IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2020, No. 3, pp. 73–96. DOI:10.13154/tches.v2020.i3.73-96

A Novel Evaluation Metric for Deep Learning-Based Side Channel Analysis and Its Extended Application to Imbalanced Data

Jiajia Zhang, Mengce Zheng, Jiehui Nan, Honggang Hu and Nenghai Yu

Key Laboratory of Electromagnetic Space Information, CAS
University of Science and Technology of China, Hefei, China
{zhang6,ustcnjh}@mail.ustc.edu.cn, {mczheng,hghu2005,ynh}@ustc.edu.cn

Abstract. Since Kocher (CRYPTO'96) proposed timing attack, side channel analysis (SCA) has shown great potential to break cryptosystems via physical leakage. Recently, deep learning techniques are widely used in SCA and show equivalent and even better performance compared to traditional methods. However, it remains unknown why and when deep learning techniques are effective and efficient for SCA. Masure et al. (IACR TCHES 2020(1):348–375) illustrated that deep learning paradigm is suitable for evaluating implementations against SCA from a worst-case scenario point of view, yet their work is limited to balanced data and a specific loss function. Besides, deep learning metrics are not consistent with side channel metrics. In most cases, they are deceptive in foreseeing the feasibility and complexity of mounting a successful attack, especially for imbalanced data.
To mitigate the gap between deep learning metrics and side channel metrics, we propose a novel Cross Entropy Ratio (CER) metric to evaluate the performance of deep learning models for SCA. CER is closely related to traditional side channel metrics Guessing Entropy (GE) and Success Rate (SR) and fits to deep learning scenario. Besides, we show that it works stably while deep learning metrics such as accuracy becomes rather unreliable when the training data tends to be imbalanced. However, estimating CER can be done as easy as natural metrics in deep learning algorithms with low computational complexity. Furthermore, we adapt CER metric to a new kind of loss function, namely CER loss function, designed specifically for deep learning in side channel scenario. In this way, we link directly the SCA objective to deep learning optimization. Our experiments on several datasets show that, for SCA with imbalanced data, CER loss function outperforms Cross Entropy loss function in various conditions.

Keywords: Side Channel Analysis · Deep Learning · Evaluation Metric · Imbalanced Data

1 Introduction

1.1 Overview

Side channel analysis has received a significant amount of attention since it was first proposed in [Koc96]. It exploits weakness of physical implementations instead of the algorithms, bringing a serious threat to various electronic devices. Frequently used physical leakages including power consumption and electromagnetic (EM) emanation, can be easily obtained with high resolution and low cost. There are many kinds of SCA against different situations to recover the secret keys. Among all of them, profiling attacks are the most powerful ones, which can even break implementations with side channel countermeasures

Received: 2020-01-15 Accepted: 2019-03-15 Published: 2020-06-19

such as masking [ISW03] or random delay. Template Attack (TA) [CRR02] proposed in 2002 is the first kind of profiling attacks, and since then several works have been done to investigate the best way to apply it in practice. As an important part, dimension reduction methods are extensively studied to reduce the computational complexity. Typical tools such as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are widely used for preprocessing in TA [APSQ06, SA08, RO04, CK13, CDP15]. There are two basic assumptions for TA. The first one is the profiling assumption, a fundamental assumption for profiling attacks, which means the attacker can manipulate a cloned device of his target arbitrarily to establish a "template" for later attack. The other one is Gaussian noise assumption, which means the noise of side channel leakage follows the Gaussian distribution. However, Gaussian noise assumption can be violated in practice sometimes and thus it will be ineffective or impossible to conduct TA.

As an alternative of TA, deep learning-based SCA makes no assumptions on the noise. In recent study of deep learning techniques in side channel scenario, it shows equivalent performance compared to TA and even outperforms it sometimes. However, it remains questionable why and when deep learning methods are suitable for SCA. Nevertheless deep learning methods seem promising and are investigated in many side channel researches [CDP17, PSB+18, KPH+19].

