Work-in-Progress: Towards Evaluating CNNs Against Integrity Attacks on Multi-tenant Computation

XIANGRU CHEN, DIPAL HALDER, KAZI MEJBAUL ISLAM, and SANDIP RAY, University of Florida, USA

We present an infrastructure for evaluating CNN models for vulnerability against a variety of integrity attacks. Our focus is on attacks that corrupt CNN computations with an impact on prediction/classification accuracy. The attack model encompasses a variety of mechanisms including injection of faults and glitches, integrity attacks on compute resources, etc. Our tool enables users to explore a variety of attack configurations, targets, and accuracy drops tolerated by the model. Experiments with our tool on publicly available CNN models show the vulnerability between layers is different, which can be exploited to protect important parts of the computation even when deployed on untrusted accelerators.

Additional Key Words and Phrases: Integrity Attacks, Multi-tenant, Neural Networks, Evaluation Tool, CNN

ACM Reference Format:
Xiangru Chen, Dipal Halder, Kazi Mejbaul Islam, and Sandip Ray. 2023. Work-in-Progress: Towards Evaluating CNNs Against Integrity Attacks on Multi-tenant Computation. In *International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES '23 Companion), September 17–22, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3607889.3609091

1 INTRODUCTION

Convolutional Neural Networks (CNNs) have received pervasive applications in recent years, with critical applications including medical, cyber security, film pre-production, and gaming, performing tasks like object locating, face recognition, target partition, and action tracking. The effectiveness of CNNs in these computations depends critically on the improved compute performance afforded by cloud-based accelerators. Consequently, along with large-scale CNNs, the past few years also witnessed a burst of efforts in the design and deployment of accelerators for CNN computation.

A key aspect of modern accelerators is *multi-tenancy*, driven by a computation-as-a-service paradigm. Multi-tenant service models are available to a variety of hardware accelerators, including FPGAs, GPUs, and even custom ASICs (Tensors), in addition to conventional CPU-based compute resources. Unfortunately, multi-tenant accelerations have one critical drawback: spatial and temporal sharing of accelerator resources makes the applications vulnerable to a variety of security exploitation by a spatially or temporally adjacent rogue tenant that can subvert the integrity of the application [4, 7]

In this paper, we study the vulnerability of CNN computations to integrity attacks in multi-tenant computations. Our threat model entails a possibly rogue spatially adjacent tenant $\mathcal{A}$ corrupting the results of intermediate computations in the CNN. The corruption is modeled through bit flips. A key threat of interest is fault attacks, where the adversary can instigate bit flip in adjacent computation through an over-driving attack on the accelerator targeting the dynamic voltage-frequency systems [5]. Our key contribution is a tool that enables comprehensive evaluation of a CNN model $\mathcal{D}$ *before* deployment. Our tool can perform fine-grained analysis of intermediate computations of $\mathcal{D}$ to identify vulnerable computation components. The inference can be used to determine which computation components can be safely off-loaded to accelerators and which components require additional protection under multi-tenancy.

CASES '23 Companion, September 17–22, 2023, Hamburg, Germany

ACM ISBN 979-8-4007-0290-7/23/09.
https://doi.org/10.1145/3607889.3609091

2 TOOL ARCHITECTURE

2.1 High-level Overview

The key idea of the tool is to orchestrate systematic insertion of bit flips in various computation layers of a CNN corresponding to the attack model, and determine the effect of the corruption on the actual inference. It takes a CNN computation model (with its training dataset) and a number of configuration parameters to perform its analysis (see Table 1). The key contribution is to enable systematic navigation of large parameter space. The flow includes the following three major steps:

Step 1. The model is trained with its traditional training dataset to estimate the original accuracy of inference (under no adversarial scenarios). In this step, several training options (*e.g.*, learning rate, epoch number, etc.) are included to get the highest accuracy.

Step 2. Bit corruption is simulated by injecting errors in the intermediate data generated by the targeted computation layer. For exploring attacks on the computation of Layer 1, errors should be reflected in the result from Layer 1. By systematically flipping part of the bits in Layer 1's result, our tool simulates the error injection and propagation.

Step 3. Multiple combinations of attack level are utilized to give a clear sense of vulnerability. The detailed combination will be discussed as different modes below.

Table 1. Different Parameters for the Tool

Parameter Name	Significance
Error Number	Induced error in the neural network model
Bit Number	Total number of bits flipped as a result of fault attack
Attack Layer	Targeted layer for the attack
Distance Level	Bit type which can be "mantissa" or "exponent"

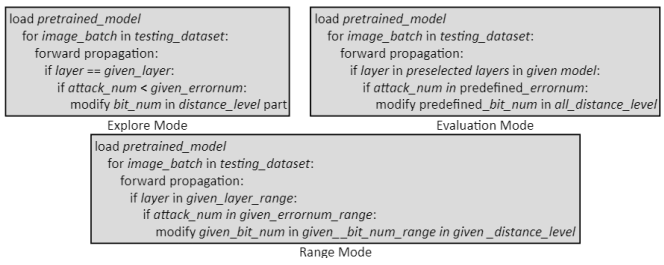

Fig. 1. Three Modes of Our Tool. Only the inference phase is included.

