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

Manaar Alam and Debdeep Mukhopadhyay
Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur
alam.manaar@iitkgp.ac.in, debdeep@cse.iitkgp.ac.in

1. Introduction

- Deep Neural Networks (DNN) is recently being used for many privacy-preserving applications where privacy of user data requires utmost attention.
- Recent attempts try to reverse engineer a DNN model to retrieve the model parameters [1, 2] or determine user inputs [3] by exploiting side-channel information leakages to compromise privacy.
- We provide an evaluation strategy to measure private information leakages during the prediction operation of a DNN using Hardware Performance Counters (HPCs), present in most of the modern processors, and basic hypothesis testing methodology.

2. Motivation

- Execution of DNN classifier consists of a series of multiplication and addition operations on the computing environment.
- Execution of any process on CPU leaks valuable side-channel information through processor cache, branch predictor unit and other low-level hardware activities [4].
- The motivation is to explore the possibility of private information leakages in terms of these hardware events during classification operation of a DNN.

3. Information Leakage from CNN Operations

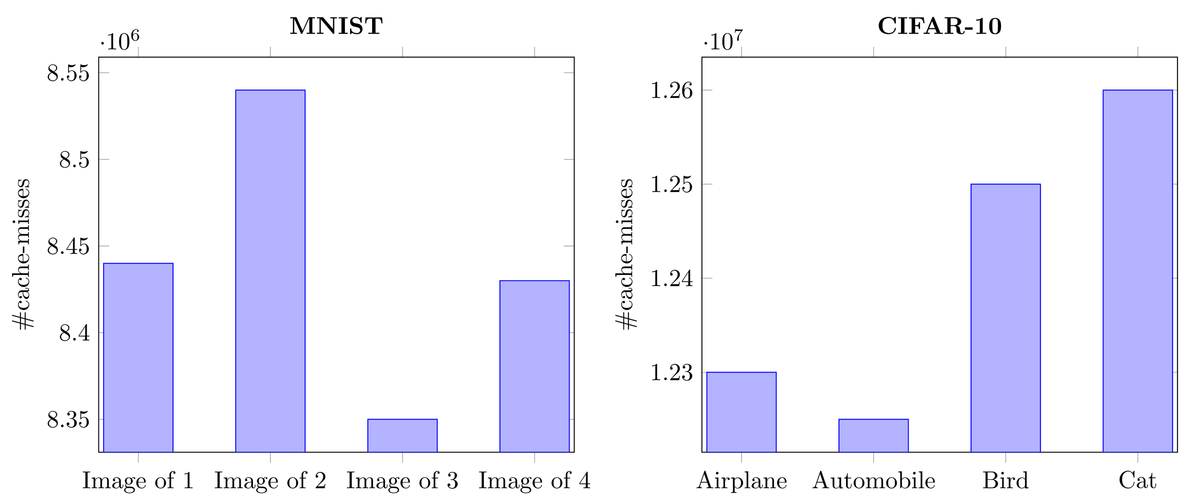

Figure 1: Information Leakages for MNIST and CIFAR-10 dataset considering different categories

- Images belonging to a particular class activates a specific 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 CNN operation affecting CPU cache, branch predictor and other units differently for different categories.

