Vulnerability analysis on noise-injection based hardware attack on deep neural networks

Liu, Wenye; Wang, Si; Chang, Chip-Hong

2019

Liu, W., Wang, S. & Chang, C.-H. (2019). Vulnerability analysis on noise-injection based hardware attack on deep neural networks. 2019 Asian Hardware Oriented Security and Trust Symposium (AsianHOST).

<https://hdl.handle.net/10356/136863>

© 2019 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Downloaded on 26 Apr 2023 22:35:00 SGT

Vulnerability Analysis on Noise-Injection Based Hardware Attack on Deep Neural Networks

Wenye Liu, Si Wang and Chip-Hong Chang

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

Email: wliu015@e.ntu.edu.sg, si003@e.ntu.edu.sg, echchang@ntu.edu.sg

Abstract—Despite superior accuracy on most vision recognition tasks, deep neural networks are susceptible to adversarial examples. Recent studies show that adding carefully crafted small perturbations on input layer can mislead a classifier into arbitrary categories. However, most adversarial attack algorithms only concentrate on the inputs of the model, effect of tampering internal nodes is seldom studied. Adversarial attack, if extends to deployed hardware system, can perturb or alter intermediate data during real time processing. To investigate the vulnerability implication of deep neural network hardware under potential adversarial attacks, we comprehensively evaluate 10 popular DNN models by injecting noise into each layer of these models. Our experimental results indicate that more accurate networks are more prone to disturbance of selective internal layers. For traditional convolutional network structures (AlexNet and VGG family), the last convolution layer is most assailable. For state-of-the-art architectures (Inception, ResNet and DenseNet families), as little as 0.1% or one element per channel of perturbations can subvert the original predictions, and over 65% of computational layers suffer from this vulnerability. Our findings reveal that optimization of accuracy, model size and computational efficiency can unconsciously sacrifice the robustness of deep learning system.

I. INTRODUCTION

Deep neural network (DNN) is a data driven learning algorithm that has achieved phenomenal successes in computer vision. Convolutional architecture [1] with aggressively increasing depth of the network [2] can make state-of-the-art DNN models surpass human capabilities in image classification tasks. With the help of hardware acceleration, DNN has also been deployed for many security critical applications such as self-driving car and face authentication. Unfortunately, even the most impressive DNN model can make very trivial misclassification errors due to adversarial examples which mix natural images with carefully crafted small perturbations [3] [4].

Various adversarial algorithms have been proposed for non-targeted, class-specific and image-agnostic attacks [3]. Most adversarial attacks aim to create small additive distortions on the inputs of the model by treating the DNN system as an end-to-end module. Nevertheless, layer activations are also vulnerable to malicious tampering especially when the deep learning model is executed on dedicated hardware. Several hardware based fault-injection attacks such as clock glitch [5] [6], optical laser [7] and hardware trojan [8] have raised the threat of deployed hardware. Globalization of hardware design flow and the increasing exploration and launch of AI products and services by non-technical companies have also provided more space and practicality to extend or modify such attacks

to stealthily inject disturbance into internal layers of DNN during inference.

So far the vulnerability of tampered layer outputs within different network architectures has not been fully studied. Liu et. al. [9] illustrates possible hardware attacks to modify model parameters, particularly the last layer's bias, which can alter the prediction output. However, targeting last layer is not stealthy and the modified parameters can be easily detected by model integrity checking. Injecting hardware Trojans in neural network accelerator can also lead to misclassification [10] [11]. While most existing studies of hardware threat on DNN [9] [10] [12] only evaluate their implementations on simple 10-class classifications. Zhao. et. al. [11] emulate the trojan attack on three large scale models but no result on individual layers is reported. This paper investigates the layer-wise vulnerability of the most lucrative ImageNet [13] models. We revisit 10 popular DNN architectures, including AlexNet [1], VGG [14], Inception (GoogleNet) [15], ResNet [2], DenseNet [16] and MobileNet [17], to inject small portions of random perturbation to evaluate the noise immunity of each layer. Based on our experimental results, we conclude that tampering the intermediate layer is more effective to subvert original prediction rather than the input of the model. With 0.3% noise injected in the last convolution layer, 43.3%, 36.3% and 29.7% misclassification rates are observed on AlexNet, VGG16 and VGG19, respectively. As for state-of-the-art models, over 65% of computational layers of Inception, ResNet and DenseNet architectures experience larger than 80% misclassification rate with an injection of 0.1% noise or single noise element per channel. The lightweight MobileNet-v1 has similar vulnerability as AlexNet, while its version 2 counterpart suffers even higher vulnerability as advanced model architectures. Based on our analysis, we find that extensive use of 1×1 convolution layer can unwittingly sacrifice robustness.

