **for all** p such that $p \in P$ **do**

11: $e_p \leftarrow t_l \times g$

12: $E \leftarrow E \cup e_p$

13: **end for**

14: $e \leftarrow \min(E)$

15: $p_s \leftarrow P(\text{index}(\min(E)))$

16: **return** p_s, e

IV. EXPERIMENTAL RESULTS

A. Experimental Setup

We mounted the glitch injection attack on DNN implemented on Xilinx ZCU102 evaluation board which has a ZYNQ UltraScale+ (XCZU9EG-2FFVB1156) MPSoC device. This SoC is manufactured in 16nm FinFET+ process equipped with a quad-core ARM Cortex-A53 host processor. In addition, the programmable fabric consists of 600k logic cells and 2520 dedicated DSP slices. We configure our experimental setup as shown in Fig. 3.

Fig. 3. Block Diagram of Experimental Setup

This setup consists of a deep learning processor unit (DPU) [24], a host PC, an ARM processor and some peripheral circuits. The DPU is a Xilinx IP which features configurable computing parallelism and DSP usage. We follow the reference design in [24] by setting DPU core to 2 and using B4096 architecture with high DSP usage. In this configuration, each DPU core has the highest parallelism, and both multiplication and accumulation are calculated by the DSP units that take full advantage of the fine-grained building blocks in Xilinx devices for efficient computation of convolutions and deconvolutions. Similar to

TABLE I
SUMMARY OF DNN MODELS RUNNING ON DPU UNDER NORMAL OPERATION

Model	Size (MB)	MACs ($\times 10^9$)	Throughput (FPS)	Accuracy (%)
Inceptionv1	6.9	1.58	26.02	65.7
Inceptionv2	11	2	24.94	68.2
Inceptionv3	23	5.7	24.79	74.8
Inceptionv4	42	12.3	8.53	78.5
MobileNetv1	4.3	0.15	35.56	61.7
MobileNetv2	3.7	0.3	33.13	60.5
DenseNet121	8.1	2.85	17.54	68.8
ResNet50	25	3.9	17.5	68.7
VGG16	132	15.5	6.7	65.3

general deep learning accelerators, the DPU has separate clock domains for data controller and computation unit (i.e., DSP slices). In our setup, we inject glitches to the DSP clock signal and control the glitch intensity by integrating four extra blocks. The 90 degree phase shift clock is generated from the DSP clock by the clock wizard. A clock multiplexer is instantiated to switch the *dsp_clk_mux* between the normal clock and the shifted clock. Glitch intensity is adjusted by setting the parameter of the DSP clock counter. The fault trigger signal is controlled by the counter's output and an enable signal from the ARM processor. Both the intensity setting and enable signals are attached to the EMIO pins of the ARM processor. Those EMIOs can be accessed from the OS kernel. UART and Ethernet connections are established between the host PC and the board for the control and data transfer. The DPU is operating at 100MHz while the DSP at 200MHz.

Besides the hardware configuration, the embedded OS and DNN application are built by the Xilinx's Petalinux and DNNDK [28] tool chain, respectively. Full precision DNN model will be converted to the 8-bit fixed point version after DNNDK compilation. The quantized model will be packaged with C++ application code to a single hybrid executable file. This file can then be loaded to configure the FPGA SoC with DPU.

B. Results of Black Box Attack

In this black-box experiment, we evaluated our proposed attack on nine cutting-edge ImageNet models, including Inception v1-v4 [29] [30] [31], MobileNet v1-v2 [32] [33], DenseNet121 [34], ResNet50 [20] and VGG16 [35], with the experimental setup of Fig. 3. A set of glitch intensities (0.01%, 0.02%, 0.033%, 0.1%, 0.2%, 0.33%, 1%, 2%, 3.3% and 10%) is used to evaluate the effectiveness of our attack. The whole ImageNet validation set of 50000 images is evaluated in this experiment.

Table I summarizes all the models running on our DPU testing platform. The reported model size is the actual weights loaded into the accelerator, which are 8-bit quantized data. The number of multiply-and-accumulate operations (MACs) and throughput in frames per second (fps) and the original top-1 accuracy of each model are also provided. Our target is to mislead the inferences of input images that can be correctly classified originally on DPU under normal operation.

