Time to leak : cross-device timing attack on edge deep learning accelerator

Won, Yoo-Seung; Chatterjee, Soham; Jap, Dirmanto; Bhasin, Shivam; Basu, Arindam

2021

Won, Y., Chatterjee, S., Jap, D., Bhasin, S. & Basu, A. (2021). Time to leak : cross-device timing attack on edge deep learning accelerator. 2021 International Conference on Electronics, Information, and Communication (ICEIC), 1-4. https://dx.doi.org/10.1109/ICEIC51217.2021.9369754

https://hdl.handle.net/10356/147150

https://doi.org/10.1109/ICEIC51217.2021.9369754

2021 International Conference on Electronics, Information, and Communication (ICEIC) | 978-1-7281-9161-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICEIC51217.2021.9369754

Time to Leak: Cross-Device Timing Attack On Edge Deep Learning Accelerator

Yoo-Seung Won
Temasek Laboratories
Nanyang Technological University
Singapore
yooseung.won@ntu.edu.sg

Soham Chatterjee
School of Electrical and Electronic Engineering
Nanyang Technological University
Singapore
soham004@e.ntu.edu.sg

Dirmanto Jap
Temasek Laboratories
Nanyang Technological University
Singapore
djap@ntu.edu.sg

Shivam Bhasin
Temasek Laboratories
Nanyang Technological University
Singapore
sbhasin@ntu.edu.sg

Arindam Basu
School of Electrical and Electronic Engineering
Nanyang Technological University
Singapore
arindam.basu@ntu.edu.sg

***Abstract*—Edge deep learning accelerators are optimised hardware to enable efficient inference on the edge. The models deployed on these accelerators are often proprietary and thus sensitive for commercial and privacy reasons. In this paper, we demonstrate practical vulnerability of deployed deep learning models to timing side-channel attacks. By measuring the execution time of the inference, the adversary can determine and reconstruct the model from a known family of well known deep learning model and then use available techniques to recover remaining hyperparameters. The vulnerability is validated on Intel Compute Stick 2 for VGG and ResNet family of models. Moreover, the presented attack is quite devastating as it can be performed in a cross-device setting, where adversary profiles constructed on a legally own device can be used to exploit the victim device with a single query and still can achieve near perfect success rate.**

***Keywords*—Timing analysis, High performance edge machine learning processing unit, Intel Compute Stick 2.**

I. Introduction

The field of Machine Learning (ML) and Deep Learning (DL) have been growing rapidly, and have been widely adopted for different applications across different disciplines. With the growing trends, more and more design strategies have been developed. This leads to the development of more advanced architectures as well as the increase of resources allocated for the training. Eventually, this trend leads to the increase in intellectual property (IP) models strategies, where different companies and research institutes keep their training data and model details undisclosed and make it available commercially on pay per use basis. Alternatively, pre-trained model might leak some information regarding the training data, which might be private and confidential, for example medical data. As such, this could also lead to potential privacy and security issue.

A. Security and Privacy in Machine Learning

The protection of deployed models from adversarial attacks as well as the protection from the extraction of model parameters is an increasing concern in the deep learning field. It has been shown that neural network weights can store information regarding the training dataset and a motivated attacker can retrieve it [1]. Furthermore, getting access to the architecture and weights of a trained network can aid attackers who might not have the same training dataset or the compute power to train a network. This is concerning since many organizations are now deploying models trained on sensitive or private data. However, these attacks are usually software based and multiple techniques to defend against such attacks have been proposed [2].

On the other hand, many different attacks compromising the security and privacy of deep learning models have been investigated. An overview on such attacks is provided in [3]. [4] surveys security of ML on edge devices. One of the main attack is the model stealing or model extraction attack. In this attack, the attacker or adversary will try to steal a copy of a remotely deployed machine learning model. In [5], the author proposed taxonomy of the model extraction attacks on machine learning.

- *Exact Extraction:* when the extracted model have same architecture and weights as the original network,
- *Functionally Equivalent Extraction:* a slightly weaker assumption, in which output of both models only have to agree to be the same for all the elements from the domain,
- *Fidelity Extraction:* In this case, the extracted model be the one that maximise the similarity function with the original model for a target distribution. For the case of functionally equivalent extraction, it achieves a fidelity of 1 on all distributions and all distance functions, and
- *Task Accuracy Extraction:* This goal is for the extracted model to match (or exceed) the accuracy of the target model.