The rest of the paper is organized as follows. DNN architectures, adversarial example and potential hardware attacks are briefly introduced in Section II. The threat model and experiment setup are described in Section III. Section IV discusses and analyzes the results. Finally, the paper is concluded in Section V.

II. PRELIMINARIES

A. Deep Neural Network Architectures

Since the record breaking architecture introduced in 2012 [1], deep convolutional structure has become the mainstream for image classification and other visual tasks. Although different variants are proposed to increase prediction accuracy

Fig. 1. Standard convolution network architecture with different variants of convolution block design

and reduce computation complexity, most of them follow the basic construction of five convolution (CONV) blocks, where each block can contain multiple convolution layers, followed by one to several fully-connected (FC) layers.

Fig. 1 shows the standard network architecture for image classification. Feature extraction is accomplished by the CONV block. It shrinks the height and width of input feature maps but expands the depth (channel), as shown in the blue block in Fig. 1. Classification is accomplished by the FC layers, which unroll the 3D features to vector and perform matrix-vector multiplications. Each computation layer (except the last FC layer) is followed by a non-linear layer which is omitted in Fig. 1 for simplicity. AlexNet uses a plain structure where each CONV block contains one convolution layer [1]. VGG Nets follow this trend by directly stacking multiple convolution layers in the same block (e.g., VGG16 contains 2 layers in the second block and 3 layers in the third block) [14]. Inception [15] provides an alternative block architecture, which utilizes different sizes of filter (1×1 , 3×3 and 5×5) in parallel. To reduce the computation complexity, it also includes additional 1×1 kernel before 3×3 and 5×5 convolutions for dimensions compression (e.g., $H \times W \times C > H \times W \times (C - N)$). The outputs from different sizes of filter are concatenated and forwarded to the next inception module. Residual mapping [2] introduces a shortcut connection between every few layers to avoid the vanishing gradient problem at training phase. Each residual block adapts to the bottleneck design by having two 1×1 convolution layers responding to squeezing and expanding dimensions, respectively. Dense connection [16] expands the shortcut paths to all subsequent layers. Similar bottleneck layers are included in DenseNet to improve computational efficiency. These variants of CONV block are shown in the lower half of Fig. 1.

As shown in [14], adding more layers can improve classification accuracy while the hardware cost (memory and computation power) also increases dramatically. For deployment of DNN on portable and endpoint devices, model design focuses on efficiency [15]. To improve classification accuracy while keeping smaller model size and lower computation complexity, 1×1 convolution kernel has been extensively utilized.

Convolution layer with 1×1 filters can reduce the channel dimension of input features. The demand on computation power and parameters for each feature extraction layer depends on the input feature dimension. Hence, inserting a 1×1 kernel before the feature extraction layer (3×3 or 5×5 kernel) can effectively reduce the model size and computation complexity. Consequently, state-of-the-art models [2] [15] [16] all utilize 1×1 kernel for feature squeezing before feature extraction.

Recently, another type of lightweight network structure called MobileNets [17] [18] has been proposed for mobile and embedded applications. Its main structure is slightly different from others as it utilizes 6 CONV blocks. In the first version (v1) of MobileNet, normal convolution is converted into one depthwise convolution and one pointwise convolution. With small degradation, this transform reduces the computation and model size dramatically. Version two of the MobileNet emulates residual mapping but uses the shortcut connection directly between the bottlenecks.

