A Comprehensive Study of Deep Learning for Side-Channel Analysis

Loïc Masure[1,2], Cécile Dumas[1] and Emmanuel Prouff[2,3]

[1] Univ. Grenoble Alpes, CEA, LETI, DSYS, CESTI, F-38000 Grenoble name.surname@cea.fr
[2] Sorbonne Universités, UPMC Univ Paris 06, POLSYS, UMR 7606, LIP6, F-75005, Paris, France
[3] ANSSI, France emmanuel.prouff@ssi.gouv.fr

Abstract. Recently, several studies have been published on the application of deep learning to enhance Side-Channel Attacks (SCA). These seminal works have practically validated the soundness of the approach, especially against implementations protected by masking or by jittering. Concurrently, important open issues have emerged. Among them, the relevance of machine (and thereby deep) learning based SCA has been questioned in several papers based on the lack of relation between the *accuracy*, a typical performance metric used in machine learning, and common SCA metrics like the *Guessing entropy* or the *key-discrimination success rate*. Also, the impact of the classical side-channel counter-measures on the efficiency of deep learning has been questioned, in particular by the semi-conductor industry. Both questions enlighten the importance of studying the theoretical soundness of deep learning in the context of side-channel and of developing means to quantify its efficiency, especially with respect to the optimality bounds published so far in the literature for side-channel leakage exploitation. The first main contribution of this paper directly concerns the latter point. It is indeed proved that minimizing the *Negative Log Likelihood* (NLL for short) loss function during the training of deep neural networks is actually asymptotically equivalent to maximizing the *Perceived Information* introduced by Renauld *et al.* at EUROCRYPT 2011 as a lower bound of the *Mutual Information* between the leakage and the target secret. Hence, such a training can be considered as an efficient and effective estimation of the PI, and thereby of the MI (known to be complex to accurately estimate in the context of secure implementations). As a second direct consequence of our main contribution, it is argued that, in a side-channel exploitation context, choosing the NLL loss function to drive the training is sound from an information theory point of view. As a third contribution, classical counter-measures like Boolean masking or execution flow shuffling, initially dedicated to classical SCA, are proved to stay sound against deep Learning based attacks.

Keywords: Side-Channel Analysis · Profiling Attacks · machine learning · deep learning

1 Introduction

1.1 Context

Side-channel analysis is a class of attacks against cryptographic primitives that exploit weaknesses of their physical implementation. During the execution of the latter implementation, some *sensitive variables* are indeed processed that depend on both a piece of public data (*e.g.* a plaintext) and on some chunk of a secret value (*e.g.* a key). Hence, combining information about a sensitive variable with the knowledge of the public data enables an attacker to reduce the key chunk search space. By repeating this attack several times, implementations of secure cryptographic algorithms such as the *Advanced Encryption*

Standard (AES) can then be attacked by recovering each byte of the secret key separately thanks to a *divide-and-conquer* strategy, thereby breaking the high complexity usually required to defeat such an algorithm. The information on sensitive variables is usually gathered thanks to physical leakages such as the power consumption or the electromagnetic emanations measured on the target device.

In an almost optimal attack scenario, the adversary runs a so-called *profiling phase* to learn about the statistical dependency between the manipulated sensitive variables and the leakage. An *attack phase* is subsequently launched during which the learned information is used to distinguish the secret. The first example of such a *modus operandi*, called *profiling attack*, has been published in the early 2000's under the name of *Gaussian Template Attacks* (GTA for short) [CRR02]. To circumvent it, many counter-measures were developed including *de-synchronization* and *masking*. Both of them have been shown to be practically effective [SVO+10, VMKS12], and their use in industrial implementations is today common.

The SCA Optimization Problem. When it comes to assess the security of an implementation against the profiling threat, an *evaluator* must have a clue on the optimal attack, namely the learning process and the distinguisher specification that together lead to a secret recovery with a minimal number of queries to the target device [BGH+17]. This problem is called *SCA Optimization*.

Optimal Analytical Solution. Denoting by Z the target sensitive variable and by $\mathbf{X}$ the corresponding leakage, an optimal analytical solution to the latter problem is to perfectly learn the *probability mass function (pmf)* $\Pr[Z|\mathbf{X}]$ and then, to involve it in a *Maximum Likelihood* distinguisher. Unfortunately, this pmf is unknown to the evaluator, which makes it necessary to find alternatives to this theoretical (optimal) solution.

Sub-Optimal solutions. To find sub-optimal solutions to the SCA Optimization Problem, a natural approach is to look at accurate and efficient estimators of the pmf $\Pr[Z|\mathbf{X}]$. The sequence of works [RSV+11, BHM+19, dCGRP19] enables to argue that the so-called *PMF Estimation* Problem is indeed equivalent to the SCA Optimization Problem (see Section 4). Namely, they share the same analytical optimal solution and they involve *compatible* metrics, in the sense that improving a sub-optimal solution for one problem directly leads to get an improved sub-optimal solution for the other one. Solving the new problem is typically the purpose of GTAs which approximate $\Pr[\mathbf{X}|Z]$ by a Gaussian distribution and apply Bayes' Theorem to deduce an estimator of the targeted pmf. Unfortunately, when the underlying Gaussian assumption does not hold, estimating $\Pr[\mathbf{X}|Z]$ is known to be hard in practice (especially in presence of counter-measures). This has led the SCA community to look for alternatives. Recently, the resurgence of *Deep Neural Networks* (DNNs) [KSH12] has been exploited to develop such an alternative, *e.g.* in [MZ13, LBM14, GHO15, LBM15, MDM16].

Deep Learning Based Solutions. Originally, the idea behind the training of DNNs in the SCA context was to replace the PMF Estimation Problem by another one, called *Supervised Classification*, that has the same analytical optimal solution (namely the true pmf $\Pr[Z|\mathbf{X}]$). Implicitly, one hoped that the two problems were equivalent. As far as we know, this equivalence has however never been proved and it is actually questionable since the *accuracy*, which is one of the classical metrics to assess the training quality in Supervised Classification, is clearly not equivalent to the pmf estimation metrics (*e.g.* the Euclidean distance, or the Kullback-Leibler divergence). Moreover, in [CDP17] Cagli *et al.* have pointed out that this accuracy is also not equivalent to regular Side-Channel metrics such as *Guessing Entropy* (GE) or *Success Rate* (SR) [SMY09], which has been later empirically confirmed by Picek *et al.* in [PHJ$^+$19].

1.2 Problem Addressed in this Paper

In view of the current state of the art, we are today in an uncomfortable situation where the replacement of the target SCA Optimization Problem by the Supervised Classification Problem shows promising efficiency gains while several recent papers question the theoretical soundness of the replacement [CDP17, PHJ$^+$19]. This situation prevents the SCA community to get a clear picture of the potential impact of Deep Learning, especially from the developers perspective. Indeed, though an attacker only needs to know an efficient practical approach to train a DNN, a developer needs a theoretically grounded approach to be able to give the best security bounds on the complexity of mounting a profiling attack, especially when the implementation is protected by counter-measures.

Our paper aims at grounding the use of DNNs in the SCA context, especially when classical counter-measures like masking, de-synchronization and shuffling are involved. It starts from the important observation that questioning the accuracy metric relevance is actually ill-posed in the specific case of Deep Learning based SCA since the accuracy is never directly optimized in Machine Learning. Indeed, the latter optimization is not feasible in practice[1] and DNNs are typically trained by minimizing a *surrogate* loss function (with the hope that the accuracy will be maximized as a side effect). This observation leads us to investigate to what extent the SCA Optimization Problem and the Supervised Classification Problem with respect to the surrogate loss function that is minimized (instead of the accuracy) are related. This is the main problem addressed in this paper.

1.3 Our contribution

In the literature, mostly two surrogate loss functions have been used in the SCA context: the *Negative Log Likelihood* (NLL) [CDP17, PSB$^+$18, KPH$^+$19] and the *Mean Square Error*[2] [MPP16, Tim19, WMM19]. As a first contribution, we propose a theoretical study of the NLL loss, by enlightening the fact that such a function is strongly linked to a side-channel information theoretic quantity called *Perceived Information* (PI) that has been formally introduced by Renauld *et al.* at EUROCRYPT 2011 [RSV$^+$11], and recently studied by Bronchain *et al.* [BHM$^+$19]. As a direct consequence, the PI can be straightforwardly computed from the NLL loss. But more interestingly, this implies that the training phase of a Deep Learning model, through the minimization of the NLL loss, is actually equivalent to giving the PI estimation that is the closest to the *Mutual Information* (MI) between the leakage and the target sensitive variable. This result, combined with the recent works in [BHM$^+$19] and [dCGRP19], proves that a lower bound of the minimal number of queries needed in a successful attack, which depends on the MI, can be accurately estimated, which justifies the *soundness* of addressing Supervised Classification when the latter is solved by training DNNs through the minimization of the NLL loss.

As we shall show in this paper, the latter result has many direct impacts. First, the training of DNNs with the NLL loss can be considered as an efficient and effective estimation of the PI, and thereby of the MI (known to be complex to accurately estimate in the context of secure implementations [PR10, BGP$^+$11].). Secondly, it implies that in

[1] Directly maximizing the accuracy of DNNs is a NP-hard problem. This also holds for *Random Forest* and *Support Vector Machines* that are investigated by Picek *et al.* in [PHJ$^+$19] to discuss the link between the accuracy and SCA metrics. See details in Subsection 3.2.

[2] From a purely optimization point of view, the Mean Square Error might suffer from problems [Nie18]. From a SCA evaluation point of view, the relevance of MSE is an open question [vdVP19], beyond the scope of this paper.

a SCA context, choosing the NLL loss function to drive the training is sound when it comes to address the SCA Optimization Problem. Thirdly, it enables to study the impact of classical SCA counter-measures on the efficiency of Deep Learning based SCA and to formally prove that they stay sound.

The second part of the paper is dedicated to the validation of our theoretical results through several experiments and simulations in the context of implementations secured by masking, shuffling and de-synchronization.

1.4 Organization of the paper

The paper is organized as follows. Notations are first introduced in Section 2. In Section 3, the profiling attack scenario is introduced. It also discusses the relevance of the Supervised Classification Problem in a Side-Channel context, and propose another way to tackle the evaluation. Section 4 states the soundness of minimizing the NLL loss since it is nothing but maximizing the Perceived Information. This will then be verified by simulations in Section 5, and also illustrated on experimental examples in Section 6.

2 Notations

Throughout the paper we use calligraphic letters as $\mathcal{X}$ to denote sets, the corresponding upper-case letter X to denote random variables (resp. random vectors $\mathbf{X}$) over $\mathcal{X}$, and the corresponding lower-case letter x (resp. $\mathbf{x}$) to denote realizations of X (resp. $\mathbf{X}$). The i-th entry of a vector $\mathbf{x}$ is denoted by $\mathbf{x}[i]$. We denote the probability space of a set $\mathcal{X}$ by $\mathcal{P}(\mathcal{X})$. If $\mathcal{X}$ is discrete, it corresponds to the set of vectors $[0,1]^{|\mathcal{X}|}$ such that the coordinates sum to 1. If a random variable X is drawn from a distribution $\mathcal{D}$, then $\mathcal{D}^N$ denotes the joint distribution over the sequence of N i.i.d. random variables of same probability distribution than X. The symbol $\mathbb{E}$ denotes the expected value, and might be subscripted by a random variable $\mathbb{E}_X$, or by a probability distribution $\underset{X\sim\mathcal{D}}{\mathbb{E}}$ to specify under which probability distribution it is computed. Likewise, $\mathbb{V}$ denotes the variance of a random variable. The output of a cryptographic primitive $\mathbf{C}$ is considered as the target sensitive variable $Z = \mathbf{C}(P, K)$, where P denotes some public variable, *e.g.* a plaintext chunk, where K denotes the part of secret key the attacker aims to retrieve, and where Z takes values in $\mathcal{Z} = \{s_1, \dots, s_{|\mathcal{Z}|}\}$. Among all the possible values K may take, $k^\star$ will denote the right key hypothesis. Side-channel traces will be viewed as discrete realizations of a random column vector $\mathbf{X}$ with values in $\mathcal{X} = [0, 2^\omega - 1]^D$ where ω depends on the vertical resolution of the oscilloscope used for the acquisitions (usually, we have $\omega \in \{8, 10, 12\}$.