Fig. 4 shows the misclassification rate of the nine models under the attacks of various glitch intensities. Different versions of Inception exhibit similar trend of vulnerability. By injecting 2% glitches in the computation clocks, more than 60% of the correctly classified images of these four models will produce wrong predictions. The misclassification rate will converge to more than 99% with 10% glitch injection. Significant successes

Fig. 4. Black-box Attack on Nine ImageNet Models

Fig. 5. Vulnerability among Model Architectures under 0.1% Glitch Intensity

on DenseNet121 and MobileNet (v1&v2) were observed. With as little as 0.01% of injected glitches, it can successfully subvert 99% of the image classification outputs. The misclassification rate of ResNet50 stays relatively constant at slightly above 50 % for glitch intensity from 0.01% to 0.2%, and increases steadily to more than 98% at 10% glitch intensity. VGG16 is most robust under our attack. Nevertheless, a misclassification rate of slightly more than 40% can still be achieved with 10% glitch intensity.

Fig. 5 plots the misclassification rates of the attack with 0.1% glitch intensity against model size, number of MACs and original accuracy for different model architectures. Besides MobileNet family, which have the smallest size with the least number of MACs, there is no clear evidence of any distinctive relationship between the vulnerability and model properties. For instance, ResNet50 has larger model size but is less robust than Inception v1-v3. DenseNet121 has higher classification accuracy than the other four models but is also more vulnerable to misclassification under our attack. Based on Fig. 4 and Fig. 5, it can be concluded that our attack is effective on ImageNet models regardless of the model size, computation requirement and original classification accuracy.

C. Results of Gray Box Attack

For the gray-box experiment, we apply Algorithm 1 on ResNet50, which is a common benchmark network. We first profile the ResNet50 network on our DPU platform. The workload and runtime for each layer are shown in Fig. 6. As discussed in Section III, the same workload can result in different runtime on the hardware accelerator. The layer-wise computation latency (T) has been extracted from the hardware profiler. We tested the first 256 images from the ImageNet validation set in this experiment.

Since only one layer is targeted for each run in this attack scenario, glitches will be injected in only those cycles when the image data is executed in this layer. This gives a localized

Fig. 6. Hardware Profiling on ResNet50

Fig. 7. Effective Attack Pattern on IMG44 and IMG154

glitch intensity of 50% if one glitch (T_{glitch}) is added in every alternate clock cycle (T_{clk}) during the computation of the target layer. The actual glitch intensity should be multiplied by $\frac{t_{layer}}{t_{total}}$, where t_{layer} and t_{total} is the runtime of the selected layer and the total runtime of all layers, respectively. To precisely inject the glitches on the target layer, the victim layer is treated as an individual task in DNNDK compilation so that *trigger* will only be toggled when the victim task is running. The output from the victim layer will still propagate to the following layers.

Two images (IMG44 and IMG154) are selected from the set of 256 images to show the localized glitch intensities at those layers that their applications can lead to misclassification in Fig. 7. Only the least localized glitch intensity on each target layer that has succeeded in misclassifying these images is presented. Layers that have no data point means that glitches injected at the execution cycles of these layers cannot change the image classification. IMG44 is highly vulnerable as clock perturbation during the DSP operations in any single layer of ResNet50 can cause this image to be misclassified. It is found that IMG44 can be easily misclassified by applying merely 0.01% of localized glitch intensity on res2a_branch2a layer of ResNet50. Based on t_{layer} from the hardware profiler, only 4 out of 7318000 clock cycles need to be perturbed. In contrast, IMG154 can only be misclassified by glitch injection onto 5 specific layers of ResNet50, as shown in Fig. 7. The minimum number of glitches required to mislead the DNN accelerator for IMG154 is 16000, which constitute 0.2% of the clock cycles in the complete forward-pass. All the 256 images evaluated in this experiment can be misclassified by glitch injection to one specific layer, which requires no more than 1.7% actual glitch intensity.