B. Adversarial Example

While highly accurate DNN model can exceed human-level accuracy, it is possible to mislead it with high confidence by small carefully crafted perturbations on the input layer. Such threat is known as adversarial example [3] [4]. Many attack algorithms have been proposed for image-specific, universal and 3D printed model perturbations [3]. Fig. 2 shows an example based on C&W adversarial attacks [19]. With imperceptible perturbations added to a legitimate image, a well-trained DNN model can be fooled to make a wrong prediction.

Fig. 2. Adversarial example generated based on C&W attack [19] [20].

Adversarial examples can be generated by solving the following constrained optimization problem [4],

$$\begin{aligned} & \text{Minimize } D(x, x+r) \quad \text{subject to:} \\ & 1. \quad f(x+r) = l \\ & 2. \quad x+r \in [0, 1]^n \end{aligned}$$

where D is the distance metric (L_0 , L_2 or L_∞) between original image and adversarial example, l is the adversarial label. The aim of this problem is to find some small r , when added onto an original image, will change its classification to l , while $x+r$ is still a valid image. To solve this problem, some approximation is required [4]. C&W attack [19] introduces some objective functions for this optimization, which successfully changes any legitimate image to an arbitrary class.

C. Hardware Attacks

It is worthy to note that most effective adversarial attack algorithms work in a white-box scenario by assuming the adversary has complete knowledge about the architecture, weight parameters and even the loss (gradient) of the model. This is not the case for practical non-invasive attacks on DNN hardware. Once a DNN has been deployed and integrated on dedicated hardware, its weights will be frozen and only the inference will be executed [21]. The loss or gradient cannot be back propagated through the hardware accelerator. Therefore, we consider the hardware threat of DNN from more practical scenarios such as clock glitch attack [6] and hardware trojans [8] [10]. Both attack methods are possible to inject disturbance or alter intermediate data when the DNN hardware is performing real-time inference.

Clock glitch attack introduces unscheduled spikes or glitches to the device, which may cause timing violations in digital circuit [6]. Fig. 3 shows a generation of clock glitch signal. Two additional delayed clock signals and a specific trigger signal are required for the generation of faulty clock. The interval between the glitch edge and the next rising edge (T_{glitch}) is much smaller than the normal clock period (T_{CLK}), thus it can cause setup time violations or the output state to be nondeterministic.

Fig. 3. Glitch clock signal generation

The untrusted global supply chain flow can also inflict hardware trojan into the DNN circuit. These small implanted Trojan circuits may remain dormant to evade detection but they can be activated in the field or remotely upon rare conditions to cause malicious modification of circuit behaviors [8]. Fig. 4 shows an example of a hardware trojan, whose payload (c') remains identical to the original signal (c) until triggered by specific input patterns. This attack can alter layer activations of DNN under the constraint of false trigger rate [10].

III. THREAT MODEL AND EXPERIMENTAL SETUP

A. Threat Model

We assume that the attacker's aim is to subvert the original classification output of the DNN hardware, and the training information is not available to the attacker. Access to the circuit internals to directly modify weight parameters is also prohibited, as invasive physical attacks cannot be performed

Fig. 4. An example of Hardware Trojan design [10]

stealthily without being detected upon re-deployment. Moreover, alteration of these parameters can be easily detected by checking the model integrity since this will affect all the input data. We assume that the attacker knows the network structure, which is not private. For real world accelerator, layer activations are quantized to specific data range (e.g., 8-bit representation) for efficient processing [21]. Therefore, the perturbation that can be introduced by the attacker is also limited to the same range of each layer's output. One way to confuse the DNN classifier is to inject noise into the intermediate results (layer activations). Such attack can potentially be launched by clock glitch [5] or hardware trojan [8].