Let $(A_n)_n$ be a sequence of random variables and let A be another random variable. We say that A_n converges in probabilities towards A, denoted as $A_n \underset{n\to\infty}{\overset{\mathcal{P}}{\longrightarrow}} A$ when the following property holds:

$$\forall\epsilon > 0, \Pr\left[|A_n - A| \geq \epsilon\right] \underset{n\to\infty}{\longrightarrow} 0.$$

We now define some Information Theoretic quantities that have been taken from [CT06]. Let $Z \in \mathcal{Z}$ be a discrete random variable. The *entropy* of Z, denoted by $\mathsf{H}(Z)$, describes the uncertainty to guess the value of a realization of a discrete random variable Z. It is formally defined by:

$$\mathsf{H}(Z) \triangleq -\sum_{s\in\mathcal{Z}} \Pr[Z = s] \log_2 \Pr[Z = s].$$

Likewise, the *conditional entropy* of a discrete random variable Z given another random variable $\mathbf{X}$ quantifies the remaining uncertainty on the guess of Z once $\mathbf{X}$ is known. It is formally defined as:

$$\mathsf{H}(Z|\mathbf{X}) \triangleq \mathop{\mathbb{E}}_{\mathbf{X}}\left[-\sum_{s\in\mathcal{Z}} \Pr[Z = s|\mathbf{X}] \log_2 \Pr[Z = s|\mathbf{X}]\right].$$

If $\mathcal{D}$ and $\mathcal{D}'$ are two probability distributions on $\mathcal{Z}$, we define the *Kullback - Leibler divergence* (or KL divergence) as:

$$\mathsf{D}\left(\mathcal{D} \,||\, \mathcal{D}'\right) \triangleq \sum_{s\in\mathcal{Z}} \mathcal{D}(s) \log_2 \frac{\mathcal{D}(s)}{\mathcal{D}'(s)}.$$

This quantity is typically used to measure the difference between two discrete probability distributions, since it is always non-negative and equals zero if and only if $\mathcal{D} = \mathcal{D}'$. Thanks to the previous definitions, we can introduce the ***M**utual **I**nformation* (MI) between two variables Z and $\mathbf{X}$ as:

$$\mathsf{MI}\left(Z; \mathbf{X}\right) \triangleq \mathsf{H}(Z) - \mathsf{H}(Z|\mathbf{X}) = \mathsf{D}\left(\Pr[\mathbf{X}, Z] \,||\, \Pr[\mathbf{X}]\Pr[Z]\right).$$

This characterizes how much information can be obtained about Z by observing $\mathbf{X}$.

3 SCA Optimization Problem *versus* Deep Learning Based SCA

3.1 Profiling Attacks and their Evaluation

This section presents the framework we will consider when attacking a device through a profiling attack. Once presented, we will set the goal of an evaluator.

The considered scenario is made of the following steps:

- *Profiling acquisition*: a dataset of N_p *profiling traces* is acquired on the prototype device. It will be seen as a realization of the random variable $S_p \triangleq \{(\mathbf{x}_1, z_1), \ldots, (\mathbf{x}_{N_p}, z_{N_p})\} \sim \Pr[\mathbf{X}, Z]^{N_p}$, where all the $\mathbf{x}_i$ (resp. all the z_i) are i.i.d. realizations of $\mathbf{X}$ (resp. Z).

- *Profiling phase*: based on S_p, a model is built that returns a set of scores for each hypothetical value of Z, that can be assimilated to a pmf (possibly after normalization). $F : \mathcal{X} \to \mathcal{P}(\mathcal{Z})$.

- *Attack acquisition*: a dataset of N_a *attack traces* is acquired on the target device. It will be seen as a realization of $S_a \triangleq (k^\star, \{(\mathbf{x}_1, p_1), \ldots, (\mathbf{x}_{N_a}, p_{N_a})\})$ such that $k^\star \in \mathcal{K}$, and for all $i \in [\![1, N_a]\!]$, $p_i \sim \Pr[P]$ and $\mathbf{x}_i \sim \Pr[\mathbf{X}|Z = \mathbf{C}(p_i, k^\star)]$.

- *Predictions*: a prediction vector is computed on each attack trace, based on the previously built model: $\mathbf{y}_i = F(\mathbf{x}_i), i \in [\![1, N_a]\!]$. For each trace, it assigns a score to each key hypothesis, namely, for every $j \in [\![1, |\mathcal{Z}|]\!]$, the value of the j-th coordinate of $\mathbf{y}_i$ corresponds to the score assigned by the model to the hypothesis $Z = s_j$ when observing $\mathbf{x}_i$.

- *Guessing*: the scores are combined over all the attack traces to output a *likelihood* for each key hypothesis; the candidate with the highest likelihood is predicted to be the right key. A maximum likelihood score can be used for the guessing. For every key hypothesis $k \in \mathcal{K}$, this score is defined as:

$$\mathbf{d}_{S_a}[k] \triangleq \sum_{i=1}^{N_a} \log\left(\mathbf{y}_i[z_i]\right) \text{ where } z_i = \mathbf{C}(p_i, k). \tag{1}$$

Based on the scores in Equation 1, the key hypotheses are ranked in a decreasing order. Finally, the attacker chooses the key that is ranked first. More generally, the *rank* $g_{S_a}(k^\star)$ of the correct key hypothesis $k^\star$ is defined as:

$$g_{S_a}(k^\star) \triangleq \sum_{k \in \mathcal{K}} 1_{\mathbf{d}_{S_a}[k] > \mathbf{d}_{S_a}[k^\star]} . \tag{2}$$

If $g_{S_a}(k^\star) = 1$, then the attack is considered as *successful*.

To assess the difficulty of attacking a target device with profiling attacks (which is assumed to be the worst-case scenario for the attacked device), it has initially been suggested to measure or estimate the minimum number of traces required to get a successful attack [Man04]. Observing that many random factors may be involved during the attack, the latter measure has been refined to study the probability that the right key is ranked first. This metric is called the ***S**uccess **R**ate* [SMY09]:[3]

$$\mathrm{SR}(N_a) \triangleq \Pr\left[g_{S_a}(k^\star) = 1\right] . \tag{3}$$

Within this framework, it is common to formulate the evaluator's goal in the worst-case scenario as follows [LPB$^+$15, HGM$^+$11, PHJ$^+$19]:

Problem 1 (SCA Optimization)**.** *Given a profiling set S_p, find a model F that minimizes N_a such that $\mathrm{SR}(N_a) \geq \beta$, where β is a threshold defined by the evaluator. We denote $N_a^\star$ the corresponding minimum number of attack traces.*

For convenience, we will denote by $N_a(F)$ the minimal number of attack traces needed to verify the condition $\mathrm{SR}(N_a) \geq \beta$ for a given fixed model F. An analytical optimal solution to Problem 1 is given by the conditional pmf of the leakage, as stated in the following proposition.

Proposition 1 (Optimal solution for Problem 1[HRG14])**.** *The model $F^\star$ defined by $\forall \boldsymbol{x} \in \mathcal{X}, \forall s \in \mathcal{Z}, F^\star(\boldsymbol{x}) = \Pr[Z = s | \boldsymbol{X} = \boldsymbol{x}]$ (or $F^\star = \Pr[Z|\boldsymbol{X}]$) is an optimal solution to Problem 1.*

Proposition 1 tells us that the conditional pmf is the best model we can build so far for a profiling attack. Yet, such a solution is still analytical and remains unknown to the evaluator, which makes it necessary to find alternatives to this theoretical (optimal) solution.

To find sub-optimal solutions to the SCA Optimization Problem, a natural approach is to look at accurate and efficient estimators of the pmf $\Pr[Z|\mathbf{X}]$. Solving the new problem is typically the purpose of GTAs which approximate $\Pr[\mathbf{X}|Z]$ by a Gaussian distribution and apply Bayes' Theorem to deduce an estimator of the targeted pmf. Unfortunately, when the underlying Gaussian assumption does not hold, estimating $\Pr[\mathbf{X}|Z]$ is known to be hard in practice (especially in presence of counter-measures) [BGH$^+$17, BGHR14]. This has led the SCA community to look for alternatives. The Machine (and especially deep) learning paradigm aims at generalizing the approach taken by GTAs, by considering wider sets of models to approach the optimal solution. This alternative will be discussed in the next section.

3.2 Problems with Supervised Classification

This section presents the concept of Supervised Classification which is commonly applied to solve the SCA Optimization Problem with machine learning. Its soundness is discussed in the context of SCA. Formally, Supervised Classification is defined as the problem of

[3] One can equivalently study average ranking of the correct guess, a.k.a. the ***G**uessing **E**ntropy*.

finding one estimator of the true pmf that maximizes the *accuracy*, namely the rate of good predictions of the value of the target sensitive variable Z over the joint distribution of the leakage $Z, \mathbf{X}$. For any function $F : \mathcal{X} \longrightarrow \mathcal{P}(\mathcal{Z})$, this accuracy is denoted by $\mathit{Acc}(F)$ and is defined as:

$$\mathit{Acc}(F) \triangleq \Pr\left[\operatorname*{argmax}_{s \in \mathcal{Z}} F(\mathbf{X})[s] = Z\right] \ .$$

The estimator is taken from a *parameterized hypotheses class* previously defined by the evaluator. The class may be seen as a collection of models of the form

$$F : \left| \begin{array}{ccc} \mathcal{X} & \longrightarrow & \mathcal{P}(\mathcal{Z}) \\ x & \longmapsto & F(x, \theta) \end{array} \right. ,$$

where $\theta \in \Theta \subseteq \mathbb{R}^q$ denotes the q-dimensional vector gathering all the parameters.[4]

It turns out that both SCA Optimization and Supervised Classification Problems are linked, thanks to the following proposition that essentially states that no estimator can have a better accuracy than the one defined by the true conditional pmf:

Proposition 2 (Bayes Error for Supervised Classification [SSBD14])**.** *Let $F^\star$ be $F^\star = \Pr[Z|\boldsymbol{X}]$. Then, for any function $F : \mathcal{X} \rightarrow \mathcal{P}(\mathcal{Z})$ we have:*

$$\mathit{Acc}(F) \leq \mathit{Acc}(F^\star) \ . \tag{4}$$

When $F^\star$ belongs to the considered hypotheses class $\mathcal{H}$, Propositions 1 and 2 tell us that the model selection through accuracy maximization is the optimal strategy for the SCA Optimization Problem. When the latter condition is not satisfied, the accuracy maximization strategy outputs another model, which has sub-optimal accuracy compared to $F^\star$ as stated in Proposition 2. This questions the soundness of the accuracy maximization for the SCA Optimization Problem, or in other words, whether the accuracy properly quantifies the quality of a solution for Problem 1. This question has first been pointed out by Cagli *et al.* who mentioned in [CDP17] that the accuracy only corresponds to finding the model that maximizes SR(1), which is different from the criterion we consider in the SCA Optimization Problem. Likewise, Picek *et al.* empirically verified in [PHJ+19] that the accuracy of some Machine Learning models such as *Support Vector Machines* (SVM) and *Random Forests* (RF) was not always related to the Guessing Entropy. More precisely, they argue that a high accuracy is a clue for good performance in SCA Optimization, though the inverse does not empirically hold. All together, these findings question the Supervised Classification Approach in a SCA context.