1.2 Related Work

The gap between deep learning and SCA has been noticed in recent years since deep learning techniques show great potential in SCA but no grounded theory is found. Deep learning methods work well on SCA in practice by mapping it to a supervised classification problem, which is thoroughly studied. However, unlike TA, deduced from Gaussian noise assumption and Bayes' theorem, it is still not clear whether SCA is equivalent to a classification problem. Recently Masure et al. [MDP19] found Negative Log Likelihood (NLL) loss (a.k.a. Cross Entropy loss, a regularly used loss function for classification in deep learning) is related to Perceived Information [RSV+11, BHM+19], a lower bound of Mutual Information between the sensitive intermediate and the leakage. The result indicates training the deep learning model might be sound for side channel scenario and relate to the lower bound of the number of traces needed to achieve a given SR in a practical attack. Thus deep learning paradigm is suitable for evaluating implementations against SCA from a worst-case scenario point of view. However, their work is limited to NLL loss function and balanced data since they targeted on sensitive intermediates assumed to be uniformly distributed.

Another gap between deep learning and SCA is deep learning metrics are not consistent with side channel metrics. In deep learning community, researchers use metrics such as accuracy and precision to evaluate the performance of a model. However, in side channel community, most common evaluation metrics are Guess Entropy (GE) and Success Rate (SR) [SMY09], which are totally different from deep learning ones. Unfortunately, estimating GE/SR is not an easy work and costs a lot of time since it requires several independent sets of traces to mount attacks. Besides, deep learning metrics might be misleading when used in SCA according to [CDP17, PHJ+19], which can be even worse for imbalanced data. As stated in [PHJ+19], how to design a new metric that reflects the SCA behavior better with reasonable computational complexity is an open question. To the best of our knowledge, until now there is no easily computed metric that is appropriate to evaluate the performance of a deep learning model for SCA.

Deep learning-based SCA also faces the problem of imbalanced data as the original deep learning algorithms do. Imbalanced training data will greatly influence the performance of deep learning models. Consider an imbalanced training set which has two classes and the proportion of the majority class is 90%. The deep learning model trained on this set will be biased and tends to give predictions towards the majority class. In this case, the model has a very high accuracy (normally higher than 90%) on the training set, but it will show significant performance penalty when applied in practice since it learns little information about the minority class. Generally, the more the imbalance is, the more the deep learning model will suffer. To solve the problems of imbalance data, many methodologies have been developed, like using resampling techniques to make the data balanced again, or revising the training algorithm to compensate for the minority class [Mor16, HG09, HG09, SWK09, SKWW07]. Deep learning-based SCA also suffers from imbalanced data since hamming weight or hamming distance labels are commonly used while highly imbalanced. Though we can use sensitive intermediates directly as labels which are balanced, they need more data for training due to too many classes. Recently, techniques from deep learning community have been introduced in [PHJ$^+$19] to deal with imbalanced data. Though the results are encouraging, they only focus on the first order case without masking countermeasures. Thus the imbalanced problems for high order scenario are still unsolved.

1.3 Our Contributions

In this paper, we propose a novel side channel metric from deep learning' perspective for the first time and properly solve the open question in [PHJ$^+$19]. This metric is further extended to a new kind of loss function for deep learning-based SCA. In detail, our main contributions in this paper are as follows:

1. We propose a deep learning-based side channel evaluation metric Cross Entropy Ratio (CER), which can be used to evaluate the performance of deep learning models for SCA. Since estimating CER does not require to mount piratical attacks like GE/SR, it can be embedded in deep learning algorithms easily and calculated as fast as deep learning metrics;

2. CER is closely related to commonly used side channel metrics GE/SR both in theory and our experiments. Hence it extends the metric family for SCA performance evaluation and fits to deep learning algorithms. Besides, it works stably with imbalanced data in different levels;

3. We adapt CER metric to a new kind of loss function for deep neural networks. Using this new loss function in the training process, we gain significant improvements on the attacking performance when dealing with imbalanced data. We show that it works for implementations in software and hardware, with/without random delays and with/without masks (i.e. both first order and high order cases). Finally we properly solve the problems of imbalanced data in side channel scenario.

1.4 Organization

The paper is organized as follows: Section 2 introduces the background. In Section 3 we propose a novel side channel evaluation metric CER. Section 4 shows the relation between CER and GE/SR. We also give an important application of CER, that is, as the loss function of deep neural networks and show its great advantages when dealing with imbalanced data in Section 5. Finally, Section 6 concludes this paper.