2.2 Tool Modes

Our tool supports three modes to enable the analysis of CNN vulnerability at different levels of granularity. Note that CNN computation generally uses floating point data with separate mantissa and exponent components. The impact of corruption on these two components on the final inference of the CNN can be significantly different. In general, increasing the mantissa's bit allocation can improve computation precision, whereas increasing the exponent's bit allocation can improve computation range [2, 6]. Table 1 shows all the available parameters that can be exercised with this tool.

Explore Mode: This mode is helpful in providing an intuitive and fine-grained grasp of the parameters critical for accuracy. It also supports user-guided exploration, providing high user control for switching on and off different parameters.

Range Mode: This mode is of intermediate granularity between explore mode and evaluation mode (see below) and enables succinct and automated exercising of various parameter ranges.

Evaluation Mode: This mode enables automated exploration of the parameter space. This mode is for "black box evaluation", *i.e.*, for a completely automated sweep across the parameter space without requiring any user control. Since it entails a complete exploration of the parameter space, the mode can incur high computational overhead.

Table 2. CNN Accuracy on Different Layers in Explore Mode

Layer Name	Accuracy(%) Distance Level "mantissa"	Accuracy(%) Distance Level "exponent"	Layer Number
Conv 1	69.67	69.99	I
ReLu 1	80.95	69.30	II
Pool 1	71.75	64.81	III
Conv 2	73.02	72.81	IV
ReLu 2	86.11	69.23	V
Pool 2	78.32	66.03	VI
Conv 3	72.98	72.55	VII
ReLu 3	84.18	67.20	VIII
Conv 4	71.30	71.86	IX
ReLu 4	82.49	66.97	X
Conv 5	71.36	70.48	XI
ReLu 5	86.92	68.02	XII
Pool 3	79.45	65.77	XIII
Error Number = 60 | Bit Number = 5			

3 EXPERIMENTAL RESULTS

Our tool is implemented in the PyTorch environment. For the experiments, we selected the AlexNet model [3] trained on the CIFAR10 dataset [1]. There are 5 convolutional layers, 5 ReLu layers, and 3 Pooling layers in AlexNet. Our experiments exercise various parameters of the model including error number, distance level, bit number, and different layers for introducing the attack scenario in all three modes. Table 2 shows the accuracy for all layers of this model in different distance levels under a specific attack condition in *Explore Mode*. Note that accuracy for Conv 1 layer decreases to 69.67% for the distance level "mantissa". On the other hand, for distance level "exponent" Pool 3 layer has the lowest accuracy of 65.77%. From this observation, we can conclude that among all thirteen layers, for distance level "mantissa", the first convolution layer is more vulnerable to attack whereas the first pooling layer is more susceptible with bit flip attack for distance level "exponent".

4 CONCLUSION

To our knowledge, our work represents the first approach for the systematic evaluation of CNN models against threats inducing bit corruptions in different computation layers. We discussed the kinds of design choices, parameter configurations, and evaluation frameworks involved in such designs.

Although the approach is promising, we have only scratched the surface. In future work, we will perform more extensive fault attack experiments. Also, we will work on detecting fault attacks in real time and find out a robust technique to mitigate this attack for a given CNN model.

Acknowledgements. This research has been partially supported by the Semiconductor Research Corporation under Contract No. 2021-HWS-3060.

REFERENCES

[1] 2009. CIFAR10. https://www.cs.toronto.edu/~kriz/cifar.html. [Accessed on March 21, 2023].
[2] David Goldberg. 1991. What Every Computer Scientist Should Know about Floating-Point Arithmetic. *ACM Comput. Surv.* 23, 1 (mar 1991), 5–48. https://doi.org/10.1145/103162.103163
[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* 60, 6 (may 2017), 84–90. https://doi.org/10.1145/3065386
[4] Tuan La, Khoa Pham, Joseph Powell, and Dirk Koch. 2021. Denial-of-Service on FPGA-based Cloud Infrastructure - Attack and Defense. *IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES)* 2021, 3 (9 July 2021), 441–464. https://doi.org/10.46586/tches.v2021.i3.441-464
[5] Majid Sabbagh, Yunsi Fei, and David Kaeli. 2020. A Novel GPU Overdrive Fault Attack. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*. 1–6. https://doi.org/10.1109/DAC18072.2020.9218690
[6] Guido D. Salvucci. 1985. Ieee standard for binary floating-point arithmetic.
[7] Shanquan Tian, Ilias Giechaskiel, Wenjie Xiong, and Jakub Szefer. 2021. Cloud FPGA Cartography using PCIe Contention. In *2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. 224–232. https://doi.org/10.1109/FCCM51124.2021.00035