This paper aims at addressing a slightly rephrased version of the question raised in [CDP17] and [PHJ+19]. Indeed, machine learning algorithms do not directly maximize the accuracy in practice,[5], while directly maximizing the accuracy is known to be computationally intractable. This also holds for the specific case of DNNs [SSBD14, Thm 20.7] where a *surrogate* loss function is minimized instead with the hope that this will also optimize the target loss function (*e.g.* to maximize the Supervised Classification accuracy). A commonly used surrogate loss is the *Negative Log Likelihood* (NLL) [CDP17, PSB+18, KPH+19]. We hereafter recall its definition:

[4] Thus, Θ is the parameters space and equivalently defines $\mathcal{H}$. In the following, both notations $\mathcal{H}$ and Θ will be used interchangeably according to the context. Likewise, a pmf from such a parameterized hypothesis class might be denoted either by $F(., \theta)$ or simply by its parameter vector θ.

[5] This is especially the case for RFs which are based on heuristics working reasonably well in practice [SSBD14, Chap. 18.2] or SVMs who minimize another loss function called *Hinge loss* [SSBD14, Chap. 15.2.3].

Definition 1 (Negative Log Likelihood)**.** Given $S_p = \{(\mathbf{X}_1, Z_1), \ldots, (\mathbf{X}_{N_p}, Z_{N_p})\} \sim \Pr[\mathbf{X}, Z]^{N_p}$, and a DNN model defined by θ from a hypothesis class $\mathcal{H}$, the Negative Log Likelihood is defined as:

$$\mathcal{L}_{S_p}(\theta) \triangleq \frac{1}{N_p} \sum_{i=1}^{N_p} -\log_2 F(\mathbf{X}_i, \theta)[Z_i]. \tag{5}$$

Furthermore, we define the *Maximum Likelihood Estimator*[6] $\hat{\theta}$ as the parameter vector from Θ that minimizes the NLL loss computed over the profiling set S_p: $\hat{\theta} \triangleq \mathrm{argmin}_{\theta \in \Theta} \mathcal{L}_{S_p}(\theta)$.

Since DNNs are trained by minimizing the NLL instead of the accuracy, we argue here that in the specific case where one considers neural networks as a hypotheses class, the question raised by [CDP17] and [PHJ+19] should be rephrased as questioning the equivalence between the NLL minimization for DNNs and the SCA Optimization Problem (*i.e.* Problem 1). As a side effect, this questioning does not extend to algorithms not minimizing this loss, such as Random Forest or (kernel based) SVMs. In order to address this question, we need first to substitute, in the next section, the SCA Optimization problem with an intermediate problem.

3.3 Model Training for Leakage Assessment

The problem substitution presented in this section comes as a direct consequence of the recent work [dCGRP19] which has stated that $N_a^\star$, namely the number of traces required to succeed an attack when the involved model corresponds to the optimal solution to Problem 1, is linked to the MI between the target sensitive variable and the leakage through the following inequality:

$$\frac{f(\beta)}{\mathsf{MI}(Z; \mathbf{X})} \leq N_a^\star, \tag{6}$$

where f is a known, invertible, strictly increasing function defined in [dCGRP19], and β is the threshold defined in Problem 1. Cherisey *et al.* argue that the lower $N_a^\star$, the tighter Inequality (6). Nevertheless, from the point of view of conservative security evaluations, it remains interesting to compute the value of the left-hand side in Inequality (6), no matter the value of $N_a^\star$.

Unfortunately, computing the MI in the denominator also requires to perfectly know the true pmf $\Pr[Z|\mathbf{X}]$. Like with the SCA Optimization and the Supervised Classification Problem, this cannot be assumed in practice. To circumvent this issue, we can fortunately use the notion of ***P**erceived **I**nformation* (PI) which extends the MI to accept pmfs estimations [RSV+11].

Definition 2 (Perceived Information [BHM+19])**.** Let Θ be the parameter space of a parameterized hypothesis class $\mathcal{H}$ and let θ be an element in Θ. The *Perceived Information* between Z and $\mathbf{X}$ for the model $F(., \theta) \in \mathcal{H}$ is denoted by $\mathsf{PI}(Z; \mathbf{X}; \theta)$ and defined as:

$$\mathsf{PI}(Z; \mathbf{X}; \theta) \triangleq \mathsf{H}(Z) + \sum_{s \in \mathcal{Z}} \Pr[Z = s] \mathop{\mathbb{E}}_{\mathbf{X}|Z=s} [\log_2 F(\mathbf{X}, \theta)[s]] \ . \tag{7}$$

Intuitively, when the pmf $\Pr[Z|\mathbf{X}]$ is perfectly learned, the PI equals the MI, otherwise the first one is always lower than the latter one [BHM+19]. This is of great interest here since it enables to derive an upper bound of the left-hand side in Inequality (6), namely

$$\frac{f(\beta)}{\mathsf{MI}(Z; \mathbf{X})} \leq \frac{f(\beta)}{\mathsf{PI}(Z; \mathbf{X}; \theta)} \triangleq \widetilde{N}_{a,\theta}. \tag{8}$$

[6] Minimizing the NLL loss is equivalent to maximize the Log Likelihood.

Moreover, we can then compare different models in terms of their PI: the higher the PI, the lower the distance to MI and thereby the better the estimation of $\frac{f(\beta)}{\mathsf{MI}(Z;\mathbf{X})}$ with $\widetilde{N_{a,\theta}}$. This leads to introduce a new intermediate Problem, named Leakage Assessment.

Problem 2 (Leakage Assessment)**.** *Given a profiling set $S_p \sim \Pr[\boldsymbol{X}, Z]^{N_p}$, find the model with the highest PI.*

At this point, we have argued that addressing the Leakage Assessment Problem is *sound* for the SCA Optimization Problem, in the sense that it will enable to estimate a lower-bound of the optimal solution $N_a^\star$ of the latter problem. The following section aims at deeply studying Problem 2. We will show that training deep learning models with the NLL loss is asymptotically equivalent to this problem which implies that conducting profiled SCA with deep learning can be argued to be relevant within this framework.

4 NLL Minimization is PI Maximization

This section is devoted to show that a deep learning model trained by minimizing the NLL loss fits with Problem 2. Subsection 4.1 studies the link between the NLL loss and an information theoretical quantity called *Cross Entropy*, that we will define hereafter. Then, Subsection 4.2 will make a link between cross entropy and PI. Finally, Subsection 4.3 discusses the gap between the MI and a PI estimated by training deep learning based models. Eventually, it will be concluded that the MI can be accurately estimated thanks to this approach.

4.1 The Consistency of the NLL Loss with Cross Entropy

This subsection is devoted to recall to the unfamiliar reader an important machine learning notion that will be used afterwards in Subsection 4.2, namely the property of *consistency*. Briefly, it states that the NLL loss minimization is asymptotically equivalent to the minimization of an information theoretic quantity called *Cross-Entropy*. We stand by recalling the latter notion hereafter.

Definition 3 (Cross Entropy)**.** Given a joint probability distribution of a target sensitive variable Z and its leakage $\mathbf{X}$ denoted as $\Pr[\mathbf{X}, Z]$, we define the *Cross Entropy* as the expected value of each term in Equation 5:

$$\mathcal{L}_{\Pr(\mathbf{X},Z)}(\theta) \triangleq \mathop{\mathbb{E}}_{\mathbf{X},Z}\left[-\log_2 F(\mathbf{X},\theta)[Z]\right] \quad . \tag{9}$$

The cross entropy is actually nothing but the expected value of the NLL loss computed over the profiling set of traces. Besides, according to the law of large numbers, for any fixed θ the NLL loss converges in probabilities towards the cross entropy [SSBD14]. However, since the true joint distribution of Z and $\mathbf{X}$ is actually unknown, one cannot exactly compute the cross entropy. The hope behind the NLL minimization is that for a number N_p of profiling traces high enough, the obtained parameter vector $\hat{\theta}$ will be a good candidate to minimize the cross entropy.

It is not trivial though that $\mathcal{L}_{S_p}(\hat{\theta})$ converges in probabilities towards $\min_{\theta\in\Theta} \mathcal{L}_{\Pr(\mathbf{X},Z)}(\theta)$, as $\hat{\theta}$ is varying for each value of N_p. Thankfully, a fundamental result of machine learning called *consistency* proves the soundness of the approach.[7]

[7] Actually, the Cramer-Rao bound (a well known result in Statistics) [Cra99] guarantees the latter convergence, but relies on assumptions that cannot be taken for granted, in particular the assumption that there exists $\theta \in \Theta$ such that $F(.,\theta) = \Pr[Z|\mathbf{X}]$.

Theorem 1 (Consistency of Maximum Likelihood Estimation [Vap99, SSBD14])**.** *Let $N_p \in \mathbb{N}$ and let S_p be a profiling set of size N_p. Assume that $\mathcal{H}$ is a hypotheses class of finite VC-dimension*[8] *(or equivalently Θ is its parameter space). Then:*

$$\sup_{\theta \in \Theta} \left\{ \mathcal{L}_{\Pr(\boldsymbol{X},Z)}(\theta) - \mathcal{L}_{S_p}(\theta) \right\} \xrightarrow[N_p \to \infty]{\mathcal{P}} 0 \tag{10}$$

In particular, if $\mathcal{H}$ is the set of Multi-Layer Perceptron (MLP), it follows that:

$$\mathcal{L}_{S_p}(\hat{\theta}) \xrightarrow[N_p \to \infty]{\mathcal{P}} \min_{\theta \in \Theta} \mathcal{L}_{\Pr(\boldsymbol{X},Z)}(\theta), \tag{11}$$

$$\mathcal{L}_{\Pr(\boldsymbol{X},Z)}(\hat{\theta}) \xrightarrow[N_p \to \infty]{\mathcal{P}} \min_{\theta \in \Theta} \mathcal{L}_{\Pr(\boldsymbol{X},Z)}(\theta). \tag{12}$$

In other words, the solution $\hat{\theta}$ given by the minimization of the NLL loss converges towards the best possible solution for the cross entropy, and the NLL loss of $\hat{\theta}$ is a good approximation of the generalization loss of $\hat{\theta}$.

Proof. The Fundamental Theorem of Statistical Learning states that the consistency holds if and only if the *VC-dimension* of $\mathcal{H}$ is finite [Vap99]. In parallel, if $\mathcal{H}$ is a class hypothesis trained by minimizing a real valued loss function, its VC-dimension equals the VC-dimension of the same hypothesis class where each model has a binary output [Vap95, p76]. For the specific class of *Multi-Layer Perceptron* (MLP) with a binary output, the VC-dimension is indeed finite [SSBD14, Theorem 20.6, p274], as it can be bounded by a function of the number of neurons. □

As mentioned in Theorem 1, the latter result also holds for any hypotheses class with finite VC-dimension. This includes for example (kernel based) softmax classifiers that are beyond the scope of this paper.

As a consequence of Theorem 1, any property verified by the cross entropy is also asymptotically verified by the NLL loss (*i.e.* when the number of profiling traces N_p converges towards infinity). Therefore we can substitute the analysis of the NLL loss with that of the cross entropy. It remains now to draw the link between cross entropy and PI, in order to address the Leakage Assessment Problem.

4.2 The Link between Cross Entropy and Perceived Information

This section aims at explaining to what extent the PI and the cross entropy introduced in the previous section are linked. It is argued here that the PI actually equals the cross entropy up to constant factors. Such a link and the reduction argued in Subsection 4.1 will allow us to guarantee that minimizing the NLL loss is a consistent approach for solving the Leakage Assessment Problem. It is recalled that the PI has been formally defined in Subsection 3.3. We also introduce hereafter the *Empirical Perceived Information*, as given in [BHM+19].

