How Secure are Deep Learning Algorithms from Side-Channel based Reverse Engineering?

Manaar Alam
Indian Institute of Technology Kharagpur
Kharagpur, West Bengal, India.
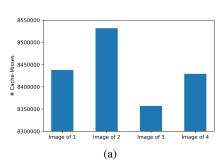alam.manaar@iitkgp.ac.in

Debdeep Mukhopadhyay
Indian Institute of Technology Kharagpur
Kharagpur, West Bengal, India.
debdeep@iitkgp.ac.in

Abstract

Deep Learning algorithms have recently become the de-facto paradigm for various prediction problems, which include many privacy-preserving applications like online medical image analysis. Presumably, the privacy of data in a deep learning system is a serious concern. There have been several efforts to analyze and exploit the information leakages from deep learning architectures to compromise data privacy. In this paper, however, we attempt to provide an evaluation strategy for such information leakages through deep neural network architectures by considering a case study on Convolutional Neural Network (CNN) based image classifier. The approach takes the aid of low-level hardware information, provided by Hardware Performance Counters (HPCs), during the execution of a CNN classifier and a simple hypothesis testing in order to produce an alarm if there exists any information leakage on the actual input.

1 Introduction

Over the past few years, we have seen an outburst of deep learning research in both industry and academia, as it considerably outperforms prior machine learning based techniques in a wide variety of domains, such as image recognition [1], machine translation [2, 3], speech processing [4], etc. Convolutional Neural Networks (CNN) have been widely used successfully for image recognition tasks, starting from identifying plant and animal species [5] to autonomous driving [6]. CNN is also recently being used for many other privacy-preserving applications like online medical image analysis [7, 8], in which the data privacy of any user requires utmost attention. There have been some recent attempts of reverse engineering and retrieving the parameters of Neural Network by side-channel information [9, 10, 11] in order to steal the model and jeopardize the privacy. Wei *et al.* [12] presented an approach which goes one step further even to determine the input image of an FPGA-based CNN implementation by observing power side-channel. This growing efforts to reverse engineer the CNN inputs would directly hamper the privacy of users, which could lead to severe issues, like discrimination, safety violations, etc. Moreover, it could lead to substantial economic losses for companies if such private data which is used to train the CNNs is revealed to other parties, which could include competitors. Hence, in this paper, we pursue an effort to develop an evaluation strategy for such divulging of neural network inputs. The target scenario in our approach is when a CNN is executing on a standard desktop, and the *evaluator* is expected to perform a dynamic evaluation and throw alarms when it detects possibilities of such leakages when the CNN classifies its inputs. The tool that the *evaluator* employs for this detection is a set of registers which are called Hardware Performance Counters (HPCs), present in most of the computing platforms, ranging from workstations to embedded processors. The data acquired from the HPCs are run-time monitored by the *evaluator*, which computes t-statistics of the data to notify that the ongoing CNN classifier is emanating side channel information which is significant for adversaries to be able to determine the input even treating the CNN implementation as a black-box.
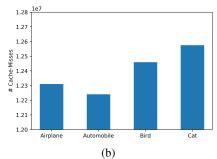
Figure 1: The average number of `cache-misses` during the classification of different categories for **a)** MNIST and **b)** CIFAR-10 dataset

Contribution

We strive to provide an evaluation strategy in order to measure private information leakages, i.e., the actual input to the deep neural network architecture during its prediction operation using readily available Hardware Performance Counters and basic hypothesis testing methodology, which to the best of our knowledge has not been attempted so far.

2 Motivation behind the Work

The process of classification by any DNN based classifier consists of a series of multiplication and addition operations that it executes on the computing environment (i.e., CPUs or GPUs)*. It is a well established fact in the literature that the execution of any process on the CPU leaks valuable side-channel information through processor cache, branch predictor unit and other low-level hardware activities [13]. The motivation behind this work is to explore the possibility of private information leakages in terms of these hardware events during the classification operation of a CNN before deploying it in a large-scale application.