4. Methodology for Evaluation

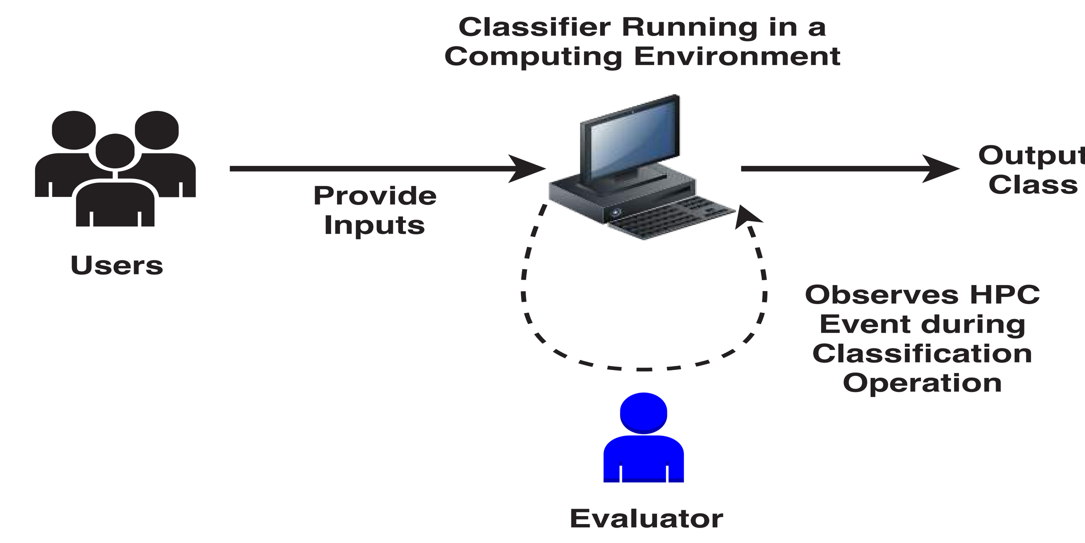

Figure 2: Evaluation Scenario

1. A group of *User* can access a CNN, trained on private information, to get predictions on their respective inputs.
2. The *Evaluator* is not provided with any details of the CNN but it can dynamically monitor HPCs during its execution using its process id and `perf` tool.
3. Various HPC events can be monitored in parallel during the classification operation of different category of input images, considering each category individually.
   - Generates distributions of different events for each class of inputs.

2,26,77,01,129	**branches**
6,24,60,873	**branch-misses**
61,95,45,765	**bus-cycles**
83,64,694	**cache-misses**
6,34,15,934	**cache-references**
16,22,12,80,350	**cycles**
12,09,42,22,814	**instructions**
15,99,20,10,924	**ref-cycles**

Figure 3: Values of different HPC events during classification of a sample MNIST image

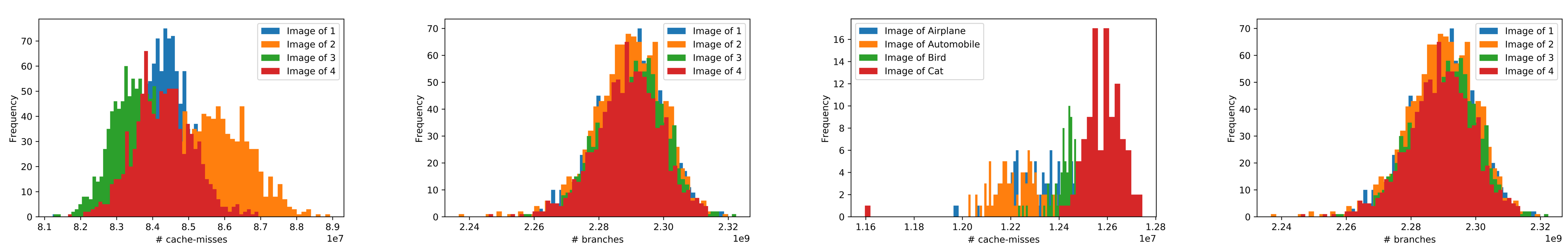

Figure 4: Distributions of differenct HPC events during the classification operation for different categories of images in MNIST and CIFAR-10

4. The *Evaluator* employs hypothesis testing methodology by computing t-statistics on the distributions of same HPC events for different categories.
   - Distinguishable distributions signify there are side-channel information leakage, which an adversary will be able to exploit to uncover private input images.
     - **Indicates an inefficient implementation of the CNN model.**

5. Results

- **Experimental Setup:**
  - Two CNNs are designed for MNIST and CIFAR-10 dataset using `tensorflow` library.
  - The CNNs are executed in *Intel Xeon E5-2690 CPU* having Ubuntu 18.04 with a 4.15.0-36-generic kernel.
- **Case Study on MNIST**

	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**

- **Case Study on CIFAR-10**

	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

* $t_{i,j}$: The t-test on distributions for category i and j.

* The **bold** faced results indicate that the two categories are distinguishable.

6. Conclusions

- We presented a strategy to evaluate the data privacy of DNN architectures with readily available Hardware Performance Counters using t-test.
- Our evaluation tool highlights the need for designing DNN architectures with indistinguishable CPU footprints while classifying different input categories in order to implement a privacy preserving classifier.

7. References

[1] Weizhe Hua, Zhiru Zhang, and G Edward Suh. Reverse engineering convolutional neural networks through side-channel information leaks. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 2018.

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

[3] Lingxiao Wei, Bo Luo, Yu Li, Yannan Liu, and Qiang 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*. ACM, 2018.

[4] 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.