Definition 4 (Empirical Perceived Information [BHM+19])**.** Let Θ be the parameter space of a parameterized hypothesis class $\mathcal{H}$. Let $\theta \in \Theta$. The *Empirical Perceived Information*, denoted as $\widehat{\mathsf{PI}_{N_p}}(Z; \mathbf{X}; \theta)$, is defined from a profiling set S_p as follows:

$$\widehat{\mathsf{PI}_{N_p}}(Z; \mathbf{X}; \theta) \triangleq \mathsf{H}(Z) + \sum_{s \in \mathcal{Z}} \Pr[Z = s] \frac{1}{N_p} \sum_{\substack{(\mathbf{x},z) \in S_p \\ z = s}} \log_2 F(\mathbf{x}, \theta)[z] \ . \tag{13}$$

[8] The Vapnik-Chervonenkis (VC) dimension of a hypotheses class is a natural number quantifying its *capacity*. Roughly speaking, the higher the VC-dimension, the slower the convergence in Equation 11 and Equation 12, but lower the limit. This trade-off is also known as the *Bias-Variance* trade-off [SSBD14].

Informally, the PI is defined the same way as the MI, but by substituting the *uncertainty* of the true pmf, namely $\log_2 \Pr[Z|\mathbf{X} = \mathbf{x}]$, with the uncertainty of the approximating pmf, namely $\log_2 F(\mathbf{X}, \theta)$. Surprisingly, this substitution is exactly what defines the cross entropy.

Proposition 3 (Our contribution)**.** *Let Z be a random variable with uniform distribution over $\mathcal{Z} = \mathbb{F}_2^n$ for some $n \in \mathbb{N}$. Then, the cross entropy and the NLL loss are respectively linked to the Perceived Information and its empirical estimation as follows:*

$$\begin{aligned} n - \mathcal{L}_{\Pr[\boldsymbol{X},Z]}(\theta) &= \mathsf{PI}\,(Z; \boldsymbol{X}; \theta)\,, && (14) \\ n - \mathcal{L}_{S_p}(\theta) &= \widehat{\mathsf{PI}_{N_p}}\,(Z; \boldsymbol{X}; \theta)\,. && (15) \end{aligned}$$

Proof. The assumption about Z implies that $\mathsf{H}(Z) = n$. Injecting the latter result into the definition of the PI, and by using the formula of total probabilities for the expected value we have:

$$\begin{aligned} \mathsf{PI}\,(Z; \mathbf{X}; \theta) &\triangleq \mathsf{H}(Z) + \sum_{s \in \mathcal{Z}} \Pr[Z = s] \mathop{\mathbb{E}}_{\mathbf{X}|Z=s} [\log_2 F(\mathbf{X})[s]]\,, \\ &= n + \mathop{\mathbb{E}}_{Z}\left[\mathop{\mathbb{E}}_{\mathbf{X}|Z} [\log_2 F(\mathbf{X}, \theta)[Z]]\right], \\ &= n - \mathop{\mathbb{E}}_{\mathbf{X},Z} [-\log_2 F(\mathbf{X}, \theta)[Z]]\,, \\ &= n - \mathcal{L}_{\Pr[\mathbf{X},Z]}(\theta). \end{aligned}$$

The proof for the empirical PI follows exactly the same reasoning substituting expected values with averages.[9] □

Proposition 3 tells us that the cross entropy and the Perceived Information are exactly the same concept. As already pointed out in [BHM+19, Thm. 6], we have for all $\theta \in \Theta$ $\mathsf{PI}\,(Z; \mathbf{X}; \theta) \leq \mathsf{MI}\,(Z; \mathbf{X})$. In other words, computing the cross entropy of any deep learning model enables to get a lower bound of the MI. This tells nothing about the tightness of such a bound though. Hopefully, based on the previous results stated in this section, we now know how to tighten this inequality, as stated by the following proposition.

Proposition 4. *Let $\hat{\theta}$ denote the parameter vector obtained by the minimization of the NLL loss, namely such that $\hat{\theta} \triangleq \operatorname{argmin}_{\theta \in \Theta} \mathcal{L}_{S_p}(\theta)$. Then: (1) $\hat{\theta}$ maximizes the empirical PI and (2) the information perceived by $\hat{\theta}$ converges in probabilities towards the maximum of Perceived Information over $\mathcal{H}$.*

$$\begin{aligned} \widehat{\mathsf{PI}_{N_p}}\left(Z; \boldsymbol{X}; \hat{\theta}\right) &\xrightarrow[N_p \to \infty]{\mathcal{P}} \max_{\theta \in \Theta} \mathsf{PI}\,(Z; \boldsymbol{X}; \theta) \leq \mathsf{MI}\,(Z; \boldsymbol{X}) && (16) \\ \mathsf{PI}\left(Z; \boldsymbol{X}; \hat{\theta}\right) &\xrightarrow[N_p \to \infty]{\mathcal{P}} \max_{\theta \in \Theta} \mathsf{PI}\,(Z; \boldsymbol{X}; \theta) \leq \mathsf{MI}\,(Z; \boldsymbol{X}) && (17) \end{aligned}$$

Roughly speaking, Proposition 4 states that the NLL loss minimization is asymptotically equivalent to the PI maximization mentioned in the Leakage Assessment Problem (*i.e.* Problem 2).

Proof. Starting from Equation 14, and applying Equation 11 from Theorem 1 to get

$$\begin{aligned} n - \mathsf{PI}\left(Z; \mathbf{X}; \hat{\theta}\right) \overset{(14)}{=} \mathcal{L}_{\Pr[\mathbf{X},Z]}(\hat{\theta}) &\overset{(11)}{\xrightarrow[N_p \to \infty]{\mathcal{P}}} \min_{\theta \in \Theta} \mathcal{L}_{\Pr(\mathbf{X},Z)}(\theta) \\ &= \min_{\theta \in \Theta} (n - \mathsf{PI}\,(Z; \mathbf{X}; \theta)) \\ &= n - \max_{\theta \in \Theta} \mathsf{PI}\,(Z; \mathbf{X}; \theta) \end{aligned}$$

[9] Actually, Equation 15 only holds if S_p is *balanced*, *i.e.* if the number of traces is the same for each class in S_p. This can be assumed without loss of generality, since Z is drawn uniformly.

Hence the result given in Equation 17. The proof for Equation 16 follows exactly the same reasoning, replacing $\mathsf{PI}\left(Z; \mathbf{X}; \hat{\theta}\right)$ by $\widehat{\mathsf{PI}_{N_p}}\left(Z; \mathbf{X}; \hat{\theta}\right)$, and applying Equation 12 instead of Equation 11. □

Therefore, on the one hand, we have a theoretically grounded method to address the Leakage Assessment Problem (*i.e.* Problem 2) thanks to Proposition 4, namely by minimizing the NLL loss. On the other hand, since it has been argued in Subsection 3.3 that solving the Leakage Assessment was sound in order to address the SCA Optimization Problem, it follows from Proposition 4 the main result of this paper, given hereafter.

Corollary 1 (Main Result)**.** *Let $\hat{\theta}$ denote the the parameter vector obtained by the minimization of the NLL loss, namely such that $\hat{\theta} \triangleq \mathrm{argmin}_{\theta \in \Theta} \mathcal{L}_{S_p}(\theta)$. Then the model given by $F(., \hat{\theta})$ asymptotically minimizes the quantity $\widetilde{N_a}_{,\theta} \triangleq \frac{f(\beta)}{\mathsf{PI}(Z;\mathbf{X};\theta)}$ which is an upper-bound of the left-hand side in Inequality 6.*

Proof. By applying Proposition Proposition 4, we get

$$\widetilde{N_a}_{,\hat{\theta}} \xrightarrow[N_p \to \infty]{\mathcal{P}} \frac{f(\beta)}{\max_{\theta \in \Theta} \mathsf{PI}\left(Z; \mathbf{X}; \hat{\theta}\right)} = \min_{\theta \in \Theta} \frac{f(\beta)}{\mathsf{PI}\left(Z; \mathbf{X}; \theta\right)} \geq \frac{f(\beta)}{\mathsf{MI}\left(Z; \mathbf{X}\right)}$$

□

In other words, Corollary Corollary 1 tells us that minimizing the NLL loss is sound for the SCA Optimization Problem (*i.e.* Problem Problem 1), in the sense that has been argued in Subsection 3.3, and that the term $\widetilde{N_a}_{,\hat{\theta}}$ might be a good approximation in view of estimating the lower bound of Inequality (6). However, this also emphasizes that in the pursuit of estimating $N_a^\star$ through the NLL minimization, some weaknesses must be discussed.

First, as recalled in Subsection 3.3, the higher $N_a^\star$, the looser Inequality (6). It is therefore of natural interest to verify to what extent the tightness of the latter inequality holds, in view of estimating $N_a^\star$ by $\frac{f(\beta)}{\mathsf{MI}(Z;\mathbf{X})}$. This must be at least empirically verified. Second, the tightness of Inequality (17) is another possible source of imprecision when one wants to substitute the MI with the PI. This will be discussed in the next section, and will eventually be verified through simulations and experiments.

4.3 To what Extent the Obtained Bound is Tight?

So far we have argued that minimizing the NLL loss is a sound approach to tackle Problem 2: it is indeed consistent with minimizing the cross entropy (*cf* Equation 9), thereby consistent with maximizing the PI (*cf* Equation 7). In the particular case where the hypothesis class $\mathcal{H}$ is a set of neural networks, it becomes now of natural interest to study the gap between the MI and the NLL loss we are minimizing (or equivalently the empirical PI we are maximizing) to assess the quality of the built solution.

Such a minimization is typically done with a *Stochastic Gradient Descent* (SGD) algorithm. The obtained model is denoted by the parameter vector θ_{SGD}. Thus, one can decompose the gap between the solution found with SGD and the MI into three parts:

$$\begin{aligned}
\widehat{\mathsf{PI}_{N_p}}\left(Z; \mathbf{X}; \theta_{SGD}\right) - \mathsf{MI}\left(Z; \mathbf{X}\right) &= \left(\widehat{\mathsf{PI}_{N_p}}\left(Z; \mathbf{X}; \theta_{SGD}\right) - \widehat{\mathsf{PI}_{N_p}}\left(Z; \mathbf{X}; \hat{\theta}\right)\right) && (18)\\
&+ \left(\widehat{\mathsf{PI}_{N_p}}\left(Z; \mathbf{X}; \hat{\theta}\right) - \sup_{\theta \in \Theta} \mathsf{PI}\left(Z; \mathbf{X}; \theta\right)\right) && (19)\\
&+ \left(\sup_{\theta \in \Theta} \mathsf{PI}\left(Z; \mathbf{X}; \theta\right) - \mathsf{MI}\left(Z; \mathbf{X}\right)\right). && (20)
\end{aligned}$$

The term (20) corresponds to the *approximation error*: this error is due to the choice of a restricted hypotheses class $\mathcal{H}$ from which we select our model. This error is of particular interest as it gives a *computational security bound*. Proposition 4 shows that no model from $\mathcal{H}$ can give a tighter lower bound on the MI. A remarkable result specific to *Multi-Layer Perceptrons*, a simple type of DNN, known as the *Universal Approximation Theorem*, states that when considering a L^2 error as a loss function, such an approximation error converges towards 0 when the number of neurons in the layers increases [Pet98].[10] Unfortunately, to the best of our knowledge, no similar result has been stated when considering the cross entropy as an approximation error.