In order to demonstrate the idea of information leakage we consider two publicly available image datasets **MNIST** [14] and **CIFAR-10** [15] in our study. Most images in these datasets have little or no clutter, and the classification objects tend to be centered in each of the images. Our main speculation from these datasets is that the effect of CNN operations on the hardware activities will be different for different categories, as different images belonging to a particular class activates a set of neuron in the CNN, which might not get activated for other images belonging to a different class. The activation and inactivation of these neurons influence the CNN operation affecting CPU cache, branch predictor and other units differently for different categories. In order to support the claim we monitor the average number of `cache-misses` (The measurement of these hardware activities is discussed in Section 3 and Section 4 in more details) for both the dataset while classifying each category through a CNN model and present the result in Figure 1[†]. The figure shows that the average number of `cache-misses` is different for different categories showing a possible venue for information leakage which depends on the input. The observation in Figure 1 motivates us to use the information provided by the low-level hardware events in order to develop our evaluation framework. In the next section, we present a brief discussion on how these hardware activities can be measured in a computing environment using HPCs.

3 Measuring Hardware Activities using Hardware Performance Counters

Hardware Performance Counters (HPCs) are a set of special purpose registers, which are built into most of the modern microprocessor's Performance Monitoring Unit (PMU) to dynamically observe the hardware related activities in a computing environment. These registers can be programmed easily to collect the number of occurrences of different micro-architectural events (like cache misses,

---

*In this work, we focus on the CPU implementation of an image classifier based on CNN.

[†]Without loss of generality, we present the result for four different categories for both the datasets.
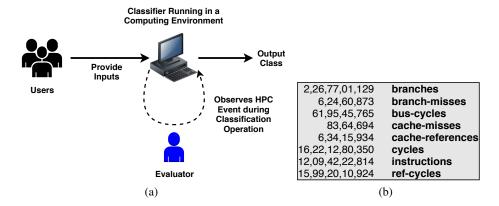
Figure 2: **Function of the *Evaluator*: a)** The *Evaluator* can only monitor the hardware events of the computing environment during the classification operation for a *User*. **b)** The values of different hardware events during the classification of a sample MNIST image. The *Evaluator* does not know the input image but can obtain these values.

branch mispredictions, retired instructions, etc.) during the execution of a program in the processor. The advantage of these performance counters is that they can be accessed very fast without affecting or slowing down any software execution. HPCs can be monitored dynamically using the `perf` tool, available in Linux kernels 2.6.31 and above, which can be invoked with administrative privilege to access these performance counters with very low granularity. The range of HPC events those can be monitored using the perf tool is more than 1000 depending on the processor Instruction Set Architectures. However, in most of the Linux based systems, the perf tool is limited to observing a maximum of 6 to 8 hardware events in parallel because of the restrictions in the number of built-in HPC registers. Moreover, some of the events are not even supported by all the processors. Hence, we have experimented with some of the basic hardware events which are supported accross the processors. The command to monitor a particular HPC event for a specific process is as follows:

```
perf stat -e <event_name> -p <process_id>
```
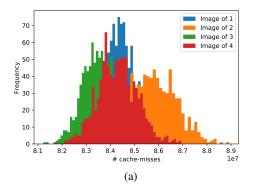
These HPCs are widely used as a source of side-channel [16, 17, 18] in order to compromise the security of several mathematically elegant cryptographic encryption algorithms. In the next section, we present the methodology to evaluate the privacy of a CNN model in the presence of information leakages in the form of HPCs.

4 Methodology for Evaluation

Let us consider a scenario where a CNN model, trained on the private information, is executed in a computing environment as shown in Figure 2(a). A group of *User* can access this model to get predictions on their respective inputs. Let us also consider the *Evaluator* who is not provided with the details of the CNN model but can monitor the HPCs during the execution of the model by its process id as discussed in Section 3. The *Evaluator* can only get information as shown in Figure 2(b), which indicates the quantitative values for different hardware events in a single classification operation. The evaluator, having the administrative privilege, can be employed to observe the HPC events for different category of images. The operations of the evaluator are as follows:

1. It monitors different HPC events in parallel during the classification operation of different categories of input images, considering each category individually. Thus, resulting in distributions of different HPC events for each class of inputs.