2 Background

2.1 Notations

Throughout the paper, we use calligraphic letters $\mathcal{X}$ to denote sets and upper-case letter X to denote random variables over $\mathcal{X}$. The cardinality of $\mathcal{X}$ is denoted by $|\mathcal{X}|$. We use lower case letters x to denote a sample or realization of X. Respectively, we use bold character $\mathbf{X}$ to denote random vectors and $\mathbf{x}$ to denote a sample of the random vector $\mathbf{X}$. $\mathbf{X}[i]$ (resp. $\mathbf{x}[i]$) denotes the i-th element of $\mathbf{X}$ (resp. $\mathbf{x}$), which is also a random variable (resp. a sample of the random variable). Since we have random variables, we can define probability over random variables. $\Pr[X]$ denotes the distribution of X and $\Pr[X = x]$ (or $\Pr[x]$ without loss of generality) is the probability when X equals x. We use $\mathbb{E}[X]$ to denote the expectation of the random variable X and the condition will be specified under the expectation symbol or as a subscript, e.g. $\underset{X\sim\Pr[X]}{\mathbb{E}}$ and $\mathbb{E}_X$. Similarly, we use $\mathbb{V}$ to denote variance.

We use G as a cryptographic primitive which computes a sensitive intermediate $V = G(K, P)$ with a known variable P and a part of unknown secret key K the attacker aims to recover. The image set of G is denoted by $\mathcal{V}$, where sensitive intermediates take values. We denote $f : \mathcal{V} \to \mathcal{L}$ the mapping to transform the sensitive intermediate V to a label L, which will be used in the training of deep neural networks. $\mathcal{L}$ denotes the label space or equivalently image space of f. Then without loss of generality the labels can be written as $l^0, l^1, l^2, \dots l^{|\mathcal{L}|-1}$. For example, f can be the identity function or the hamming weight function.

Let X_n be a sequence of random variables and X be another random variable. We say that X_n converges in probability to X, denoted as $X_n \xrightarrow{p} X$ for $n \to \infty$ when the following property holds:

$$\forall\epsilon > 0, \lim_{n\to\infty} \Pr[|X_n - X| \geq \epsilon] = 0.$$

Let X be a random variable with finite expected value and finite non-zero variance. For any $b > 0$, Chebyshev' inequality holds:

$$\Pr[|X - \mathbb{E}[X]| \geq b] \leq \frac{\mathbb{V}[X]}{b^2}.$$

Through Chebyshev' inequality we can get the Law of Large Numbers (**LLN**). Let $X_1, X_2, \dots, X_n$ be i.i.d with finite expected value $\mathbb{E}[X_1] = \mathbb{E}[X_2] = \cdots = \mu$, then the sample average

$$\overline{X} = \frac{1}{n}(X_1 + X_2 + \cdots + X_n)$$

converges in probability to the expected value, i.e.,

$$\overline{X} \xrightarrow{p} \mu \text{ for } n \to \infty.$$

2.2 Deep Neural Network

2.2.1 MLP and CNN

First we simply introduce two widely used deep neural networks Multi-Layer Perceptron (MLP) and Convolutional Neural Network (CNN) for supervised learning, where training data is taged with corresponding labels. For a given input, they can give the best possible prediction with high accuracy or probabilities for all possible classes after a prior training process.

Inspired by the human nervous system, MLP uses artificial neurons, i.e. perceptrons (1(a)) as the minimal units to build a full network. For a given input, a perceptron first performs an affine transformation and then feed the result to an activation function to get the final output. Denote $\mathbf{w}$, b parameters for the affine transformation and Act the activation function, and then this process can be written as

$$\mathrm{Act}(\mathbf{w} \cdot \mathbf{x} + \mathrm{b}).$$

Perceptrons can be joined together to form a MLP by connecting each layer's output to the next layer's input (1(b)). Each layer also performs an affine transformation followed by an activation function with multidimensional inputs and outputs. Denote λ the affine transformation (a.k.a. fully connected layer) and Act the activation function, and then this process can be written as

$$\mathrm{Act} \circ \lambda(\mathbf{x}).$$

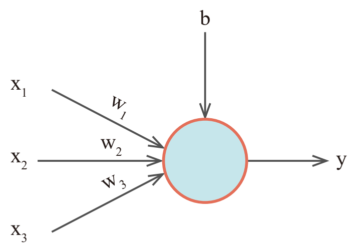
So the model of MLP with parameter θ can be expressed as follows:

$$\mathrm{M}_\theta(\mathbf{x}) = \mathrm{Act}_n \circ \lambda_n \circ \mathrm{Act}_{n-1} \circ \lambda_{n-1} \circ \cdots \circ \mathrm{Act}_1 \circ \lambda_1(\mathbf{x}).$$

