Characterizing Side-Channel Leakage of DNN Classifiers though Performance Counters

Saikat Majumdar, Mohammad Hossein Samavatian, Radu Teodorescu
Department of Computer Science and Engineering
The Ohio State University, Columbus, OH, USA
{majumdar.42, samavatian.1, teodorescu.1}@osu.edu

2022 IEEE International Symposium on Hardware Oriented Security and Trust (HOST) | 978-1-6654-8532-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/HOST54066.2022.9839882

***Abstract*—Rapid advancements in Deep Neural Networks (DNN) have led to their deployment in a wide range of commercial applications. DNN classifiers are powerful tools that drive a broad spectrum of important applications, from image recognition to autonomous vehicles. Like other applications, they have been shown to be vulnerable to side-channel information leakage. There have been several proof-of-concept attacks demonstrating the extraction of their model parameters and input data. However, no prior study has examined the possibility of using side-channels to extract the DNN classifier's decision or output. In this initial study, we aim to understand if there exists a correlation between the output class selected by a classifier and side-channel information collected while running the inference process on a CPU. Our initial evaluation shows that with the proposed approach it is possible to accurately recover the output class for model inputs via multiple side-channels: primarily power, but also branch mispredictions and cache misses.**

I. Introduction

Deep neural networks (DNNs) have emerged as the backbone of an increasing number of important applications. Some of these applications benefit from the deployment of sophisticated DNN models into so-called edge devices such as smartphones, smart home devices, autonomous driving systems, healthcare, and many others [15]. Applications such as autonomous driving, are safety-critical and their failure can endanger lives.

Unfortunately, DNNs are known to be vulnerable to a variety of security threats. DNN models are part of the intellectual property of the entity that deploys them, are expensive to train, and maybe trained using proprietary data. Prior work has shown that DNNs are vulnerable to side-channel leakage that targets extracting the DNN model parameters [3], [9], [14]. However, no prior work has examined the vulnerability of DNN model decisions or outputs to such attacks.

In this work, we study the sensitivity of DNN inference to leakage of *classification results*. In this initial study, we aim to understand if there exists a correlation between the output class selected by a classifier and side-channel information collected while running the inference process on a CPU. To that end we examine three types of processor activity that can be observed through side-channels: power consumption, L1-Cache misses and Branch misprediction events, captured through CPU performance counters.

This work was funded in part by the National Science Foundation under awards CCF-2028944 and CCF-1629392 and by the Air Force Research Laboratory under the Assured and Trusted Microelectronics Solutions award FA8650-20-C-1719.

We target both the VGG16 image classifier with the ImageNet dataset and the DenseNet classifier with the CIFAR-10 dataset. We train a simple time-series-based classifier on power traces, branch prediction events, and L1 cache misses obtained from performance counters. Using this classifier we achieve 93%, 84%, and 76% accuracy at identifying the correct output class when analyzing traces from VGG16 with unseen inputs. DenseNet is somewhat less vulnerable, but we still achieve up to 50% accuracy. This shows that the DNN output is also potentially vulnerable to side-channel leakage, and further investigation is needed to understand which DNN models are vulnerable to this type of leakage and how it can be mitigated. To our best knowledge, this is the first work to demonstrate this vulnerability.

II. Background and Related Work

DNN implementations, e.g., in healthcare applications, use models which have been trained using private data and are considered the intellectual property (IP) of the organizations training them. Shokri et al. [17] has shown that ML models could leak sensitive information about the individual data records over which the model was trained. Additionally, the inputs to the model must also be protected from being compromised for obvious security concerns.

Every physical implementation generates unintentional side-channel emissions such as timing delay, power consumption, or electromagnetic emanation (EM), which can be observed and analyzed. This process can eventually disclose internal data from the system and effectively leak private information. Physical side-channels rely on the transistor switching activity of a computing device and hence are dependent on processing internal states and processed data. By statistically analyzing the measured side-channel data for each internal state together with a hypothesis on the processed data, an attacker can deduce some intermediate computation states. This allows the extraction of information on the processed data that would be otherwise inaccessible. Side Channel Analysis (SCA) has enabled the recovery of secret keys in cryptographic algorithms, which are otherwise considered mathematically secure [2].

Prior SCA attacks fall under two categories: (1) Simple Power/EM Analysis (SPA and SEMA, respectively), which uses coarse-grained correlations in power/EM emanations for

deducing the secret value; and (2) Differential Power/EM Analysis (DPA and DEMA) an advanced technique that requires statistical analyses on many traces collected from sets of operations to deduce the secret value through fine-grained data dependencies in power/EM signatures [11]. Batina et al. [3] recovered the complete model architecture of a DNN through EM side-channels using correlation power analysis (CPA) [5]. There have also been a few attacks targeting the recovery of network inputs using background model recovery and template matching techniques [7].