This imperceptible misclassification attack exposes the vulnerability of the voltage-frequency scaling of DNN accelerator by separating the clock/voltage domains for the PE array and

peripherals [25]. To increase the resilience of DNN hardware against such fault induced attacks, one potential method is to improve the robustness of DNN models. By constraining the Lipschitz constant, defensive quantization [36] can reduce the error amplification effect during the forward propagation. Another straightforward countermeasure is to employ one single domain for the entire accelerator. The energy efficiency gained from voltage-frequency scaling will be reduced. To maintain the same throughput as separate domain's accelerator, double DSP usage and 33.7% overhead of LUTs are required [37]. The security enhancement will inevitably incur some trade-off in energy efficiency and hardware resource requirements without compromising the throughput.

V. CONCLUSION

In this paper, we present an imperceptible misclassification attack to delude deep learning accelerator by perturbing the clock signal to the PE array infrequently. Our attack is stealthy as it leaves no sign after the attack in comparison with other parameter interpolation attacks on DNN hardware. We demonstrated our attack on FPGA based DNN accelerator for both black-box and gray-box attack scenarios. For the black-box attack, above 98% misclassification on 8 out of 9 ImageNet models can be achieved with 10% glitch intensity. For the gray-box attack, 5.9x less (1.7%) glitch intensity needs to be injected to successfully subvert the correct classification results of all the images input to the hardware accelerator of ResNet50.

ACKNOWLEDGMENT

This research is supported by Singapore Ministry of Education AcRF Tier 1 Grant No. 2018-T1-001-131.