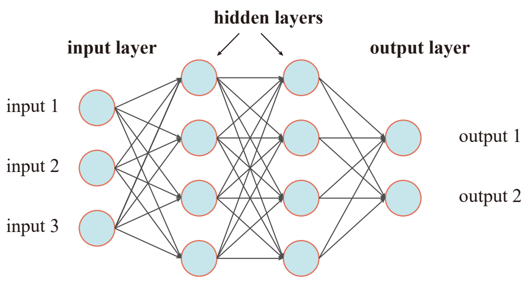
$\mathrm{Act}_1, \ldots, \mathrm{Act}_{n-1}$ can be chosen from various activation functions (e.g. ReLU and its variants) whereas Act_n is always a softmax function

$$softmax(\mathbf{x})[j] = \frac{e^x[j]}{\sum_{i=1}^{c} e^x[i]},$$

where c is the number of classes/outputs. This activation function guarantees the output of the model can be interpreted as probabilities for all possible classes, which is useful for later classification.



(a) Single perceptron



(b) MLP

Figure 1: Single perceptron and MLP.

CNN has a similar hierarchical structure with various types of layers. It usually contains convolutional layers, pooling layers, batch normalization layers, and fully connected layers followed by softmax activation function at the end of the network (Figure 2). It is a popular candidate in side channel domain to deal with traces with random delays or clock jitters due to its shift invariant property. Nowadays CNN is widely applied in many research areas and is proved to be powerful classifier for time series data [HCE+17, Kim14, KSH12]. More details about CNN can be found in literature of deep learning [RW17, GBC16].

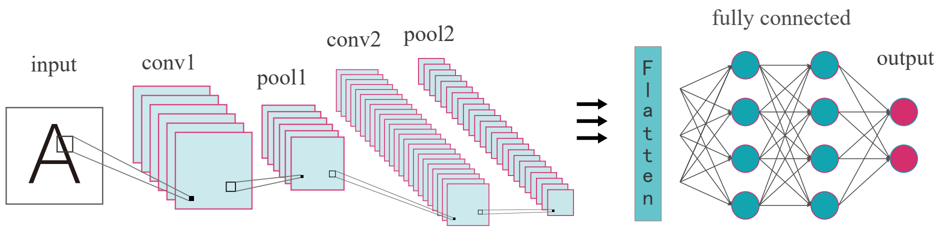


Figure 2: Convolutional Neural Network.

2.2.2 Cross Entropy Loss Function

Loss function is the key point in supervised learning since the parameters of the deep neural network are exactly updated by minimizing the loss function in the back propagation algorithm. Among all the loss functions for supervised classifications in deep learning, Cross Entropy loss is frequently used due to its stable performance and high efficiency. In deep-learning based SCA, many previous works use Cross Entropy loss for their training [CDP17, PSB$^+$18, KPH$^+$19], some works also use Mean Square Error [Tim19], [WMM19] or self defined loss function [RQL19].

Given a joint probability distribution of the label L and corresponding leakage $\mathbf{X}$ denoted as $\Pr[\mathbf{X}, Y]$ and a predicted distribution $\mathrm{M}_\theta(\mathbf{X})[Y]$ of a deep learning model M_θ with parameter θ, we define the Cross Entropy (CE) for the two distributions as follows:

$$H(\Pr[\mathbf{X}, Y], \mathrm{M}_\theta(\mathbf{X})[Y])) = \mathop{\mathbb{E}}_{\mathbf{X},Y} - \log_2 \mathrm{M}_\theta(\mathbf{X})[Y]$$