Based on this threat model, we comprehensively evaluate every layer of the target models in order to locate the most assailable layer/layers. Our layer-wise vulnerability analysis identifies the weak links and provides an early warning to potential low-cost and precise attacks. Other than evaluating the relationship between accuracy and robustness of DNN model in an end-to-end manner [20], our analysis also sheds light on quality scaling of custom chip implementation of DNN for accuracy, cost and robustness.

B. Experimental Setup

In this study, we evaluate 10 DNN models among six network families for ImageNet classification. The first 300 correctly classified images from the validation dataset are sieved out for our experiment. Internal layer responses are collected during the forward pass of all the 300 images. The manipulation of layer activations is conducted in Caffe [22] environment, where random elements between the maximum and minimum values of the current layer responses are picked as noise. Such noise array will be added on the original layer activations and put back to the DNN architecture for classification. We evaluate the vulnerability of the model layer by layer. Besides internal layers, we also compare the misclassification rate with perturbed input layer. The input layer is not exactly the image, as rescaling image pixels to $[-0.5, 0.5]$ is a common preprocessing requirement. It should be mentioned that the last FC layer is not in our evaluation list. The classification probability is based on a linear calculation on the last FC outputs. Modification of FC layer will directly alter the output prediction. Since the outcome is predictable, tampering of FC layer, especially for those models with single FC layer, is easily detectable.

To investigate the degree of vulnerability of each layer, we evaluate various noise percentages (number of noise elements

per layer activation) from 0.1% to 10%. For example, 0.1% noise refers to three noise elements per channel for a feature size 56×56 ($\lfloor 0.001 \times 56 \times 56 \rfloor = 3$). Although the number of noise elements for each channel is the same, the noise values and locations of different channels are shuffled. At least one noise element will be injected into each channel in this study.

IV. SIMULATION RESULTS AND ANALYSIS

This section presents and analyzes the results of applying perturbations on internal layers of different network architectures. The results are reported based on the misclassification rate on 300 selected images.

A. Result Presentation

The misclassification rates on all evaluated models are summarized in Table I. 0.3% noise perturbation is applied onto each but the last layer of AlexNet, VGG16 and VGG19 whereas 0.1% noise perturbation is used for Inception-v1, ResNet50, ResNet101, DenseNet121, Densenet169, MobilNet-v1 and MobileNet-v2. As shown in Table I, injecting small portion ($\leq 0.3\%$) of noise into the input layer has barely any effect on misclassification. On the other hand, adding the same rate of noise into internal layers can yield at least 29% misclassification. Moreover, error rates of over 90% are observed on higher accuracy models including Inception, ResNet, DenseNet and MobileNet architectures. The results of different network structures are detailed as follows.

Fig. 5. Vulnerability of AlexNet, VGG16 and VGG19 with 0.1% and 0.3% noise injections.

Plain Network Structure. AlexNet, VGG16 and VGG19 belong to this structure as they directly stack computational layers. Fig. 5 shows the layer-wise vulnerability of this structure. With 0.1% noise disturbance indicated by dash lines, all the layers remain reliable except conv1_2 of VGG16, which shows 15% error rate. However, prominent (43.3%, 36.3% and 29.7%) misclassification can happen by injecting 0.3% noise (indicated by the solid lines) to conv5 of AlexNet, conv5_3 of VGG16 and conv5_4 of VGG19, respectively. Despite varying depth of different models, plain structure design has a common problem that the last convolutional layer of the network is most vulnerable. VGG nets have higher accuracy than AlexNet, but they are also more prone to noise perturbations. While several layers from VGG nets yield more than 15% misclassification, only conv5 of AlexNet is most apparently assailable.