B. Related works

The first model extraction attack was proposed by Tramer *et al.* [6] where the authors assume black box model, with no prior knowledge of the model parameter and training data. The aim is to duplicate the functionality of the target model. Hua *et al.* [7] proposed to exploit information leakage through timing (and memory) side-channels targeting DNN accelerators running in a secure enclave like SGX. Batina *et al.* [8] proposed a generic model recovery by side-channels. Their primary side-channel was electromagnetic leakage but they recovered activation functions using timing leakage. Later, Duddu *et al.* [9] show use of timing leakage to recover the model architecture. Another timing based reverse engineering attack is reported in [10]. The proposed method is to measure the timing information from the power consumption trace, during the execution of floating point multiplication operation.

In [11], the authors introduced DeepRecon, a method based cache side-channel attack. It can recover the target architecture by observing function invocations that map directly to architecture attributes of the victim network. The attack is performed on two networks (VGG19 and ResNet50). In [12], the authors performed model extraction attacks by first recovering the architecture of the network through electromagnetic emanation (EM) leakage, followed by estimation of the parameters, using active adversarial learning.

All the previous work target general purpose hardware like CPU, FPGA or microcontroller. Recently, accelerators to speed up deep learning inferences on the edge have been introduced [13] [14] [15]. In this work, we investigate the vulnerability of model extraction attack on a high performance edge deep learning processing unit using timing side channels. We have performed our experiments on Intel Compute Stick 2 (a.k.a Movidius) [13] running OpenVINO toolkit [16]. Our attack allows us to recover the model architecture with a single query to victim device with near perfect success. The attacks are conducted in cross-device setting [17], [18], considering a real-attack scenario. Once the architecture is recovered, hyperparameters can be easily recovered following techniques as proposed in [9]. In summary, our model extraction attack can be considered as task accuracy extraction, based on the taxonomy.

C. Our Contributions

The main contributions of this work are as follows:

- By characterizing the execution time on known networks, we successfully build the kernel density estimator (KDE [19]) for high performance edge machine learning processing unit (a.k.a. Movodius) [13].
- From the KDE, we investigate the cross-device scenario [17], [18] and successfully perform architecture recovery.
- Finally, we show that for well-known network architecture, the success rate of recovery is almost 100%.

The rest of the paper is organised as follows. Section 2 presents the considered adversary model and describes the methodology for the proposed attacks. Section 3 presents practical experiments for Intel Compute Stick 2 with VGG and ResNet like network. Finally, conclusions and countermeasures are discussed in Section 4.

II. Threat Model and Attack Methodology

In this paper, we consider a victim neural network $\mathcal{F}_v$ with domain input $\mathcal{X} \subseteq \mathbb{R}^n$ and output $\mathcal{Y} \subseteq \mathbb{R}^m$. The main goal of an adversary is to accurately reverse engineer the large-scale network architecture through timing information. We only consider common architectures such as VGG [20] and ResNet [21]. That is, $\mathcal{F}_v = \{\text{ResNet18}, \text{ResNet34}, \text{ResNet50}, \text{ResNet101}, \text{ResNet152}, \text{VGG11}, \text{VGG13}, \text{VGG16}, \text{VGG19}\}$.

A. Adversary Assumption

The victim has deployed a trained model for inference on the target accelerator. The adversary, acting as a client with user-level privilege, has only access to the device through conventional channels such as USB-type hardware accelerator [13] or cloud-based services such as Amazon, Google, Microsoft, BigML, and others. In [4], this is called to the API-access assumption. Moreover, software side-channels assume that an adversary also has some ability to measure side-channels through the device's legitimate outputs, *e.g.*, measure the latency of network/USB responses or the amount of traffic the device is sending to the cloud. In this paper, an adversary requests an inference on a edge device. Then, the adversary measures the timing information about the difference between input sent to the accelerator and output received from the accelerator, and has no access rights to pre-trained (and secret) model running on the accelerator.

The main goal of the adversary is that utilizing the queries to USB-type hardware accelerator [13] for inference with the secret model, we can recover the model parameters. It is assumed that the victim uses one from a list of popular models while internal hyperparameters are proprietary. We show our experiments by assuming that the model belong to widely popular VGG or ResNet family. Other models can be like included in the study.

B. Evaluation Metric

Given the timing information t_v while a victim neural network $\mathcal{F}_v$ is operated, the kernel density estimator (KDE) $\Pr_{KDE}$ over the n train set $(t_s^1, ..., t_s^n)$ from substitute neural network $\mathcal{F}_s$ can be calculated by:

$$KDE_s(t_v) := \Pr_{KDE}(t_v|\mathbf{t}_s) \tag{1}$$

For several substitute neural networks, we calculate the $KDE_s(t_v)$. The highest $KDE_s(t_v)$ can be considered as correct neural network.

For multiple m observations $(t_v^1, ..., t_v^m)$ from a victim device, we can extend the Equation 1 as below:

$$KDE_s(\mathbf{t}_v) := \prod_{i=1}^{m} \Pr_{KDE}(t_v^i|\mathbf{t}_s) \tag{2}$$

The experiments are repeated 100 times to compute the KDE.