The term (19) corresponds to the *estimation error*. It is the error due to the fact that we do not maximize the PI (as the true pmf is unknown) but rather its empirical estimation, since we only have a finite set of profiling traces. An upper bound of this error, based on the value of the VC-dimension, can be derived in the context of the NLL loss minimization [Vap99], thereby extending the consistency result recalled in Theorem 1 by providing convergence rates. Unfortunately, the recent deep learning literature has shown that such a bound is very conservative, regarding the potentially high value of the VC-dimension [Aro17].

The term (18) corresponds to the *optimization error*. This is the error made by the SGD algorithm, since it is not clearly proved yet to converge towards the solution NLL loss minimization. Indeed, for convex functions, SGD is shown to converge towards the minimum [SSBD14]. Unfortunately this assumption does not hold for the NLL loss applied to DNNs [GBC16]. Hopefully, the nature of the loss landscape implies that SGD still remains a good heuristic to approach the minimum [LBH15, Bot12, KLY18, CS18].

We remark that each error term refers to a restriction in the capacity of an evaluator (finite hypothesis class, finite profiling set, heuristic for MLE instead of an exact solution). That is why, in order to practically assess the quality of the estimation of the MI, it is interesting to emulate cases where such restrictions can be ignored, so that each error term can be evaluated separately. The experiments conducted in Sections 5 and 6 assess each error term.

4.4 Partial Conclusions

The results we have stated so far are threefold.

First, it has been argued that addressing the SCA Optimization Problem may be done by considering another problem, called Leakage Assessment, which aims at finding a model that extracts the most perceived information, rather than choosing the model maximizing the accuracy.

Second, the loss function we are usually minimizing, namely the NLL loss can be interpreted as a perceived information that aims at being maximized. That is why in Section 5 and Section 6, we will plot the PI, as computed with Equation 15, since it will enable to replace the accuracy in order to compare and evaluate the efficiency of a trained model.

Third, to discuss the tightness of Inequality (16), we can decompose the gap into three terms, namely the approximation error, the estimation error and the optimization error. Each error term refers to a restriction in the capacity of an evaluator. The experiments conducted in Sections 5 and 6 study the practical impact of each term.

Eventually, the whole discussion conducted in this section, aiming at emphasizing the links between machine learning concepts and metrics and the ones used in SCA, can be synthesized in Table 1.

[10] There actually exists many versions of the Universal Approximation Theorem, relying on different notions of convergence, or on particular properties of the activation function of the neural network. The interested reader may refer to [Pin99].

Table 1: machine learning metrics and their meaning in Side-Channel Analysis

ML description	ML metric		SCA metric	SCA description
Perfect model	$\mathsf{H}(Z\vert\mathbf{X})$	$=$	$n - \mathsf{MI}\left(Z;\mathbf{X}\right)$	Informational security bound on $Z\vert\mathbf{X}$
+ Approximation error	$\inf\limits_{\theta\in\Theta} \mathcal{L}_{\Pr(\mathbf{X},Z)}(\theta)$	$\Longleftrightarrow$	$\sup\limits_{\theta\in\Theta} \mathsf{PI}(\mathbf{X};Z;\theta)$	Computational bound
Cross Entropy	$\mathcal{L}_{\Pr(\mathbf{X},Z)}(\theta)$	$=$	$n - \mathsf{PI}\left(Z;\mathbf{X};\theta\right)$	Perceived Information
NLL loss	$\mathcal{L}_{S_p}(\theta)$	$=$	$n - \widehat{\mathsf{PI}_{N_p}}\left(Z;\mathbf{X};\theta\right)$	Estimated PI

5 Study on Simulated Data

This section confronts the different propositions made so far with simulated experiments. The aim of these experiments are: (1) to show experimentally that the PI, as computed in Equation 14, is indeed a lower bound of the MI; (2) to show, in some cases where we can compute the exact MI between a sensitive target variable and a leakage, that the latter lower bound is tight, so that the PI gives an accurate estimation of the MI; (3) to see to what extent the commonly used counter-measures adapted for SCA have a practical impact on the training of DNNs. To this end, we first present the settings of our simulations in Subsection 5.1, and we afterwards analyze them in Subsection 5.2.

5.1 Settings of the experiments

To verify the tightness of the bounds, we simulate simple D-dimensional leakages from an n-bit sensitive variable Z. The traces are defined such that for every $t \in [\![1, D]\!]$:

$$\mathbf{x}_i[t] = \begin{cases} U_i + B_i, \text{ if } t \notin \{t_1, \dots, t_{d+1}\} \\ hw(z_{t,i}) + B_i \text{ otherwise} \end{cases}, \tag{21}$$

where $(U_i)_i, (B_i)_i$ and all $(z_{t,i})_i$ are independent, $U_i \sim \mathcal{B}(n, 0.5)$ (*i.e.* U_i is drawn from a binomial law of parameters n and 0.5), $B_i \sim \mathcal{N}(0, \sigma^2)$, where hw denotes the Hamming weight function and where $(z_{1,i}, \dots, z_{d+1,i})$ is a $(d+1)$-sharing of z_i for the bit-wise addition law.[11] This example corresponds to a situation where the leakages of the shares are hidden among values that have no relation with the target, but have the same marginal pmf. Since the $z_{t,i}$ are drawn uniformly, $hw(z_{t,i})$ follows a binomial marginal pmf so they are indistinguishable without prior knowledge. Hence the choice of a binomial law for U_i when emulating non-informative components. Every possible combination of the $(d+1)$-sharing has been generated and replicated a given number of times (denoted by q) before adding the noise, in order to have an *exhaustive dataset*. Therefore, it contains $q \times 2^{(d+1)n}$ simulated traces. Once the data were generated, we trained a MLP with one hidden layer made of $r = 1,000$ neurons. The training loss is naturally the NLL loss.[12] The training lasts $T = 200$ epochs[13], with a Stochastic Gradient Descent and a learning rate of 10^{-3}.

Our simulations comprise three main campaigns:

Experiment 1 (masking only): in this experiment, we set $D = d + 1$ in order to avoid to consider irrelevant input features. The simulations are done over $n = 4$ bits, $d \in \{0, 1, 2, 3\}$ and $\sigma \in \{0.01, 0.1, 0.2, 0.4, 0.8, 1.6, 3.2\}$. We also generate enough data so that the training

[11] A masking scheme of order d consists in a $d + 1$-sharing of the sensitive target variable.

[12] Beware that in Pytorch and Tensorflow, the NLL loss is computed with natural logarithms, whereas one must consider the logarithm in base 2.

[13] One epoch refers to the number of iterations needed to process the whole dataset through the SGD algorithm.

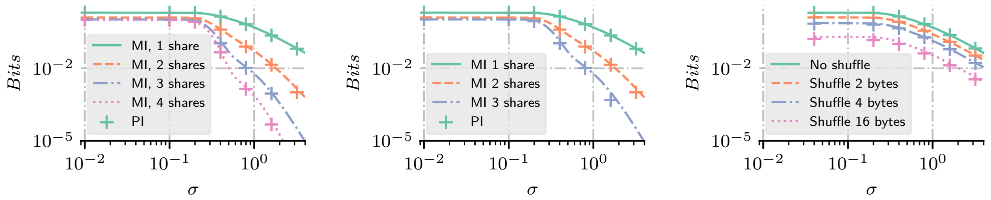

Figure 1: Information perceived by the MLP in Experiments 1 (top left), 2 (top right) and 3 (bottom).

set is "exhaustive", *i.e.* the number of replicas is $q = 2,000$. With such generated dataset, we expect to make the estimation error (19) negligible. The gap between the MI and the PI should therefore only be composed of the optimization error (18) and the approximation error (20).

Experiment 2 (masking, with uninformative components): in a second experiment, we have $D = 40$, including the uninformative components. Since all components share the same margin law, we recall that they cannot be distinguished without knowing Z. Compared to Experiment 1, we might expect the optimization error to be more important because of the potential difficulty induced by the presence of uninformative components.

Experiment 3 (shuffling, no masking): in a third experiment, we set $d = 0$, $D \in \{2, 4, 16\}$; in other words, $D - 1$ uninformative components are added like in Experiment 2, but this time they are randomly shuffled with the only informative component. Note that the shuffling is different for each simulated trace so that one cannot guess in which position the informative leakage lies. Therefore, we expect the information perceived by the model to be lower than without shuffling [VMKS12]. Besides, $\sigma \in \{0.04, 0.2, 0.4, 0.8, 1.6, 3.2\}$ here.

From those experiments, the Perceived Information $\mathsf{PI}\left(Z; \mathbf{X}; \theta_{SGD}\right)$ is estimated thanks to a hold-out dataset of $1/5$-th of the size of the training set size. For the sake of comparison, we estimate the MI between the target sensitive 4-bit variable and its simulated leakage model with a *Monte Carlo* sampling of the leakage pmf $\Pr[\mathbf{X}|Z]$.

5.2 Analysis of the Results

In this section we analyze the results obtained by running Experiments 1 (Figure 1, left), 2 (Figure 1, right) and 3 (Figure 1, bottom). On each figure, the plain lines correspond to the estimated MI and the crosses correspond to the information perceived by the trained MLP, as computed from the NLL loss with Equation 14. Based on these results, several observations can be done.

First, on each result the crosses are always below the lines, which is in line with the results given in the literature: the estimated PI is a lower bound of the MI. But more interestingly, since the crosses are always close to the line no matter the MI magnitude. In the case of Experiment 1, we argued that the error was composed of the approximation and optimization errors. Since it turns out that the sum of those errors is negligible, we conclude that even for a simple MLP with one layer and $1,000$ neurons, both errors can be ignored. This is of particular interest concerning the approximation error, as it decreases with the number of layers and the number of neurons inside each layer of the MLP. Therefore, in the case of a Hamming weight leakage model with additive Gaussian noise, any *more sophisticated* MLP (*i.e.* with more layers or more neurons by layer) will also have a negligible approximation error.

Secondly, the PI plotted in Figure 1 (right) shows that the presence of uninformative components in Experiment 2 does not annihilate the capacity of the MLP to optimally extract information about the target variable, provided that these components are not shuffled with informative ones. This shows that the optimization error, which was thought to be increased compared to Experiment 1, remains stable.

Finally, the preceding observations hold when considering masking (Figure 1, left) or shuffling (Figure 1, right). This can be interpreted as the fact that the MLP trained through the NLL loss minimization is able to give a model optimally extracting the remaining informative leakage, while being "agnostic" concerning the presence or not of such counter-measures. Nevertheless, since both counter-measures have been shown to decrease the MI (exponentially with the level of noise for masking [PR13, DFS15], or linearly for shuffling [VMKS12]), they remain sound against Deep Learning.

At this stage, we have argued thanks to our simulations that the approximation error is negligible, no matter the considered counter-measure, nor the architecture of a MLP, while the optimization error is likely to remain negligible as well. Therefore, our MI estimation obtained by PI maximization seems accurate. This provides an empirical validation of Proposition Proposition 4. As another consequence, we are fairly confident that in the case of such simple leakage models, which often happen on real use cases, replacing an optimal architecture by another should not degrade too much the MI estimation.[14] These observations must be challenged by tests on experimental traces, where one cannot have an exhaustive dataset. This will naturally lead to discussions regarding the estimation error which has not been investigated here.

6 Application on Experimental Data

So far, we have seen that deep neural networks could reach the informational security bounds of a leakage in simulated experiments, thereby giving useful estimations for the developer. This success did not rely on any prior knowledge on the leakage, but was achieved thanks to a simple MLP with one hidden layer. To confirm these observations, we propose to complete the investigations by considering experimental leakage traces. Subsection 6.1 presents the acquisition of the dataset used for the experiments, Subsection 6.2 presents the methodology of our experiments, and Subsection 6.3 discusses their results. Besides, details on the used DNN architectures for these experiments can be found in Appendix D.