REFERENCES

- [1] V. Sze, Y. H. Chen, T. J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.
- [2] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," presented at Int. Conf. Learning Representations (ICLR), Banff, Canada, Apr. 2014.
- [3] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," presented at Int. Conf. Learning Representations (ICLR), San Diego, CA, USA, May 2015.
- [4] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Priv. (SP)*, San Jose, CA, USA, May 2017, pp. 39–57.
- [5] J. Breier, X. Hou, D. Jap, L. Ma, S. Bhasin, and Y. Liu, "Practical fault attack on deep neural networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, ser. CCS '18, New York, NY, USA, Oct. 2018, pp. 2204–2206.
- [6] Y. Zhao, X. Hu, S. Li, J. Ye, L. Deng, Y. Ji, J. Xu, D. Wu, and Y. Xie, "Memory trojan attack on neural network accelerators," in *Proc. Des. Automat. Test Euro. Conf. Exhibi. (DATE)*, Grenoble, France, March 2019, pp. 1415–1420.
- [7] M. Agoyan, J.-M. Dutertre, D. Naccache, B. Robisson, and A. Tria, "When clocks fail: On critical paths and clock faults," in *Proc. Smart Card Res. Adv. App.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 182–193.
- [8] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee *et al.*, "Flipping bits in memory without accessing them: An experimental study of dram disturbance errors," in *Proc. 41st Int. Symp. Comput. Archit.*, ser. ISCA '14, Piscataway, NJ, USA, Jun. 2014, pp. 361–372.
- [9] Y. Liu, L. Wei, B. Luo, and Q. Xu, "Fault injection attack on deep neural network," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Des. (ICCAD)*, Irvine, CA, USA, Nov 2017, pp. 131–138.
- [10] P. Zhao, S. Wang, C. Gongye, Y. Wang, Y. Fei, and X. Lin, "Fault sneaking attack: A stealthy framework for misleading deep neural networks," in *Proc. 56th Annual Des. Automat. Conf. 2019*, ser. DAC '19. New York, NY, USA: ACM, Jun. 2019, pp. 165:1–165:6.
- [11] S. Hong, P. Frigo, Y. Kaya, C. Giuffrida, and T. Dumitras, "Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks," in *Proc. 28th USENIX Securi. Symp.*, Santa Clara, CA, USA, Aug. 2019, pp. 497–514.
- [12] M. M. Alam, S. Tajik, F. Ganji, M. Tehranipoor, and D. Forte, "Ram-jam: Remote temperature and voltage fault attack on fpgas using memory collisions," in *Proc. Work. Fault Diag. Toler. Crypt. (FDTCTC)*, Atlanta, GA, USA, Aug 2019, pp. 48–55.
- [13] J. Clements and Y. Lao, "Hardware trojan design on neural networks," in *Proc. IEEE Int. Symp. Circ. Syst. (ISCAS)*, Sapporo, Hokkaido, Japan, May 2019, pp. 1–5.
- [14] T. A. Odetola, H. R. Mohammed, and S. R. Hasan, "A stealthy hardware trojan exploiting the architectural vulnerability of deep learning architectures: Input interception attack (iia)," 2019.
- [15] X. Jiao, M. Luo, J.-H. Lin, and R. K. Gupta, "An assessment of vulnerability of hardware neural networks to dynamic voltage and temperature variations," in *Proc. 36th Int. Conf. Computer-Aided Design*, ser. ICCAD '17, Nov. 2017, pp. 945–950.
- [16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. Comput. Vision Pattern Recogn. (CVPR)*, Miami, FL, USA, Jun. 2009, pp. 248–255.
- [17] R. Girshick, "Fast r-cnn," in *Proc. IEEE Int. Conf. Comput. Vision (ICCV)*, Washington, DC, USA, Dec 2015, pp. 1440–1448.
- [18] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Patt. Recogn. (CVPR)*, Boston, Massachusetts, USA, June 2015, pp. 3431–3440.
- [19] N. F. Ghalaty, B. Yuce, M. Taha, and P. Schaumont, "Differential fault intensity analysis," in *Proc. Work. Fault Diag. Toler. Crypt.*, ser. FDTCTC '14. Washington, DC, USA: IEEE Computer Society, Sep. 2014, pp. 49–58.
- [20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Patt. Recogn. (CVPR)*, Las Vegas, Nevada, USA, Jun. 2016, pp. 770–778.
- [21] Y. H. Chen, T. Krishna, J. Emer, and V. Sze, "14.5 eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, San Francisco, CA, USA, Jan. 2016, pp. 262–263.
- [22] B. Moons and M. Verhelst, "An energy-efficient precision-scalable convnet processor in 40-nm cmos," *IEEE J. Solid-State Circuits*, vol. 52, no. 4, pp. 903–914, Apr. 2017.
- [23] Y. Ma, Y. Cao, S. Vrudhula, and J. Seo, "Optimizing the convolution operation to accelerate deep neural networks on FPGA," *IEEE Trans. VLSI Syst.*, vol. 26, no. 7, pp. 1354–1367, Jul. 2018.
- [24] Xilinx, *DPU for Convolutional Neural Network v1.2*.
- [25] B. Moons and M. Verhelst, "A 0.3–2.6 TOPS/W precision-scalable processor for real-time large-scale convnets," in *Proc. IEEE Symp. VLSI Circuits (VLSI-Circuits)*, Honolulu, Hawaii, USA, Jun. 2016, pp. 1–2.
- [26] P. N. Whatmough, S. K. Lee, H. Lee, S. Rama, D. Brooks, and G. Wei, "14.3 a 28nm SoC with a 1.2GHz 568nJ/prediction sparse deep-neural-network engine with >0.1 timing error rate tolerance for IoT applications," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, San Francisco, CA, USA, Feb. 2017, pp. 242–243.
- [27] A. Tang, S. Sethumadhavan, and S. Stolfo, "CLKSCREW: Exposing the perils of security-oblivious energy management," in *Proc. 26th USENIX Security Symposium*, Vancouver, BC, Aug 2017, pp. 1057–1074.
- [28] Xilinx, *Deep Neural Network Development Kit V3.0*.
- [29] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Patt. Recogn. (CVPR)*, Boston, Massachusetts, USA, Jun. 2015, pp. 1–9.
- [30] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, June 2016, pp. 2818–2826.
- [31] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proc. 31st AAAI Conf. Artif. Intell.*, ser. AAAI'17, Feb. 2017, pp. 4278–4284.
- [32] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.
- [33] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Patt. Recogn.*, Salt Lake City, USA, June 2018, pp. 4510–4520.
- [34] G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Patt. Recogn. (CVPR)*, Honolulu, Hawaii, July 2017, pp. 2261–2269.
- [35] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," presented at Int. Conf. Learning Representations (ICLR), San Diego, CA, USA, May 2014.
- [36] J. Lin, C. Gan, and S. Han, "Defensive quantization: When efficiency meets robustness," presented at Int. Conf. Learning Representations (ICLR), New Orleans, LA, USA, May 2019.
- [37] https://www.xilinx.com/xilinxtraining/assessments/portal/ml-edge/dpu-overview/story_html5.html.