III. Timing Side-Channel Analysis Against Edge Machine Learning Accelerator

In this section, we explain how to measure the execution time for the main target and demonstrate the common used machine learning architecture can be recovered.

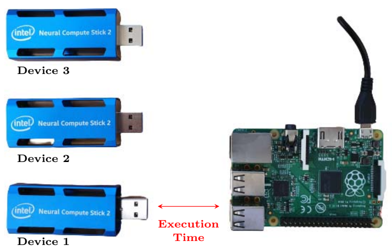

Fig. 1: Experimental Setup for the measurement of execution time on 3 different Intel Compute Stick 2.

A. Experimental Setup

To measure the execution time when the target ML inference is processed, an USB-type hardware accelerator [13] is plugged into the main processor such as laptop, desktop, or IoT device. Moreover, in order to demonstrate the consistency for execution time, we employ the cross-device attack [17], [18] as well, as shown in Figure 1.

B. Evaluation Results

Based on the setup, we measure the execution time for various VGG and ResNet architectures in three devices. We can estimate that the difference of execution time for each device is negligible if hardware accelerator is stable.

On three device, the inference time of each architecture is very similar in Figures 2 and 3 when we repeat 100 times for same architecture. Due to the I/O interrupt, we can observe that some operations get too far off the center line such as average. But, this observation is rarely observed.

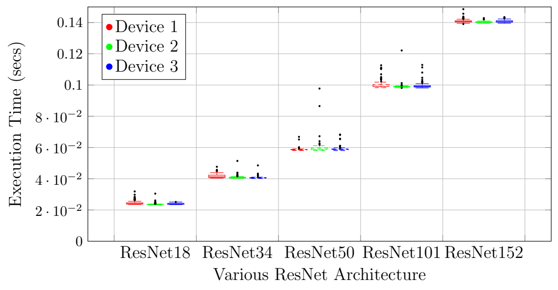

Fig. 2: Timing Side-Channel Analysis for Various ResNet Architectures.

For ResNet, we can see that the execution time scale nicely with model depth. Moreover, we observe a higher difference in time for models bigger than ResNet50. This can be attributed to the depth of conv4_x [21] which is increased from 50-layer to 101-layer.

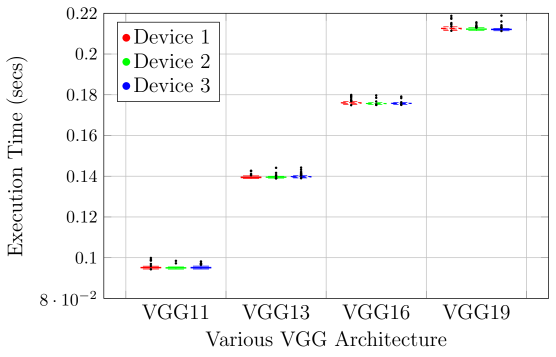

Fig. 3: Timing Side-Channel Analysis for Various VGG Architectures.

With VGG architectures, as show in Figure 3, the execution time is more or less directly proportional to the network depth. The execution time increases about 0.04-0.05 secs, with increasing length.

Overall, The execution timing of the studied architecture are unique and can be easily distinguished from one another. This is even valid on a cross-device setting, making the attack more critical. In cross-device setting, an adversary can measure the time on one device here he is the legal owner and has all access rights. later, on victim device, just by measuring the delay, the exact architecture can be determined when it belongs to a given a family of architecture.

TABLE I: Success for Various Machine Learning Architecture.

Architecture	**Device 2**	**Device 3**
ResNet18	100%	100%
ResNet34	100%	100%
ResNet50	98%	100%
ResNet101	97%	100%
ResNet152	100%	98%
VGG11	100%	100%
VGG13	99%	98%
VGG16	100%	100%
VGG19	100%	100%

Based on Section II-B, we compute the KDE value of execution time from Device 2 and 3 (victim device) after building the execution time for each architecture of Device 1 (adversary's device). We use 100 measurements per architecture from Device 1. The KDE for observations on Device 2 and 3 are made with only single observation per architecture. The experiments are repeated 100 times to compute the success rate. As shown in Table I, the success rate is close to 100%, only with a single observation from the victim device. It is well known that an adversary can easily perform reinforcement learning to recover the hyperparameter of architecture with distillation [9].

IV. Conclusion and Further Work

We have investigated the timing based vulnerability to recover architecture of commonly used machine learning models on high performance edge machine learning processing unit like Intel Compute Stick 2. The attacks are reported with close to 100% success rate with only a single observation from the victim device running the victim model. The attacks performs well in cross-device setting as well. Future work will investigate if internal details of the network architecture like number of layers, activation function, *etc* can also be recovered using timing information.