6.1 Presentation of the Dataset

The leakage traces represent the power consumption of a XMEGA128D4 chip supported on a Chip Whisperer Lite board [OC14]. The program ran on the chip aims at simulating the leakage of several shares that may be processed by a protected implementation of a cryptographic primitive. The firmware is directly written in assembly code and consists in loading each byte of an input plaintext array to a register, setting it to zero and then storing it back to the input array. Some details of the code are given in Appendix B. $500,000$ traces of $2,500$ time samples each have been acquired, along with the corresponding bytes array denoted by `plain[i]`, $i \in [\![0, 15]\!]$. The complete acquisition has been done within 15 hours.

To reproduce conditions similar to the simulations, we only target the $n = 4$ most significant bits of the target variable. In other words, $|\mathcal{Z}| = 2^n = 16$. Eventually, we verified that the traces did not contain unexpected leakages that might help targeting masked variables (see Figure 4 and the corresponding discussion in Appendix C).

[14] However, one cannot get such a conclusion if one considers another leakage model.

6.2 Methodology

Common settings The trainings have been done with a variant of the SGD algorithm called *Adam* [KB15] through a number of epochs denoted by T, *i.e.* each trace has been processed T times by the Adam algorithm. Over the $500,000$ profiling traces, a portion α is used for the training, and the remaining is used as a hold-out set for computing an unbiased estimate of the perceived information. In other words, the profiling set is made of $N_p = \alpha \times 500,000$ traces while the hold-out set is made of $N_v = (1 - \alpha) \times 500,000$ traces. We fix the limit $\alpha \leq 4/5$ so that the quality of the estimation over the hold-out set remains satisfying: the error margin will be at most 10^{-2} with a confidence at least 90% in the worst case, according to Chebychev's inequality (see Appendix A).

Experiment 4 on masking When considering masking, the generated target values are $Z = \bigoplus_{i \in [\![0,d]\!]} \texttt{plain}[i]$ for $d \in \{0, 1, 2\}$, where $\oplus$ denotes the `xor` operation between two bytes. This way, it can simulate leakages of order d.

Provided with these target values, we selected Points of Interest (PoIs) based on the magnitude of the *Signal-to-Noise Ratio* [MOP07]: between 4 and 6 PoIs are selected in decreasing order of magnitude of SNR from each of the three first bytes of the plaintext array.[15] The time coordinates 13 to 16, 25 to 30 and 37 to 41 respectively correspond to the PoIs of the latter bytes manipulation. This gives an input dimension of $D = 15$. This way, we hoped to reduce the quantity of irrelevant components, which would have made the optimization with SGD harder, and therefore hoped to get a good estimate that corresponds the best to the approximation error (20). Details of the trained MLP can be found in Appendix D. We set $T = 200$ and let α vary so that $N_p \in [\![1,000; 400,000]\!]$. This way, we will be able to plot the so-called *learning curve*, namely plotting the values of $\mathsf{PI}\left(Z; \mathbf{X}; \theta_{SGD}\right)$ and $\widehat{\mathsf{PI}_{N_p}}\left(Z; \mathbf{X}; \theta_{SGD}\right)$ depending on N_p. This is a classical representation in machine learning that will enable to discuss the estimation error (19) according to the size of the profiling set.[16]

Experiment 5 on shuffling When considering shuffling, the generated target values are $Z = \texttt{plain}[i]$ where i is randomly drawn from a subset of $[\![0, 15]\!]$ of size c, c denoting the number of shuffled bytes.

Contrary to the experiments on masking, we did not selected PoIs but only restricted the target window to the $D = 250$ first time samples of the traces, which was sufficient to cover the leakages of every shuffled plaintext byte (see Appendix C). Afterwards, a CNN with a VGG-like architecture has been used for those trainings. Details of the trained CNN can be found in Appendix D.

We set $\alpha = 4/5$, $T = 100$, and $c \in \{1, 2, 4, 16\}$. The aim of this experiment is to empirically verify the trend observed on the Experiment 3 (Figure 1, bottom), namely a linear decrease of PI with the number of shuffled bytes.

6.3 Results and Discussions

Figure 2 (left) presents the learning curves of Experiment 4, when targeting respectively $1, 2$ or 3 shares among the considered ones. The dotted curves are the estimated PI over the N_p profiling traces whereas the plain curves denote the PI estimated with the N_v validation traces.

It may first be observed that the amount of information leaking on the sensitive un-split variable seems to decrease at an exponential rate in the number of shares, as expected from both theory [PR13, DFS15] and our simulations (see Section 5). More interestingly,

[15] See Figure 4 in Appendix C.

[16] On a learning curve, it is expected that the empirical PI decreases with N_p while the true PI increases, and both converge towards the supremum of the PI [Vap95].

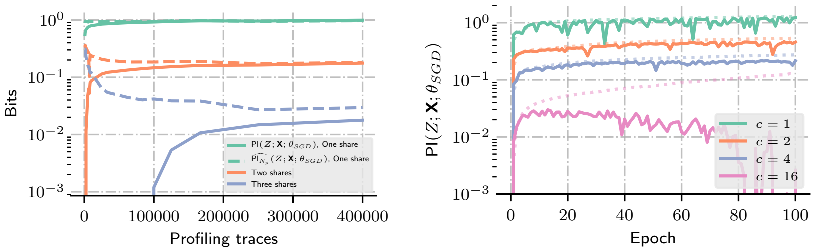

Figure 2: Left: learning curve of Experiment 4 on masking. Right: results of Experiment 5 on shuffling.

the gap between dotted curves and their corresponding plain ones exactly corresponds to the estimation error term (19). It appears then that the latter one becomes negligible relatively to the PI when the profiling set size exceeds respectively a few thousands when targeting one share, or one hundred thousand when targeting two shares. When targeting three-share, the estimation error is not completely negligible, even with $400,000$ profiling traces. It is furthermore particularly noticeable that when profiling the three shares masking scheme with less than $100,000$ traces, the learning phase completely failed since the PI was null. This indicates that, in addition to the effect on MI predicted by theoretical works [PR13, DFS15], the masking counter-measure also has an effect on the PI through an increasing estimation error, making the MI estimation poorer.

Figure 2 (right) presents the results of Experiment 5 on shuffling. It is recalled that contrary to Experiment 4 where PoIs where extracted, here 250-dimensional traces have been processed through a CNN. The gap in Figure 1 (top right) between each curve remains observable when considering experimental traces. However, the PI obtained when the attack target is shuffled among 16 random values seems decreasing starting the 20-th epoch, while the empirical PI (in dotted curves) keeps increasing. This is a sign of over-fitting, denoting a high estimation error, probably due to the high dimensionality of the traces. Therefore, the PI reached in the graph is not necessarily optimal: more profiling traces might be required to improve the CNN training.

Altogether, our experiments show that similarly to the approximation and optimization errors discussed in Section 5, the estimation error is also negligible relatively to the MI, when considering unprotected scenarios where the profiling set size is reasonably high (*i.e.* $10,000$ traces or above). This therefore leads to a tight estimation of the MI through the maximization of the PI (*i.e.* the minimization of the NLL loss). When considering protected devices, the investigated counter-measures impact the estimation error, and thereby on the tightness of the lower bound computed through PI maximization. Nevertheless this can be controlled by increasing the size of the profiling set. More precisely, the *harder* the counter-measure (*i.e.* the higher the masking order, or the more shuffled bytes), the higher the profiling set size.

Another way to decrease the estimation error would be to decrease the *capacity* of the hypotheses class, *i.e.* its VC-dimension, by decreasing the number of layers or the number of neurons on each layer. Since we have argued in Subsection 5.2 that the approximation error was negligible even for a simple architecture, we are quite confident that this would not strongly affect the quality of the MI estimation.

6.4 Application on Public Datasets

So far, we have considered our experimental investigations through the view of the Leakage Assessment Problem (*i.e.* Problem 2). However, we remind that the final task an evaluator is given to achieve is the SCA Optimization (*i.e.* Problem 1), namely to find $N_a^\star$.

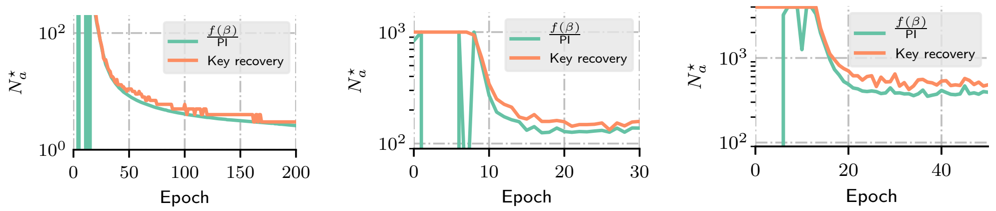

Figure 3: Comparison between the estimation of $N_a^\star$ through the lower bound (orange lines) and through a key enumeration (green lines) on the AES-RD dataset (left), the ASCAD dataset (center), and the AES-HD dataset (right).

It is recalled that Corollary 1 argued that by solving the Leakage Assessment Problem, one could get an accurate estimation $\widetilde{N}_{a,\hat{\theta}}$ of the quantity $\frac{f(\beta)}{\mathsf{MI}(Z;\mathbf{X})}$, known to be a lower-bound of the optimal solution of the SCA Optimization Problem, namely $N_a^\star$.

One could wonder whether this inequality still holds for any model, maybe sub-optimal, *i.e.* when estimating the minimal number of queries $N_a(\theta)$ to the target device for such a model with the quantity $\widetilde{N}_{a,\theta}$. A formal proof would be a promising further work, though beyond the scope of this paper. Nevertheless we propose here to empirically verify this hypothesis by training a CNN on two public datasets and by implementing the key enumeration in order to evaluate the smallest N_a such that $\mathrm{SR}(N_a) \geq \beta$, as defined in the SCA Optimization Problem. In the following, we will restrict to $\beta = 0.9$.

To this end, we considered three public datasets. The first one is the *Random Delay Counter-Measure Dataset* (AES-RD) released by Coron and Kizhvatov [CK09].[17] The target smart-card is an 8-bit Atmel AVR micro-controller, protected by a *Random Delay* counter-measure, which has an effect on the misalignment of the traces, making some attacks like Gaussian Templates much harder but possibly having no effect on deep learning based attacks [CDP17]. The targeted variable is the output of the first S-Box. $50,000$ traces of $D = 3,500$ time samples each are given in this dataset. We use $N_p = 40,000$ traces for profiling and the remaining $N_v = 10,000$ for validation.

The second one is the *ASCAD dataset* [PSB+18].[18] The target platform is an 8-bit ATMEGA8515 running a masked AES-128 implementation and measurements are made using electromagnetic emanation. The targeted variable is the output of the third S-Box. The dataset provides $60,000$ traces of $D = 700$ time samples each, where $N_p = 50,000$ traces are used for profiling and $N_v = 10,000$ for validation.

The third one is the *AES-HD* dataset, gathering traces measured on an unprotected AES-128 on FPGA. The dataset contains $100,000$ traces of $1,250$ time samples each.[19] $80,000$ traces have been used for profiling and the remaining $20,000$ have been used for the key recovery phase.

For each training, a VGG-like CNN architecture has been used. Specific details about the parameters used can be found in Appendix D. The training have been run on 200 epochs on the AES-RD, 50 epochs on the AES-HD, and stopped after 30 epochs on the ASCAD since the model started over-fitting. After each epoch, an estimation of $N_a(\theta_{SGD})$ (*i.e.* for the current model given by the Adam optimizer) is computed thanks to a key enumeration, according to the procedure detailed in Appendix E.

The results are given in Figure 3. On each graph, $\widetilde{N}_{a,\theta}$ is denoted in green, whereas the enumeration key estimation $N_a(\theta_{SGD})$ is denoted by the orange curve.[20]

[17] The traces are available at https://github.com/ikizhvatov/randomdelays-traces.