Side-Channel Attacks (SCA) have traditionally been used to attack cryptographic implementations of theoretically secure algorithms in order to recover the secret key [4]. However, side-channels analyses are more general and have been successfully applied to other targets in different scenarios, including DNNs which are vulnerable to physical SCAs. These attacks aim to either reverse engineer the network architecture by recovering various model hyperparameters (like activation function, number of layers, neurons, etc), model weights [3], [14] or recover the network inputs [7], [19]. Hua et al. [9] reverse engineered two popular Convolutional NNs (AlexNet and SqueezeNet) using memory and timing side-channel leakages from off-chip memory access patterns. Ours is the first study to examine the leakage of the classification output, which we study through multiple side-channels.

Most modern microprocessors contain a set of special-purpose registers to measure hardware-related activities known as Hardware Performance Counter (HPC) [6], which collect detailed performance information such as CPU cycles, cache misses, and branch mispredictions of an application that is running on the system. Some attacks [4] have also leveraged these performance counters for compromising the security of the system.

III. Threat Model

We assume that the attacker receives side-channel information from the execution of the DNN and that attacker is capable of measuring HPC and power side-channel information leaked from the implementation of the DNN without interrupting the normal execution. However, no information is assumed to leak from other sources. We also assume that the attacker has no prior knowledge of the model input but has information about the model architecture. In practice, AI developers usually design their models based on the existing, pre-trained, and publicly available architectures. The models are fine-tuned based on their own training sets. Prior work has already shown that it is possible to reveal the actual model architecture from real-world applications. The scope of this work is only related to performance-counter based side-channel techniques. We demonstrate that our proposed technique is easier to perform and the data extraction process is much simpler compared to CPA, DPA, and other statistical attacks that require larger amounts of data. We evaluate two popular deep convolutional neural networks, VGG16 [18] and DenseNet [10], on the ImageNet [13] and CIFAR [12] datasets because of their wide applicability to popular applications.

IV. Approach and Methodology

A. Monitoring Power/Energy Consumption

We use Intel's Running Average Power Limit (RAPL) [16] to capture energy consumption while running DNN inference workloads. RAPL allows monitoring of energy across multiple domains including the CPU cluster, attached DRAM, and on-chip GPU. We focus on the power plane *PP0* which contains the energy estimates of the CPU cores. The RAPL energy counters are read through model-specific registers (MSRs), accessed directly using the *msr* drivers in the Linux kernel. Listing 1 shows the code used to access the counters for a specific RAPL domain using the *msr* driver in Intel's Kaby Lake architecture.

Listing 1: Reading RAPL (PP0) package energy consumption.

```
int fd = open("/dev/cpu/0/msr", O_RDONLY);
uint64_t msr_data;

/*MSR_INTEL_PP0_ENERGY_STATUS
is at address 0x639*/
if ( pread(fd, &msr_data, sizeof msr_data, 0x639)
    != sizeof msr_data ) {
        perror("rdmsr:pread");
        exit(127);
}

/*Kaby Lake has 61 microjoules of energy units*/
printf("%lf\n", (double) (msr_data * 0.000061));
```

B. Monitoring Hardware Performance Counters (HPCs)

We also simultaneously measure the *Total L1-Cache Misses* and *Branch Mispredictions* of the DNN model during inference using Hardware Performance Counters. Hardware Performance Counters (HPC) are a set of specialized registers built into most modern microprocessors to store the counts of various low-level hardware events. We use the *Performance API (PAPI)* [8] tool to gain access to these performance counters. PAPI provides a simple high level and fully programmable interface for accessing the available HPCs. The events that can be monitored include a wide range of performance-related architectural features like: executed cycles, number of instructions, cache hits/misses, branch mispredictions, floating-point operations among many others which are defined in groups called *EventSets*. PAPI uses the Linux/x86 Performance Monitoring counters otherwise known as *perfctr* which are available in many modern x86-class processors.

C. DNN Datasets and Models

We evaluate two popular convolution-based DNN models and two datasets (Figure 1).

1. **DenseNet and CIFAR-10**: The CIFAR-10 [12] dataset consists of 60,000 32×32 color images in 10 classes, with 6K images per class. There are 50K training images and 10K test images. We use a DenseNet [10] model to evaluate the CIFAR-10 dataset.

2. **VGG16 and ImageNet**: The ImageNet [13] project is a large visual database designed for use in visual object recognition software research. The most used subset of ImageNet is the ImageNet Large Scale Visual Recognition Challenge

(ILSVRC) image classification and localization dataset. This dataset spans 1K object classes and contains 1,281K training images, 50K validation images and 100K test images. We use VGG16 [18] to classify the ImageNet dataset.

We perform our experiments on a total of 10K images consisting of 100 Classes from the ImageNet Dataset and a total of 5K images consisting of 10 Classes for the CIFAR-10 dataset. For each image from each dataset we collect a total of 5 different traces for each of the aforementioned side-channels.