Advanced Network Structure. In this category, alarming misclassification rates on Inception-v1, ResNets and DenseNets are observed. By injecting 0.1% perturbation or single noise element per channel, more than 65% of the convolution layers from all three network families can mislead the classifier with higher than 80% success rate. The first two CONV blocks of Inception-v1 are similar to plain structure, hence they are also robust against noise disturbance. Inception modules have been introduced in the remaining three CONV blocks which occupy more than 90% computational layers in Inception-v1. Extremely high misclassification rates can be observed in those three CONV blocks. In particular, CONV5 block can experience as high as 91.7% misclassification rate as shown in Table I. The results are grouped in average value for these advanced models. For example, "5a-5b 0.917" means the average misclassification rate on 5a and 5b layers is 91.7%. Unless otherwise specified, the results are reported in average value for those layers that share similar vulnerability.

Fig. 6 gives more details of the layer-wise vulnerability on ResNets with 0.1% noise. The gray solid line represents ResNet50, and the red dotted line describes ResNet101. These two networks have different layer depth as indicated in their name, hence the gray line is not continued at the positions of CONV4 block. Similar vulnerability is observed on both networks, where majority of the convolution layers experience more than 90% misclassification rate. In other words, perturbing any vulnerable layer from this architecture can subvert 90% correct predictions. Only a small number of layers are robust against noise disturbance, which is reported as "Res" layers in Table I. The "Res" layers are the element-wise addition layer, which are responsible for the residual mapping function. Residual mapping will add the data from the shortcut path to current layer activations. No feature extraction or feature squeezing is performed in these layers. An interesting pattern is found in Fig. 6, where all the vulnerable layers are in group of three consecutive layers. This pattern perfectly matches the bottleneck design structure. Therefore, we conjecture that the increased vulnerability is caused by the 1×1 feature squeezing layer. This also explains why the residual mapping layers are resilient to disturbance, as the feature information before squeezing are directly fed to them through the shortcut path.

DenseNet is an extension of ResNet, which broadcasts input features to many subsequent layers through shortcut connections. The bottleneck layer is reserved in DenseNet, which improves the computational efficiency for the 3×3 feature extraction layer. It is not surprising that four CONV blocks exhibit more than 90% misclassification rate as shown in Table I. The worst error rates are 99.5% and 99.8% on DenseNet121 and DenseNet169, respectively. "Trans" layer refers to the transition layer between the dense blocks, which are utilized to downsample the feature maps. Transition layer is found to be robust against perturbations.

Lightweight Network Structure. MobileNets are lightweight models for embedded systems. For this structure, considerable misclassifications on MobileNet-v1 and similar alarming error rate as advanced networks on MobileNet-v2 are noticed under

TABLE I

MISCLASSIFICATION RATES OF DIFFERENT NETWORK ARCHITECTURES UNDER NOISE PERTURBATION. EACH BUT THE LAST FC LAYER OF ALEXNET, VGG16 AND VGG19 IS SUBJECTED TO 0.3% NOISE PERTURBATIONS. EACH BUT THE LAST LAYER OF THE REMAINING NETWORKS IS SUBJECTED TO 0.1% NOISE PERTURBATIONS.