Let $\mathcal{T} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ be the training data with labels and M_θ a deep learning model with parameter θ. Then the Cross Entropy loss $Loss_{CE}(\mathcal{T}, \mathrm{M}_\theta)$ is defined as the estimation of Cross Entropy over the training set and the model[1], i.e.,

$$H_N(\mathcal{T}, \mathrm{M}_\theta) = -\sum_{i=1}^{N} \frac{1}{N} \log_2 \mathrm{M}_\theta(\mathbf{x_i})[y_i]. \tag{1}$$

Apparently Equation 1 is a Monte Carlo estimate of the Cross Entropy for the original distribution $\Pr[\mathbf{X}, Y]$ and the predicted distribution $\mathrm{M}_\theta(\mathbf{X})[Y]$, where the training set is treated as samples from $\Pr[\mathbf{X}, Y]$. That's why it is called Cross Entropy loss. Actually, the expectation of Equation 1 is exactly the Cross Entropy between the former distributions since:

$$\begin{aligned} \mathop{\mathbb{E}}_{\mathbf{x},y \sim \Pr[\mathbf{X},Y]} H_N(\mathcal{T}, \mathrm{M}_\theta) &= \mathop{\mathbb{E}}_{\mathbf{x_i},y_i \sim \Pr[\mathbf{X},Y]} -\sum_{i=1}^{N} \frac{1}{N} \log_2 \mathrm{M}_\theta(\mathbf{x_i})[y_i] \\ &= \frac{1}{N} \sum_{i=1}^{N} \mathop{\mathbb{E}}_{\mathbf{x_i},y_i \sim \Pr[\mathbf{X},Y]} - \log_2 \mathrm{M}_\theta(\mathbf{x_i})[y_i] \\ &= H(\Pr[\mathbf{X}, Y], \mathrm{M}_\theta(\mathbf{X})[Y])) \end{aligned} \tag{2}$$

and by **LNN**, we also have:

$$H_N(\mathcal{T}, \mathrm{M}_\theta) \xrightarrow{p} H(\Pr[\mathbf{X}, Y], \mathrm{M}_\theta(\mathbf{X})[Y]) \text{ for } n \to \infty. \tag{3}$$

Equation 1 is also called Negative Log Likelihood (NLL) loss in [MDP19] as minimizing the CE loss is exactly maximizing the Log Likelihood hence minimizing NLL loss. In the rest of the paper, we use CE loss for convenience.

2.3 Deep Learning-Based Profiling Attacks

Unlike non-profiling SCA (e.g. DPA [KJJ99], CPA [BCO04]), which can be done using only one set of traces, profiling attacks need two independent sets of traces. One is the profiling set, used to train the model aiming at leveraging the information maximally. The other is the attacking set, used to mount a practical attack. For the profiling set, the attacker knows all the information needed including corresponding secret key and the

[1] In practical implementations of deep learning algorithms, Categorical Cross Entropy is used when the labels are encoded in one-hot form, which is a typical case. However, it is equivalent to Cross Entropy in a more complex form so we only use Equation 1 for simplicity.

public variable. In other words, the attacker is accessible to a profiling set $\mathcal{S}_P$ including a number of (namely N_P) traces together with additional information

$$\mathcal{S}_P : \{(p_i, k^*, \mathbf{x}_i)|1 \leq i \leq N_P\},$$

where k^* denotes the secret key from key space $\mathcal{K}$ and p_i denotes the i-th known public variable with $\mathbf{x}_i$ as the corresponding physical leakage (e.g. power or EM traces)[2]. Note that $\mathbf{x}_1, \ldots, \mathbf{x}_{N_P}$ are i.i.d. since they are independent observations.

Before the attacker starts the process of profiling, a transformation is usually conducted to fit the data to deep learning tasks

$$\mathcal{S}_P : \{(l_i, \mathbf{x}_i)|1 \leq i \leq N_P\},$$

where $l_i = f(v_i)$ denotes the label for $\mathbf{x}_i$ corresponding to i-th sensitive intermediate $v_i = G(k^*, p_i)$. Then the profiling phase can be done by training a model with traces as inputs and transformed intermediates as labels. The model with its parameters, denoted by M_θ will be updated during training process. After the training, the final prediction model $\mathrm{M}_\theta : \mathbf{x} \to \mathbf{y}$ will be generated, where $\mathbf{y}$ can be viewed as probability for the labels.