[18] The traces are available at https://github.com/ANSSI-FR/ASCAD.

[19] The traces are available at https://github.com/AESHD/AES_HD_Dataset

[20] Each curve has been clipped in order to be in $[0, 10^3]$, except for the AES-HD dataset where the high threshold is 4.10^3 since the literature presumed a higher $N_a^\star$ [KPH+19, ZBHV19].

On Figure 3 (left), we can first remark that the first epochs of the profiling of the AES-RD dataset show a chaotic behavior. This is explained by the fact that the NLL loss is initially close to $n = 8$ bits, or in other words, the PI is close to zero, leading to unstable estimations of $N_a(\theta_{SGD})$. Once the model has started extracting some information, *i.e.* after approximately 20 epochs, the PI starts to be higher than 0 and the instability vanishes. We can then observe that $N_a(\theta_{SGD})$ is always lower than the key enumeration estimation, while remaining tight through the epochs: the average relative error, computed starting the 20-th epoch is of 0.16. The final model is able to recover the secret key in 3 traces, and has a PI of 2.95 bits.

Likewise, for the ASCAD, the results are presented in Figure 3 (center). We can observe the same instability at the beginning of the training, though the quantity $\widetilde{N_{a,\theta}}$ remains lower than the estimation through the enumeration key afterwards, while staying quite tight. The average relative error is here of 0.16, and the final PI is 0.065.

Finally, for the AES-HD, the results are presented in Figure 3 (right). Similarly to the two other experiments, a tight estimation is obtained, since the relative error is 0.18, while the final PI is 0.020.

As a consequence, those three experiments enable to confirm that the quantities $\widetilde{N_{a,\theta}}$ and $N_a(\theta)$ are effectively related, at least for $\beta = 0.9$. This is of great interest in the evaluation of the security of a device, since this not only empirically shows the relevance of minimizing the NLL loss, but this also provides a relevant tool to predict the required number of queries to succeed the key recovery, or at least to give a lower-bound to such a number, which is still useful since we look for a worst case scenario in a SCA evaluation.

7 Conclusion

In this paper, we have given some theoretical and experimental reasons why the deep learning paradigm is suitable for evaluating implementations against SCA from a worst-case scenario point of view, regardless the nature of the counter-measures.

Contrary to what was commonly believed until the works of Picek *et al.* [PHJ$^+$19], the supervised classification approach is not theoretically grounded generally speaking. Yet, deep learning based attacks still worked. The reason is that in the specific case where the NLL is used as a surrogate loss function, it turns out that the latter one is actually consistent with maximizing the PI, solving the so-called Leakage Assessment Problem. Since the latter problem was argued to be sound with the SCA Optimization Problem, we conclude that the choice of the NLL as a surrogate loss function is sound from an evaluation point of view, in the sense that it enables to accurately estimate a lower bound of the minimal number of queries required by an attacker provided with an optimal leakage model in order to successfully recover the secret key.

Simulations and experiments verified that the PI maximization via NLL minimization was an efficient method in order to estimate the MI in several configurations, *i.e.* on different architectures and with different types of counter-measures, including higher order masking, shuffling or de-synchronization through random delays.

This leads to the takeaway messages of this paper: the minimization of the NLL loss via a neural network model enables to give relevant estimations of the mutual information between a sensitive variable and the corresponding side-channel traces, thereby quantitatively measuring the impact of counter-measures (and their implementations) so that an evaluator can precisely assess whether the latter one stays sound or not.

References

[Aro17] Sanjeev Arora. Generalization Theory and Deep Nets, An introduction. http://offconvex.github.io/2017/12/08/generalization1/, 2017.

[BGH+17] Nicolas Bruneau, Sylvain Guilley, Annelie Heuser, Damien Marion, and Olivier Rioul. Optimal side-channel attacks for multivariate leakages and multiple models. J. Cryptographic Engineering, 7(4):331–341, 2017.

[BGHR14] Nicolas Bruneau, Sylvain Guilley, Annelie Heuser, and Olivier Rioul. Masks will fall off - higher-order optimal distinguishers. In Palash Sarkar and Tetsu Iwata, editors, Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II, volume 8874 of Lecture Notes in Computer Science, pages 344–365. Springer, 2014.

[BGP+11] Lejla Batina, Benedikt Gierlichs, Emmanuel Prouff, Matthieu Rivain, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Mutual information analysis: a comprehensive study. J. Cryptology, 24(2):269–291, 2011.

[BHM+19] Olivier Bronchain, Julien M. Hendrickx, Clément Massart, Alex Olshevsky, and François-Xavier Standaert. Leakage certification revisited: Bounding model errors in side-channel security evaluations. In Alexandra Boldyreva and Daniele Micciancio, editors, Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I, volume 11692 of Lecture Notes in Computer Science, pages 713–737. Springer, 2019.

[Bot12] Léon Bottou. Stochastic gradient descent tricks. In Grégoire Montavon, Genevieve B. Orr, and Klaus-Robert Müller, editors, Neural Networks: Tricks of the Trade - Second Edition, volume 7700 of Lecture Notes in Computer Science, pages 421–436. Springer, 2012.

[BR14] Lejla Batina and Matthew Robshaw, editors. Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings, volume 8731 of Lecture Notes in Computer Science. Springer, 2014.

[CDP17] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures - profiling attacks without pre-processing. In Wieland Fischer and Naofumi Homma, editors, Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings, volume 10529 of Lecture Notes in Computer Science, pages 45–68. Springer, 2017.

[CK09] Jean-Sébastien Coron and Ilya Kizhvatov. An efficient method for random delay generation in embedded software. In Christophe Clavier and Kris Gaj, editors, Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings, volume 5747 of Lecture Notes in Computer Science, pages 156–170. Springer, 2009.

[Cra99] Harald Cramér. Mathematical methods of statistics. Princeton University Press, 1999. OCLC: 185436716.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers, volume 2523 of Lecture Notes in Computer Science, pages 13–28. Springer, 2002.

[CS18] Pratik Chaudhari and Stefano Soatto. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net, 2018.

[CT06] Thomas M. Cover and Joy A. Thomas. Elements of information theory (2. ed.). Wiley, 2006.

[dCGRP19] Eloi de Chérisey, Sylvain Guilley, Olivier Rioul, and Pablo Piantanida. Best information is most successful mutual information and success rate in side-channel analysis. IACR Trans. Cryptogr. Hardw. Embed. Syst., 2019(2):49–79, 2019.

[DFS15] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete - or how to evaluate the security of any leaking device. In Elisabeth Oswald and Marc Fischlin, editors, Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I, volume 9056 of Lecture Notes in Computer Science, pages 401–429. Springer, 2015.

[GBC16] Ian J. Goodfellow, Yoshua Bengio, and Aaron C. Courville. Deep Learning. Adaptive computation and machine learning. MIT Press, 2016.

[GHO15] Richard Gilmore, Neil Hanley, and Máire O'Neill. Neural network based attack on a masked implementation of AES. In IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2015, Washington, DC, USA, 5-7 May, 2015, pages 106–111. IEEE Computer Society, 2015.

[HGM$^+$11] Gabriel Hospodar, Benedikt Gierlichs, Elke De Mulder, Ingrid Verbauwhede, and Joos Vandewalle. Machine learning in side-channel analysis: a first study. J. Cryptographic Engineering, 1(4):293–302, 2011.

[HRG14] Annelie Heuser, Olivier Rioul, and Sylvain Guilley. Good is not good enough - deriving optimal distinguishers from communication theory. In Batina and Robshaw [BR14], pages 55–74.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis R. Bach and David M. Blei, editors, Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015, volume 37 of JMLR Workshop and Conference Proceedings, pages 448–456. JMLR.org, 2015.

[KB15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.

[KLY18] Robert Kleinberg, Yuanzhi Li, and Yang Yuan. An alternative view: When does SGD escape local minima? In Jennifer G. Dy and Andreas Krause, editors, Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018, volume 80 of Proceedings of Machine Learning Research, pages 2703–2712. PMLR, 2018.

[KPH+19] Jaehun Kim, Stjepan Picek, Annelie Heuser, Shivam Bhasin, and Alan Hanjalic. Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis, 2019.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States., pages 1106–1114, 2012.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. Deep learning. Nature, 521(7553):436–444, 2015.

[LBM14] Liran Lerman, Gianluca Bontempi, and Olivier Markowitch. Power analysis attack: an approach based on machine learning. IJACT, 3(2):97–115, 2014.

[LBM15] Liran Lerman, Gianluca Bontempi, and Olivier Markowitch. A machine learning approach against a masked AES - reaching the limit of side-channel attacks with a learning model. J. Cryptographic Engineering, 5(2):123–139, 2015.

[LPB+15] Liran Lerman, Romain Poussier, Gianluca Bontempi, Olivier Markowitch, and François-Xavier Standaert. Template attacks vs. machine learning revisited (and the curse of dimensionality in side-channel analysis). In Stefan Mangard and Axel Y. Poschmann, editors, Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers, volume 9064 of Lecture Notes in Computer Science, pages 20–33. Springer, 2015.

[Man04] Stefan Mangard. Hardware countermeasures against DPA ? A statistical analysis of their effectiveness. In Tatsuaki Okamoto, editor, Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings, volume 2964 of Lecture Notes in Computer Science, pages 222–235. Springer, 2004.

[MDM16] Zdenek Martinasek, Petr Dzurenda, and Lukas Malina. Profiling power analysis attack based on MLP in DPA contest V4.2. In 39th International Conference on Telecommunications and Signal Processing, TSP 2016, Vienna, Austria, June 27-29, 2016, pages 223–226. IEEE, 2016.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. Power analysis attacks - revealing the secrets of smart cards. Springer, 2007.

[MPP16] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. In Claude Carlet, M. Anwar Hasan, and Vishal Saraswat, editors, Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings, volume 10076 of Lecture Notes in Computer Science, pages 3–26. Springer, 2016.

[MZ13] Zdenek Martinasek and Vaclav Zeman. Innovative method of the power analysis. Radioengineering, 22:586–594, 06 2013.

[Nie18] Michael A. Nielsen. Neural networks and deep learning. http://neuralnetworksanddeeplearning.com/, 2018.

[OC14] Colin O'Flynn and Zhizhang (David) Chen. Chipwhisperer: An open-source platform for hardware embedded security research. In Emmanuel Prouff, editor, Constructive Side-Channel Analysis and Secure Design - 5th International Workshop, COSADE 2014, Paris, France, April 13-15, 2014. Revised Selected Papers, volume 8622 of Lecture Notes in Computer Science, pages 243–260. Springer, 2014.

[Pet98] P. Petrushev. Approximation by ridge functions and neural networks. SIAM Journal on Mathematical Analysis, 30(1):155–189, 1998.

[PHJ+19] Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni. The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. IACR Trans. Cryptogr. Hardw. Embed. Syst., 2019(1):209–237, 2019.

[Pin99] Allan Pinkus. Approximation theory of the MLP model in neural networks. Acta Numerica, 8:143–195, 1999.

[PR10] Emmanuel Prouff and Matthieu Rivain. Theoretical and practical aspects of mutual information-based side channel analysis. IJACT, 2(2):121–138, 2010.

[PR13] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Thomas Johansson and Phong Q. Nguyen, editors, Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings, volume 7881 of Lecture Notes in Computer Science, pages 142–159. Springer, 2013.

[PSB+18] Emmanuel Prouff, Remi Strullu, Ryad Benadjila, Eleonora Cagli, and Cecile Dumas. Study of deep learning techniques for side-channel analysis and introduction to ascad database. Cryptology ePrint Archive, Report 2018/053, 2018. https://eprint.iacr.org/2018/053.