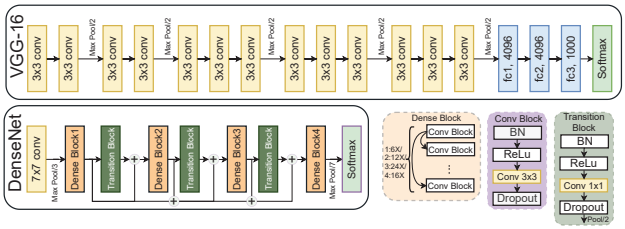

Fig. 1: VGG16 and DenseNet classifier architectures.

D. Side-Channel Leakage Assessment

Most DNNs have a final dense layer, the output of which goes through a softmax activation function, producing the output probability for every class. We focus on collecting and analyzing side-channel information while executing inference through this final layer of the DNN. We investigate the existence of side-channel leakage of the inference output by training a simple classifier based on time series forest (TSF), using the collected traces and the known output classes they belong to. We then test the TSF models on traces unseen during training.

The TSF splits the input data into a given number of windows and generates three features for each window: the mean, standard deviation, slope, and then trains the model based on these features. It, therefore, captures the characteristic interval features for various segments of the data using simple, interpretable statistical methods which are very effective and robust for time series classification and are useful for distinguishing time series for different classes. The input to the TSF is in the form of (X, y) where X is the trace of time series data represented as $X = (x_1, x_2, x_3, ..., x_k)$ collected over a time interval $T = (t_1, t_2, t_3, ...t_k)$ and where each element $x_i = \delta(t_i)$. y is the ground truth label for each input.

Figure 2 illustrates the process for capturing three different side-channels (power, cache misses, and branch mispredictions) and training a TSF-based classifier for each channel type.

E. Evaluation System

We conduct our experiments on an Intel Core-i5 CPU@3.00GHz CPU using TensorFlow version 1.14 [1] running on Linux-based Ubuntu 18.04 Operating System.

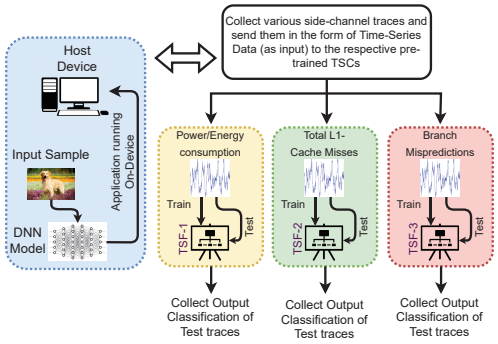

Fig. 2: Side-channel trace capture and analysis with pre-trained adversarial TSF-based classifiers.

V. Results

Figure 3 shows the classification accuracy of the three TSF classifiers, trained on models using three different sets of output classes (10, 50, and 100) from the ImageNet dataset. We show average accuracy for unseen traces that have not been used to train the TSF. We also show the accuracy variance as error bars.

The Power/Energy side-channel shows very high accuracy at close to 93% for VGG16 with 10 output classes. As we increase the number of classes the average accuracy falls, as it becomes more challenging for the TSF classifier to differentiate between outputs. However, even for 100 classes, the adversarial classifier achieves $> 50\%$ accuracy, compared to 1% for random guess. For the Cache and Branch side-channels, accuracy is lower, but still very good, at 84% and 76% accuracy respectively on 10 classes. The accuracy decreases to 46.25% and 37.11% respectively for 100 classes.

In order to verify if the three side-channels can provide complementary leakage information we also examine the accuracy of a *Combined* adversarial classifier that uses an "Oracle" to choose which of the three classifiers to use for a given input. The *Combined* classifier accuracy ranges between 99% accuracy on 10 classes to $> 75\%$ for 100 classes. This demonstrates that the three side-channels can be combined to improve the accuracy of the adversarial classification.

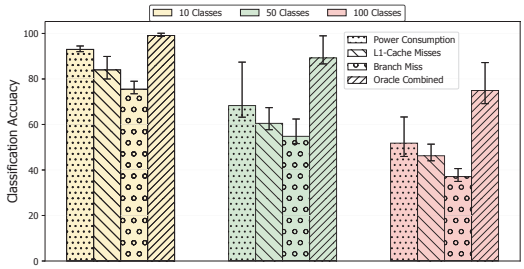

Fig. 3: Classification accuracy of the adversarial TSFs for different side-channels on the ImageNet dataset.

In the case of CIFAR-10 dataset, the input images are almost

49 × smaller compared to ImageNet data. Hence, the inference execution time of the model is much shorter which leads to fewer samples in the collected traces. We, therefore, explore the possibility of incorporating side-channel leakages from other layers beyond just the last layer to pre-train the TSFs. Figure 4 shows correlation of different side-channel leakages when incorporating more layers. Compared to just collecting traces from the last layer we notice that the classification accuracy for the Cache miss side-channel increases almost $2.18\times$ and the overall accuracy of the Combined result increases to 62.12% when using the last 30 layers of the DenseNet model.