2. It employs a hypothesis testing methodology by computing t-statistics on the distributions of same HPC events for different categories.

The evaluator raises the alarm if the *null hypothesis* is rejected, which signifies that the distributions are different. If the distributions of an HPC event for different inputs are not distinguishable from
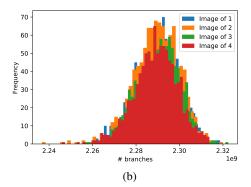
Figure 3: Distributions of HPC events **a)** `cache-misses` and **b)** `branches` during the testing operation for different categories of images present in MNIST dataset.

each other, we say that an adversary will not be able to exploit this side-channel information in order to uncover the private input images, thus indicating an efficient implementation of the CNN model.

In the next section, we evaluate the security of a CNN model considering MNIST and CIFAR-10 dataset as mentioned previously.
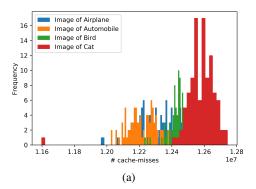
5   Results

5.1   Experimental Setup

We have implemented two CNN models for MNIST and CIFAR-10 dataset using the `tensorflow` library for our evaluation. We executed the models in Intel Xeon E5-2690 CPU having Ubuntu 18.04 with 4.15.0-36-generic kernel. All the hardware related activities are measured using the `perf` tool as mentioned previously.

5.2   Case Study on MNIST

We monitored different hardware events during the prediction of each categories for MNIST dataset. We observed that some of the events can produce different distributions for different categories. The distributions of events `cache-misses` and `branches` for all the test images belonging to different categories are shown in Figure 3. If we apply t-test on the distributions shown in Figure 3(a), we can easily distinguish them. The t and p values for the tests are shown in Table 1. The t-test is conducted with 95% confidence interval. $t_{i,j}$ signifies the t-test on the distribution for category i and category j. The **bold** faced results indicates that the two categories can be distinguished using the t-test. All the distributions shown in Figure 3(b) can not be distinguished using the t-test, but some of the categories can be. The results are presented in Table 1, which shows that the p-values are significantly higher for most of the results. Hence, the event `cache-misses` leaks valuable information on all the input categories and the event `branches` can be exploited to distinguish the inputs belonging to category 2 and category 3, which triggers the evaluator to raise an alarm.

5.3   Case Study on CIFAR-10

We also perform the same experimentation for CIFAR-10 dataset. The distributions of events `cache-misses` and `branches`, in this case, are shown in Figure 4. The results of t-tests for the distributions shown in Figure 4(a) and Figure 4(b) are presented in Table 2. In this case also we can distinguish between all the distributions if we consider the event `cache-misses` and the distributions of category 1 and category 3 can be distinguished using the evant `branches`. This triggers the evaluator to raise an alarm about the information leakage related to the input image.
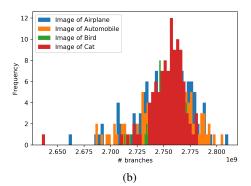
4

Figure 4: Distributions of HPC events **a)** `cache-misses` and **b)** `branches` during the testing operation for different categories of images present in CIFAR-10 dataset.

Table 1: Results of t-test on the distributions obtained from the HPC events `cache-misses` and `branches` for MNIST dataset

	cache-misses		branches	
	t-values	p-values	t-values	p-values
$t_{1,2}$	**-21.8166**	**≈0**	0.4303	0.6669
$t_{1,3}$	**-25.7566**	**≈0**	1.6565	0.0977
$t_{1,4}$	**2.5334**	**0.0113**	0.9537	0.3403
$t_{2,3}$	**40.5268**	**≈0**	**-2.0064**	**0.0449**
$t_{2,4}$	**22.6505**	**≈0**	0.4941	0.6212
$t_{3,4}$	**-20.9758**	**≈0**	**2.5435**	**0.0110**