Network Architecture	Input Layer	CONV1 Block	CONV2 Block	CONV3 Block	CONV4 Block	CONV5 Block	FC
AlexNet	0.037	0.070	0.067	0.043	0.073	0.433	0.057 0.043
VGG16	0.040	1_1 0.060 1_2 0.333	2_1 0.067 2_2 0.170	3_1 0.063 3_2 0.050 3_3 0.133	4_1 0.050 4_2 0.050 4_3 0.107	5_1 0.063 5_2 0.083 5_3 0.363	0.060 0.043
VGG19	0.040	1_1 0.047 1_2 0.177	2_1 0.057 2_2 0.243	3_1 0.093 3_2 0.040 3_3 0.047 3_4 0.060	4_1 0.043 4_2 0.047 4_3 0.037 4_4 0.110	5_1 0.073 5_2 0.067 5_3 0.093 5_4 0.297	0.060 0.040
Inception-v1	0.057	0.060	0.056	3a-3b 0.830	4a-4e 0.892	5a-5b 0.917	N.A.
ResNet50	0.030	0.213	2a-2c 0.952 Res 0.025	3a-3c 0.949 Res 0.028	4a-4f 0.954 Res 0.047	5a-5c 0.897 Res 0.121	N.A.
ResNet101	0.053	0.240	2a-2c 0.953 Res 0.049	3a-3c 0.952 Res 0.053	4a-4b22 0.983 Res 0.047	5a-5c 0.919 Res 0.420	N.A.
DenseNet121	0.050	0.498	2.1-2.6 0.963 Trans 0.044	3.1-3.12 0.950 Trans 0.038	4.1-4.24 0.951 Trans 0.111	5.1-5.16 0.995 Trans 0.343	N.A.
DenseNet169	0.047	0.521	2.1-2.6 0.972 Trans 0.042	3.1-3.12 0.968 Trans 0.040	4.1-4.32 0.969 Trans 0.586	5.1-5.32 0.998 Trans 0.437	N.A.
MobileNet-v1	0.040	0.040	0.053	0.042	0.047	5.1-5.6 0.484 6* 0.094	N.A.
MobileNet-v2	0.040	0.062	2_1 0.434 2_2 0.291	3_1 0.920 3_2 0.049	4.3&4.7 0.040 rest 0.946	5.1-5.2 0.93 5.3 0.211 6.1-6.2 0.948 6.3-6.4 0.155	N.A.

Fig. 6. Vulnerability of ResNet50 and ResNet101 with 0.1% noise injection.

0.1% noise injection. Only the CONV5 block of MobileNet-v1 has 48.4% misclassification, other layers can resist small noise disturbance. This property is similar to plain networks, as version 1 has the same backbone of stacked layers. It should be pointed out that CONV5 block is the penultimate convolution block before the layer (6*) as CONV6. With the introduction of shortcut connection and bottleneck design, MobileNet-v2 can gain higher accuracy and smaller model size compared with its version 1 counterpart. However, its robustness also decreases dramatically. CONV3-6 blocks of MobileNet-v2 suffer more than 90% misclassification rate. Meanwhile, only a few layers (e.g., 3_2, 4.3 and 6.3) demonstrate good immunity to perturbations.

B. Observation and Analysis

Although adversarial example can mislead DNN model, effective attack algorithm requires to alter every pixel with small disturbance on the input layer [19]. According to our results, tampering internal layer is much more effective than input layer, as less than 0.3% noise elements are sufficient to subvert correct predictions. From attack perspective, interpolating less number of elements is more attractive.

Alarming misclassification rate has been observed on advanced network structures. Comparing with the layer-wise vulnerability and network structure, we believe the extensive use of 1×1 feature squeezing kernel is the key reason for robustness degradation. Narrow layer is reported to have lower capacity for feature representation [18], however, state-of-the-art models still heavily rely on the bottleneck layer for fine

tuning the architecture performance. Reducing the channel dimension can compact the feature size which efficiently reduce the computation complexity in the following feature extraction layer. However, it also drains out the redundancy of the original features. Thus, 1×1 convolution layer for dimension reduction can unconsciously remove the noise tolerance of the DNN models.

Recently, a memory trojan on DNN accelerator was reported [11]. The attack was emulated by zeroizing a random portion of the layer outputs. All the CONV layers and all the FC layers were investigated and no layer-wise vulnerability was reported. Accuracy degradation almost linear to the percentage of error injected into the FC layers was claimed. We argue contrary as we find that except the last FC layer, other FC layers are generally robust to noise disturbance as shown in Table I. The effect reported by [11] on FC layers is mainly due to the error injected into last FC layer. Manipulation on the output of last FC layer is equivalent to falsification on final classification output. Such direct modification on results is arguably not stealthy, and can be easily detected and circumvented. Comparing with 10% error injection to every convolution layers for 40% misclassification rate [11], our study reveals that only 0.1% (100x less) error injection on single convolution layer is sufficient to achieve more than 90% (2.25x more) misclassification rate. Adding constrained noise is more powerful than reset layer outputs to zero.