After the profiling phase, the attacker can start the attacking phase to mount a practical attack. For an attacking set $\mathcal{S}_A = \{(p_i, \mathbf{x}_i)|1 \leq i \leq N_A\}$ having more than one trace from the same but unknown secret key, maximum likelihood method is the optimal way to mount a successful attack. A log likelihood score will be assigned to every key candidate $k \in \mathcal{K}$ for later key discrimination process, i.e.,

$$d_{\mathcal{S}_A}[k] = \frac{1}{N_A} \sum_{i=1}^{N_A} \log_2(\mathbf{y}_i[f \circ G(p_i, k)]). \tag{4}$$

Here we use log likelihood for convenience.

The key candidate with the highest score is assumed to be the right key. Actually, training a deep learning model with CE loss is exactly maximizing the log likelihood score for the right key k^* with the profiling set.

3 A Novel Side Channel Evaluation Metric

In side channel scenario, GE and SR are fair metrics for the comparison between side channel attacks and countermeasures. They can give information on how the attack is going, how many traces are needed for a successful attack and to what extent we can reduce the uncertainty of the right key. GE and SR are qualified for both profiling and non-profiling attacks, but they are time consuming since they need to be estimated on a set of experiments. Subsection 3.1 elaborates GE/SR and their estimations. In Subsection 3.2, we propose a novel deep learning-based side channel evaluation metric CER and also give its estimation. Subsection 3.3 gives some discussions about CER.

3.1 Guessing Entropy and Success Rate

At the attacking phase of profiling attacks we mentioned in Subsection 2.3, a log likelihood score $d_{\mathcal{S}_A}[k]$ is computed for every key hypothesis. The attacker can next calculate a guess vector $\mathbf{g} = [g[1], g[2], \ldots, g[|\mathcal{K}|]]$ with the different key candidates sorted in descending

[2]Here we can assume all the profiling traces are generated with the same secret key without loss of generality since the secret key k^* will not be used directly as training labels. Typical public available datasets also generate profiling traces with same secret key and varying known public varible p. For a profiling trace generated from a different key k with a training label $l = f \circ G(k, p)$, it can be viewed as a trace generated with the same key k^* and another public variable p' having the same label $l = f \circ G(k^*, p')$.

order according to their scores $d_{\mathcal{S}_A}[k]$: the most likely candidate being $g[1]$. The secret key k^* is successfully recovered when $g[1] = k^*$ or equivalently $d_{\mathcal{S}_A}[k^*] > d_{\mathcal{S}_A}[k]$ for any $k \neq k^*$.

Guessing Entropy For any attacking set $\mathcal{S}_A$ drawn uniformly from the joint distribution $\Pr[\mathbf{X}, P]$ with N_A traces under secret key k^*, we define an attacker $\mathbf{A}_{ge}$, who will output the position of the right key candidate k^* in the guessing vector $\mathbf{g}$ as

$$\mathbf{A}_{ge}(\mathcal{S}_A) = i, \text{if } \mathbf{g}[i] = k^*. \tag{5}$$

The guessing entropy is defined as the expected value of $\mathbf{A}_{ge}(\mathcal{S}_A)$

$$\mathbf{GE} = \mathop{\mathbb{E}}_{\mathcal{S}_A}[\mathbf{A}_{ge}(\mathcal{S}_A)]. \tag{6}$$

Success Rate For any attacking set $\mathcal{S}_A$ drawn uniformly from the joint distribution $\Pr[\mathbf{X}, P]$ with N_A traces under secret key k^*, we define an attacker $\mathbf{A}_{sr}$, who will output the result whether the secret key k^* is successfully recovered as

$$\mathbf{A}_{sr}(\mathcal{S}_A) = \begin{cases} 1 & \text{if } \mathbf{g}[1] = k^* \\ 0 & \text{otherwise} \end{cases}. \tag{7}$$