[RSV+11] Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre. A formal study of power variability issues and side-channel attacks for nanoscale devices. In Kenneth G. Paterson, editor, Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings, volume 6632 of Lecture Notes in Computer Science, pages 109–128. Springer, 2011.

[SHK+14] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1):1929–1958, 2014.

[SMY09] François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In Antoine Joux, editor, Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings, volume 5479 of Lecture Notes in Computer Science, pages 443–461. Springer, 2009.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press, 2014.

[STIM18] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada., pages 2488–2498, 2018.

[SVO$^+$10] François-Xavier Standaert, Nicolas Veyrat-Charvillon, Elisabeth Oswald, Benedikt Gierlichs, Marcel Medwed, Markus Kasper, and Stefan Mangard. The world is not enough: Another look on second-order DPA. In Masayuki Abe, editor, Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings, volume 6477 of Lecture Notes in Computer Science, pages 112–129. Springer, 2010.

[Tim19] Benjamin Timon. Non-profiled deep learning-based side-channel attacks with sensitivity analysis. IACR Trans. Cryptogr. Hardw. Embed. Syst., 2019(2):107–131, 2019.

[Vap95] V. Vapnik. The Nature of Statistical Learning Theory. Information Science and Statistics. Springer New York, 1995.

[Vap99] Vladimir Vapnik. An overview of statistical learning theory. IEEE Trans. Neural Networks, 10(5):988–999, 1999.

[vdVP19] Daan van der Valk and Stjepan Picek. Bias-variance decomposition in machine learning-based side-channel analysis. IACR Cryptology ePrint Archive, 2019:570, 2019.

[VMKS12] Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against side-channel attacks: A comprehensive study with cautionary note. In Xiaoyun Wang and Kazue Sako, editors, Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings, volume 7658 of Lecture Notes in Computer Science, pages 740–757. Springer, 2012.

[WMM19] Felix Wegener, Thorben Moos, and Amir Moradi. DL-LA: deep learning leakage assessment: A modern roadmap for SCA evaluations. IACR Cryptology ePrint Archive, 2019:505, 2019.

[ZBHV19] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Methodology for efficient CNN architectures in profiling attacks. IACR Cryptology ePrint Archive, 2019:803, 2019.

A Confidence Interval with a Hold-Out Set

The bounds on the estimation error, discussed in Subsection 4.3 might be too high in practical uses with Deep Neural Nets. This is why usually, the estimation of the Cross Entropy is done otherwise. In cases where the evaluator has lots of data, he can take a so-called *hold-out* that will be distinct from the profiling set S_p for the minimization of the NLL loss. Therefore, these *fresh* data will give a more correct estimation of the Cross Entropy.

Lemma 1 (Chebychev's inequality [SSBD14])**.** *Let $\mathcal{H}$ be a parametrized hypothesis class and Θ its corresponding parameter space. Let $S_v \triangleq \{(\boldsymbol{x}_1, z_1), \ldots, (\boldsymbol{x}_{N_v}, z_{N_v})$ be a hold-out set of N_v i.i.d. leakages and the corresponding values of the sensitive target variable. Assume that $\mathbb{V}(-\log_2 F(\boldsymbol{X}, \theta)[Z]) \leq 1$. Then for all $\theta \in \Theta$, it holds with probability at least $1 - \delta$ that:*

$$|\mathcal{L}_{\Pr(\boldsymbol{X},Z)}(\hat{\theta}) - \mathcal{L}_{S_v}(\hat{\theta})| \leq \sqrt{\frac{1}{\delta N_v}} \tag{22}$$

Equation 22 will be used in Section 5 and Section 6 to estimate provide a conservative confidence interval of the Cross Entropy and thereby the information between a sensitive target variable and a leakage perceived by a Neural Net (PI).

B Source Code for the Acquisitions

Algorithm 1 loadData

```
LD r0, X          ▷ Loads the first byte in r0
CLR r0            ▷ Clears the register
ST X, r0          ▷ Stores 0 in the plaintext array
LD r0, X          ▷ Do it again to clear the bus
CLR r0
ST X, r0
LD r0, X          ▷ One more time to be sure
CLR r0
ST X+, r0
```

C The Experimental Traces

To verify that there is no leakage implying a combination of different bytes, we have also computed a SNR of *order 2*. That is to say that for each combination of 2 among the 16 bytes, the `xor` has been computed and used as a target variable in order to compute the SNR. The absence of peaks confirms that there is no undesirable leakage. An example of a trace and the SNRs of order 1 and 2 can be found in Figure 4.

D The hypothesis class $\mathcal{H}$

The hypothesis class $\mathcal{H}$ that will be used for the experiment has been defined as the set of MLP with one hidden layer and $r = 500$ hidden neurons. In other words, there are two *linear* layers: the hidden one, denoted by λ_{θ_1}, and the output one denoted by λ_{θ_2}. Symbols θ_1 and θ_2 denote the associated real parameters of the hidden layer and the output layer respectively. Between these linear layers, an *activation* layer called *ReLU* and denoted

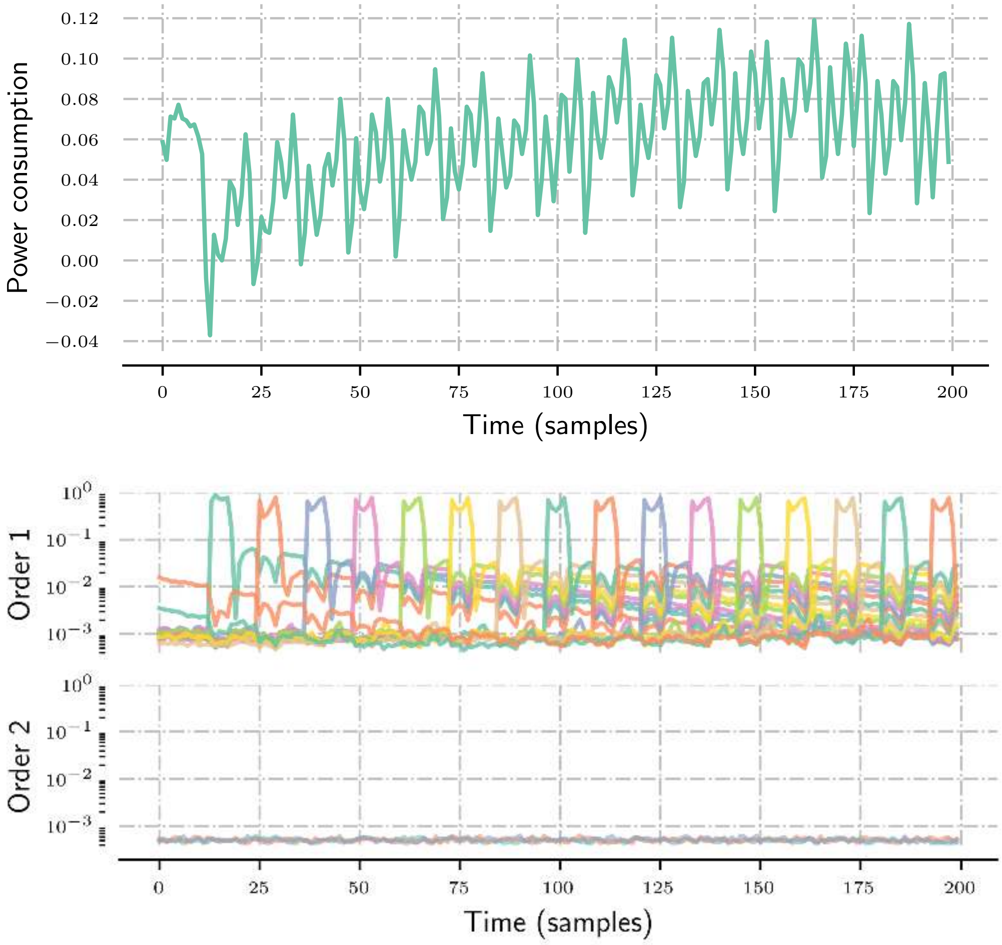

Figure 4: Top: an example of trace. Bottom: the SNRs of order 1 and 2 (in logarithm scale).

by σ is added. This is a non-linear real valued function that is responsible of the high capacity of MLP to approximate any pmf [GBC16].

In addition, two *batch normalization* layers [IS15] have been applied at the input of each linear layer. Batch Normalization (BN) layers simply normalize the input features to a mean and a deviation that are automatically set by the SGD algorithm. BN has been shown to make the loss function smoother, making the optimization easier and faster [STIM18]. Finally, a *dropout* layer δ_p [SHK+14, GBC16] has been added on the input of the softmax classifier. Dropout is known to prevent DNNs from *overfitting* (*i.e.* to prevent the estimation error to explode), which is useful when one lacks data. The dropout parameter has been set to $p = 0.1$ *i.e.* each neuron of the hidden layer is randomly set to 0 with probability p each time an output $F(\mathbf{x}, \theta)$ is computed during the optimization.

All together, we can sum up the architecture of our MLP as follows:

$$F(\mathbf{x}, \theta) = s \circ \lambda_{\theta_2} \circ \mu \circ \delta_p \circ \sigma \circ \lambda_{\theta_1} \circ \mu(x) \tag{23}$$

For the experiment on the ASCAD dataset, we have considered the same architecture as proposed in [CDP17, PSB+18], with the same notations:

$$s \circ [\lambda \circ \sigma]^{n_1} \circ \delta_G \circ [\delta \circ \sigma \circ \mu \circ \gamma]^{n_3} \ , \tag{24}$$

where γ denotes a convolutional layer, σ denotes an activation function *i.e.* a non-linear function applied elementwise, μ denotes a batch-normalization layer, δ denotes an average pooling layer, λ denotes a dense layer and s denotes the softmax layer. Furthermore, n_1 denotes the number of *dense blocks*, namely the composition $[\lambda \circ \sigma]$. Likewise, n_3 denotes the number of *convolutional blocks*, namely $[\delta \circ \sigma \circ \mu \circ \gamma]$. A *global pooling layer* δ_G, has been added at the top of the last block. Its pooling size equals the width of the feature maps in the last convolutional layer, so that each feature maps are reduced to one point.

More specifically, the following parameters have been used: $n_1 = 2$, $n_3 = 7$, the convolutional filters are of length 11. 10 filters are in the first layer, and they are doubled at each convolutional layer. The dense layers contains $1,000$ intermediate neurons. The same architecture has been used in the experiments on the AES-HD dataset.

For experiments on the AES-RD dataset, the same *VGG-like* architecture as the one presented in [KPH$^+$19] has been used. More specifically, n_3 is set to 9 so that there is enough pooling layers to get feature maps on the last convolutional layer whose width equals one. Besides, $n_1 = 0$, *i.e.* there is no intermediate dense layer, except softmax.

E Success Rate Estimation

In practice, to compute $\mathrm{SR}(N_a)$, sampling many attack sets may be very prohibitive in an evaluation context, especially if we need to reproduce the estimations for many values of N_a until we find the smallest value such that ; one solution to circumvent this problem is, given a validation set S_v of N_v traces, to sample some attack sets by permuting the order of the traces into the validation set (*e.g.* 500 times in our experiments). $\mathbf{d}_{S_a}$ can then be computed with a cumulative sum to get a score for each $N_a \in [\![1, N_v]\!]$, and so is $g_{S_a}(k^\star)$. For each value of N_a, the success rate is estimated by the occurence frequency of the event $g_{S_a}(k^\star) = 1$.[21]

[21] While this trick gives good estimations for $N_a \ll N_v$, one has to keep in mind that the estimates become biased when $N_a \to N_v$. Hopefully, in our experiments, the validation set size remains much higher than *numTracesAttacks* afterwards.