V. CONCLUSION

In this paper, we present a layer-wise vulnerability analysis on 10 popular DNN models under potential noise-injection based hardware attacks. Injecting 0.3% noise to the last convolutional layer of AlexNet, VGG16 and VGG19 can lead to 43.3%, 36.3% and 29.7% misclassifications. More accurate models have tendency to be more unreliable under such attacks. Misclassification rates in excess of 90% were observed on Inception-v1, ResNets and DenseNets by falsifying 0.1% or one element per channel on only one single vulnerable convolutional layer. MobileNet also suffers from this vulnerability after the introduction of bottleneck layer. Exploiting the 1×1 convolution layer before feature extraction is a widely used approach to compact model size and reduce computation complexity. However, it is found that the frequent use of 1×1 convolution layer for feature squeezing can wipe out the noise tolerance of the model and weaken the model robustness against noise injection attacks.

ACKNOWLEDGMENT

This research is supported by Singapore Ministry of Education AcRF Tier 2 Grant No. MOE-2015-T2-2-013.

REFERENCES

- [1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Neural Information Processing Syst. (NIPS)*, Lake Tahoe, Nevada, USA, Dec. 2012, pp. 1097–1105.
- [2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, Nevada, USA, Jun. 2016, pp. 770–778.
- [3] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14410–14430, 2018.
- [4] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," Banff, Canada, Apr. 2014. [Online]. Available: <http://arxiv.org/abs/1312.6199>
- [5] M. Agoyan, J.-M. Dutertre, D. Naccache, B. Robisson, and A. Tria, "When clocks fail: On critical paths and clock faults," in *Smart Card Research and Advanced Application*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 182–193.
- [6] T. Korak, M. Hutter, B. Ege, and L. Batina, "Clock glitch attacks in the presence of heating," in *Proc. 2014 Workshop on Fault Diagnosis and Tolerance in Cryptography*, Sep. 2014, pp. 104–114.
- [7] J. G. J. van Woudenberg, M. F. Witteman, and F. Menarini, "Practical optical fault injection on secure microcontrollers," in *2011 Workshop on Fault Diagnosis and Tolerance in Cryptography*, Sep. 2011, pp. 91–99.
- [8] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, "Hardware trojan attacks: Threat analysis and countermeasures," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1229–1247, Aug 2014.
- [9] Y. Liu, L. Wei, B. Luo, and Q. Xu, "Fault injection attack on deep neural network," in *Proc. 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Irvine, CA, USA, Nov 2017, pp. 131–138.
- [10] J. Clements and Y. Lao, "Hardware trojan design on neural networks," in *Proc. 2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, Sapporo, Hokkaido, Japan, May 2019, pp. 1–5.
- [11] 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 *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, Grenoble, France, March 2019, pp. 1415–1420.
- [12] J. Breier, X. Hou, D. Jap, L. Ma, S. Bhasin, and Y. Liu, "Practical fault attack on deep neural networks," in *Proc. 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: ACM, Oct. 2018, pp. 2204–2206.
- [13] 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.
- [14] 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. [Online]. Available: <http://arxiv.org/abs/1409.1556>
- [15] 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. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, Massachusetts, USA, Jun. 2015, pp. 1–9.
- [16] G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, Hawaii, July 2017, pp. 2261–2269.
- [17] 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," *CoRR*, vol. abs/1704.04861, 2017. [Online]. Available: <http://arxiv.org/abs/1704.04861>
- [18] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, USA, June 2018, pp. 4510–4520.
- [19] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. 2017 IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, USA, May 2017, pp. 39–57.
- [20] D. Su, H. Zhang, H. Chen, J. Yi, P.-Y. Chen, and Y. Gao, "Is robustness the cost of accuracy?—a comprehensive study on the robustness of 18 deep image classification models," in *Proc. European Conference on Computer Vision (ECCV)*, Munich, Germany, Sept. 2018, pp. 631–648.
- [21] 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.
- [22] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22Nd ACM International Conference on Multimedia*, ser. MM '14, Nov. 2014, pp. 675–678.