Then the success rate is defined as the probability of $\mathbf{A}_{sr}(\mathcal{S}_A)$ outputting 1

$$\mathbf{SR} = \Pr[\mathbf{A}_{sr}(\mathcal{S}_A) = 1]. \tag{8}$$

The value of **GE** can be estimated thanks to **LLN**. Consider a set of experiments with N independent attacking set $\mathcal{S}_A^1, \mathcal{S}_A^2, \ldots, \mathcal{S}_A^N$. Then by **LLN**, the sample average of $\mathbf{A}_{ge}(\mathcal{S}_A)$, denoted by $\widehat{\mathbf{GE}}$, converges in probability to **GE**, i.e.,

$$\widehat{\mathbf{GE}} = \frac{\mathbf{A}_{ge}(\mathcal{S}_A^1) + \mathbf{A}_{ge}(\mathcal{S}_A^2) + \cdots + \mathbf{A}_{ge}(\mathcal{S}_A^N)}{N} \xrightarrow{p} \mathbf{GE} \text{ for } N \to \infty, \tag{9}$$

which means $\widehat{\mathbf{GE}}$ is a suitable estimation of **GE**. Also, the estimation error can be bounded by Chebyshev' inequality.

For success rate **SR**, the probability can be estimated using the frequency of $\mathbf{A}_{sr}(\mathcal{S}_A)$ outputting 1 with a set of independent experiments similar to previous one, i.e.,

$$\widehat{\mathbf{SR}} = \frac{\sum_{i=1}^{N} \mathbf{1}_{\mathbf{A}_{sr}(\mathcal{S}_A^i)=1}}{N}. \tag{10}$$

3.2 Cross Entropy Ratio Metric and Its Estimation

GE and SR can not be embedded in deep learning algorithms smoothly due to high computational complexity and therefore we will propose a novel metric for SCA directly from deep learning's perspective.

Recall the fact that training deep learning models using CE loss is exactly maximizing the log likelihood score for the right key k^* in Equation 4. Let $\Pr[\mathbf{X}, L^k]$ be the joint distribution and the $\mathrm{M}_\theta(\mathbf{X})[L^k]$ be the predicted distribution under key hypothesis k with the labels generated by $L^k = f \circ G(P, k)$, and then their Cross Entropy denoted as $\mathrm{CE}(k)$ is as follows:

$$\begin{aligned} \mathrm{CE}(k) &= H(\Pr[\mathbf{X}, L^k], \mathrm{M}_\theta(\mathbf{X})[L^k]) \\ &= \mathop{\mathbb{E}}_{\mathbf{X}, L^k}[-\log_2 \mathrm{M}_\theta(\mathbf{X})[L^k]] \\ &= \mathop{\mathbb{E}}_{\mathbf{X}, P}[-\log_2 \mathrm{M}_\theta(\mathbf{X})[f \circ G(P, k)]]. \end{aligned} \tag{11}$$

During the training process, $\mathrm{CE}(k^*)$ is supposed to be minimized if CE loss function is used. However, the efficiency of the attack will also be influenced by $\mathrm{CE}(k)$ for $k \neq k^*$ and the attacking result might become misleading when $\mathrm{CE}(k)$ for $k \neq k^*$ increases. Hereafter, we define a novel metric Cross Entropy Ratio for deep learning model performance evaluation in side channel scenario as follows:

$$\mathbf{CER} = \frac{\mathrm{CE}(k^*)}{\underset{k\neq k^*}{\mathbb{E}}[\mathrm{CE}(k)]}. \tag{12}$$

As accurately calculating CER requires perfect knowledge about the joint distribution $\Pr[\mathbf{X}, L^k]$ for all possible k, which is not accessible to the attacker, we need to give it an appropriate estimation like GE and SR. To estimate CER, we first introduce a lemma.

Lemma 1. *Let $\{A_n\}_{n=1}^{\infty}$, $\{B_n\}_{n=1}^{\infty}$ be two sequences of random variables. A_n converges to a constant α in probability. $B_n \geq 1$ for $n \in \mathbb{Z}^+$, and B_n converges to a constant $\beta \geq 1$ in probability. Then A_n/B_n converges to α/β in probability.*

Proof. For $n \in \mathbb{Z}^+$, we have

$$\frac{A_n}{B_n} = \frac{\alpha}{B_n} + \frac{A_n - \alpha}{B_n}$$

Thus,

$$\begin{aligned}\left|\frac{A_n}{B_n} - \frac{\alpha}{\beta}\right| &= \left|\frac{\alpha}{B_n} - \frac{\alpha}{\beta} + \frac{A_n-\alpha}{B_n}\right| \\ &\leq \left|\frac{\alpha}{B_n} - \frac{\alpha}{\beta}\right| + \left|\frac{A_n-\alpha}{B_n}\right| \\ &\leq \left|\frac{\alpha}{B_n} - \frac{\alpha}{\beta}\right| + |A_n - \alpha|.\end{aligned}$$