Table 2: Results of t-test on the distributions obtained from the HPC events `cache-misses` and `branches` for CIFAR-10 dataset

	cache-misses		branches	
	t-values	p-values	t-values	p-values
$t_{1,2}$	**4.4643**	**0.0001**	-0.8796	0.3801
$t_{1,3}$	**11.0415**	**≈0**	**2.0810**	**0.0392**
$t_{1,4}$	**-16.3093**	**≈0**	-1.7474	0.0823
$t_{2,3}$	**-16.9589**	**≈0**	-1.0332	0.3032
$t_{2,4}$	**-21.2428**	**≈0**	-0.7535	0.4521
$t_{3,4}$	**-8.4637**	**≈0**	0.2997	0.7647

6   Conclusion and Future Work

In this work, we presented a strategy to evaluate the data privacy of deep neural network architectures with readily available tools. We took the aid of low-level HPC events and t-test in designing the evaluation strategy. We presented the result based on a CNN based image classifier on two publicly available datasets, MNIST and CIFAR-10. Our evaluation tool highlights the need for designing CNN architectures with indistinguishable CPU footprints while classifying different image categories in order to implement a privacy preserving classifier. In a future scope of this work, we would also like to explore the vulnerabilities in other deep learning models with different application scenarios.

References

[1] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[2] Karl Moritz Hermann and Phil Blunsom. Multilingual models for compositional distributed semantics. *arXiv preprint arXiv:1404.4641*, 2014.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[5] Guobin Chen, Tony X Han, Zhihai He, Roland Kays, and Tavis Forrester. Deep convolutional neural network based species recognition for wild animal monitoring. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 858–862. IEEE, 2014.

[6] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2722–2730, 2015.

[7] Nima Tajbakhsh, Jae Y Shin, Suryakanth R Gurudu, R Todd Hurst, Christopher B Kendall, Michael B Gotway, and Jianming Liang. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE transactions on medical imaging*, 35(5):1299–1312, 2016.

[8] Syed Muhammad Anwar, Muhammad Majid, Adnan Qayyum, Muhammad Awais, Majdi Alnowami, and Muhammad Khurram Khan. Medical image analysis using convolutional neural networks: A review. *Journal of medical systems*, 42(11):226, 2018.

[9] Weizhe Hua, Zhiru Zhang, and G Edward Suh. Reverse engineering convolutional neural networks through side-channel information leaks. In *Proceedings of the 55th Annual Design Automation Conference*, page 4. ACM, 2018.

[10] Mengjia Yan, Christopher Fletcher, and Josep Torrellas. Cache telepathy: Leveraging shared resource attacks to learn dnn architectures. *arXiv preprint arXiv:1808.04761*, 2018.

[11] Lejla Batina, Shivam Bhasin, Dirmanto Jap, and Stjepan Picek. Csi neural network: Using side-channels to recover your artificial neural network information. *IACR Cryptology ePrint Archive*, 2018:477, 2018.

[12] Lingxiao Wei, Yannan Liu, Bo Luo, Yu Li, and Qiang Xu. I know what you see: Power side-channel attack on convolutional neural network accelerators. *arXiv preprint arXiv:1803.05847*, 2018.

[13] Qian Ge, Yuval Yarom, Frank Li, and Gernot Heiser. Your processor leaks information-and there's nothing you can do about it. *arXiv preprint arXiv:1612.04474*, 2016.

[14] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

[15] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 2014.

[16] Leif Uhsadel, Andy Georges, and Ingrid Verbauwhede. Exploiting hardware performance counters. In *2008 5th Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 59–67. IEEE, 2008.

[17] Sarani Bhattacharya and Debdeep Mukhopadhyay. Who watches the watchmen?: Utilizing performance monitors for compromising keys of rsa on intel platforms. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 248–266. Springer, 2015.

[18] Manaar Alam, Sarani Bhattacharya, and Debdeep Mukhopadhyay. Tackling the time-defence: An instruction count based micro-architectural side-channel attack on block ciphers. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 30–52. Springer, 2017.