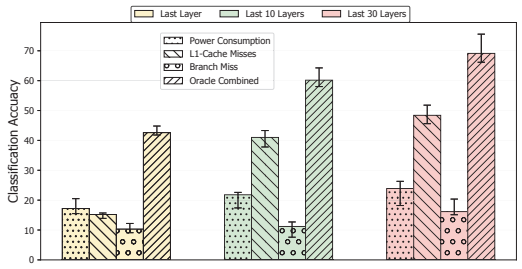

Fig. 4: Classification accuracy of the adversarial TSFs for different side-channels on the CIFAR-10 dataset trained using multiple layers.

A. Impact of Training TSFs on Multiple Traces

In order to examine the impact of measurement noise on the accuracy of the TSF classifiers, we generate multiple traces while running the same set of inputs. Even though the input sets are the same, variability in the execution environment and measurement error leads to variation side-channel output. To account for this variation we use multiple traces to train the TSF classifiers. Table I shows the impact of training with multiple traces on TSF classification accuracy. We show the average classification accuracy for each side-channel for 50 classes from the ImageNet set. We can see that training on 2 traces helps improve accuracy to some extent, but training on more than two brings no significant benefits. This indicates that measurement noise is well controlled.

TABLE I: Impact of training TSFs on variable number of traces.

Side-channel	Total # of Traces used			
	1	2	3	4
Power Consumption	61.7%	68.3%	68.9%	68.8%
L1-Cache Misses	52.85%	60.5%	59.95%	61.21%
Branch Misprediction	51.25%	54.8%	55.3%	55.1%

VI. Conclusion and Future Works

In conclusion, this initial study shows that there exists a strong correlation between the output class selected by a classifier and multiple side-channels collected while running the inference process on a CPU. Further investigation is needed to understand why this correlation exists, why some models are more vulnerable than others, and what can be done to defend against this type of leakage. We are currently investigating additional DNN models and extending this study to other platforms, including GPUs and FPGAs.

References

[1] "Tensorflow," https://www.tensorflow.org/. [Online]. Available: https://www.tensorflow.org/

[2] U. Banerjee, L. Ho, and S. Koppula, "Power-based side-channel attack for AES key extraction on the ATMega 328 microcontroller." MIT CSAIL computer systems security group, 2015. [Online]. Available: http://css.csail.mit.edu/6.858/2015/projects/utsav-lisayz-skoppula.pdf

[3] 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 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 515–532.

[4] S. Bhattacharya and D. Mukhopadhyay, "Who watches the watchmen?: Utilizing performance monitors for compromising keys of RSA on Intel platforms," in *CHES*. Springer, 2015, pp. 248–266.

[5] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems - CHES 2004*, M. Joye and J.-J. Quisquater, Eds. Springer Berlin Heidelberg, 2004, pp. 16–29.

[6] J. Demme, M. Maycock, J. Schmitz, A. Tang, A. Waksman, S. Sethumadhavan, and S. Stolfo, "On the feasibility of online malware detection with performance counters," *SIGARCH Comput. Archit. News*, vol. 41, no. 3, p. 559–570, jun 2013.

[7] 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)*, 2019, pp. 155–161.

[8] J. Dongarra, K. London, S. Moore, P. Mucci, and D. Terpstra, "Using PAPI for hardware performance monitoring on linux systems," in *Conference on Linux Clusters: The HPC Revolution*, Linux Clusters Institute. Linux Clusters Institute, 2001-06 2001.

[9] 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*, ser. DAC '18. New York, NY, USA: Association for Computing Machinery, 2018.

[10] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2017.

[11] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology — CRYPTO' 99*, M. Wiener, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 388–397.

[12] A. Krizhevsky, "Learning multiple layers of features from tiny images," *University of Toronto*, 05 2012.

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

[14] S. Maji, U. Banerjee, and A. P. Chandrakasan, "Leaky nets: Recovering embedded neural network models and inputs through simple power and timing side-channels—attacks and defenses," *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 12 079–12 092, 2021.

[15] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for IoT big data and streaming analytics: A survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2923–2960, 2018.

[16] J. Phung, Y. C. Lee, and A. Y. Zomaya, "Modeling system-level power consumption profiles using RAPL," in *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, 2018, pp. 1–4.

[17] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 3–18.

[18] 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.

[19] L. Wei, B. Luo, Y. Li, Y. Liu, and Q. Xu, "I know what you see: Power side-channel attack on convolutional neural network accelerators," in *Proceedings of the 34th Annual Computer Security Applications Conference*, ser. ACSAC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 393–406.