In terms of countermeasure, the standard countermeasure from side-channel field could be deployed, such as introducing random delay, inserting dummy operations during the execution of inference, such that there is no direct dependence between network architecture and execution delay. In [22], the authors proposed several countermeasures, by running extra operations as decoy process as well as obfuscated architectures. However, as this is only constructed for specific family of networks and deployed against cache timing vulnerability, whether this could prevent attacks across different network families and different side-channels might need further investigation.

Acknowledgment

This research is supported in parts by the National Research Foundation, Singapore, under its National Cybersecurity Research & Development Programme / Cyber-Hardware Forensic & Assurance Evaluation R&D Programme (Award: NRF2018NCR-NCR009-0001).

References

[1] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning models that remember too much," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 587–601.

[2] H. X. Y. M. Hao-Chen, L. D. Deb, H. L. J.-L. T. Anil, and K. Jain, "Adversarial attacks and defenses in images, graphs and text: A review," *International Journal of Automation and Computing*, vol. 17, no. 2, pp. 151–178, 2020.

[3] N. Papernot, P. D. McDaniel, A. Sinha, and M. P. Wellman, "Sok: Security and privacy in machine learning," in *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, London, United Kingdom, April 24-26, 2018*. IEEE, 2018, pp. 399–414. [Online]. Available: https://doi.org/10.1109/EuroSP.2018.00035

[4] M. Isakov, V. Gadepally, K. M. Gettings, and M. A. Kinsy, "Survey of attacks and defenses on edge-deployed neural networks," in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2019, pp. 1–8.

[5] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot, "High accuracy and high fidelity extraction of neural networks," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020.

[6] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 601–618.

[7] W. Hua, Z. Zhang, and G. E. Suh, "Reverse engineering convolutional neural networks through side-channel information leaks," in *Proceedings of the 55th Annual Design Automation Conference, DAC 2018, San Francisco, CA, USA, June 24-29, 2018*. ACM, 2018, pp. 4:1–4:6. [Online]. Available: https://doi.org/10.1145/3195970.3196105

[8] L. Batina, S. Bhasin, D. Jap, and S. Picek, "CSI NN: reverse engineering of neural network architectures through electromagnetic side channel," in *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, N. Heninger and P. Traynor, Eds. USENIX Association, 2019, pp. 515–532. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/batina

[9] V. Duddu, D. Samanta, D. V. Rao, and V. E. Balas, "Stealing neural networks via timing side channels," *arXiv preprint arXiv:1812.11720*, 2018.

[10] G. Dong, P. Wang, P. Chen, R. Gu, and H. Hu, "Floating-point multiplication timing attack on deep neural network," in *2019 IEEE International Conference on Smart Internet of Things (SmartIoT), Tianjin, China, August 9-11, 2019*. IEEE, 2019, pp. 155–161. [Online]. Available: https://doi.org/10.1109/SmartIoT.2019.00032

[11] S. Hong, M. Davinroy, Y. Kaya, S. N. Locke, I. Rackow, K. Kulda, D. Dachman-Soled, and T. Dumitraş, "Security analysis of deep neural networks operating in the presence of cache side-channel attacks," *arXiv preprint arXiv:1810.03487*, 2018.

[12] H. Yu, H. Ma, K. Yang, Y. Zhao, and Y. Jin, "Deepem: Deep neural networks model recovery through em side-channel information leakage."

[13] "Neural Compute Stick 2," https://software.intel.com/content/www/us/en/develop/hardware/neural-compute-stick.html.

[14] "EdgeTPU Accelerator," https://coral.ai/docs/accelerator/datasheet/.

[15] "EdgeTPU Dev Board," https://coral.ai/docs/dev-board/datasheet/.

[16] "Intel OpenVINO Toolkit," https://software.intel.com/content/www/us/en/develop/tools/openvino-toolkit.html.

[17] S. Bhasin, A. Chattopadhyay, A. Heuser, D. Jap, S. Picek, and R. R. Shrivastwa, "Mind the portability: A warriors guide through realistic profiled side-channel analysis," in *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020.

[18] D. Das, A. Golder, J. Danial, S. Ghosh, A. Raychowdhury, and S. Sen, "X-deepsca: Cross-device deep learning side channel attack," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.

[19] S. Weglarczyk, "Kernel density estimation and its application," in *ITM Web of Conferences*, vol. 23. EDP Sciences, 2018, p. 00037.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1409.1556

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[22] S. Hong, M. Davinroy, Y. Kaya, S. N. Locke, I. Rackow, K. Kulda, D. Dachman-Soled, and T. Dumitras, "Security analysis of deep neural networks operating in the presence of cache side-channel attacks," *CoRR*, vol. abs/1810.03487, 2018. [Online]. Available: http://arxiv.org/abs/1810.03487