Since $|\alpha/B_n| \leq |\alpha|$ for $n \in \mathbb{Z}^+$, α/B_n converges to α/β in probability, and A_n converges to α in probability, we have A_n/B_n converges to α/β in probability. □

Remark 1. The case $B_n \geq c > 0$ can be treated the same by defining $\tilde{B}_n = B_n/c$.

Remark 2. $B_n > 1$ for $n \in \mathbb{Z}^+$ can be relaxed to $B_n > 1$ for $n > N, n \in \mathbb{Z}^+$ by modifying the condition in the proof "For $n \in \mathbb{Z}^+$" to "For $n > N, n \in \mathbb{Z}^+$".

Given the training set $\mathcal{S}_P$ and the deep learning model M_θ, the following property holds according to Equation 3:

$$H_{N_P}(\mathcal{S}_P, \mathrm{M}_\theta) \xrightarrow{p} H(\Pr[\mathbf{X}, L^{k^*}], \mathrm{M}_\theta(\mathbf{X})[L^{k^*}])) = \mathrm{CE}(k^*) \text{ for } n \to \infty \tag{13}$$

Also, since G is a cryptographic primitive, the output of $G(P,k)$ can be regarded as pseudo-random, thus for $k \neq k^*$, $L^k = f \circ G(P,k) = f(U)$, where U is randomly drawn from the image set of G, i.e. $\mathcal{V}$. Then for $k \neq k^*$ and k independent of M_θ, we have:

$$\begin{aligned}\underset{k\neq k^*}{\mathbb{E}}[\mathrm{CE}(k)] &= \underset{k\neq k^*}{\mathbb{E}}\ \underset{\mathbf{X},P}{\mathbb{E}}[-\log_2 \mathrm{M}_\theta(\mathbf{X})[f\circ G(P,k)]] \\ &= \underset{k\neq k^*}{\mathbb{E}}\ \underset{\mathbf{X},U}{\mathbb{E}}[-\log_2 \mathrm{M}_\theta(\mathbf{X})[f(U)]] \\ &= \underset{\mathbf{X},U}{\mathbb{E}}[-\log_2 \mathrm{M}_\theta(\mathbf{X})[f(U)]].\end{aligned} \tag{14}$$

Consequently, we can give an estimation of $\underset{k\neq k^*}{\mathbb{E}}[\mathrm{CE}(k)]$.

Let $\mathcal{S}_P^r = \{(\mathbf{X_1}, f(U_1)), \ldots, (\mathbf{X}_{N_P}, f(U_{N_P}))\}$ be a randomized training set where $U_1, U_2, \ldots, U_{N_P}$ are randomly drawn from $\mathcal{V}$, then by applying **LLN** we have:

$$\frac{1}{N}\sum_{i=1}^{N} -\log_2 \mathrm{M}_\theta(\mathbf{X}_i)[f(U_i)] \xrightarrow{p} \underset{\mathbf{X},U}{\mathbb{E}}[-\log_2 \mathrm{M}_\theta(\mathbf{X})[f(U)]] = \underset{k\neq k^*}{\mathbb{E}}[\mathrm{CE}(k)], \tag{15}$$

Equation 15 can be rewritten using the estimation of Cross Entropy as follows:

$$H_{N_P}(\mathcal{S}_p^r, \mathrm{M}_\theta) = \frac{1}{N_P}\sum_{i=1}^{N} -\log_2 \mathrm{M}_\theta(\mathbf{X_i})[f(U_i)] \xrightarrow{p} \mathop{\mathbb{E}}_{k\neq k^*}[\mathrm{CE}(k)] \tag{16}$$

Combing Lemma 1 with Equation 13 and Equation 16, and supposing $\mathop{\mathbb{E}}_{k\neq k^*}[\mathrm{CE}(k)] \geq c > 0$ for sufficiently large n, we can give an estimation of CER:

$$\widehat{\mathbf{CER}} = \frac{H_{N_P}(\mathcal{S}_P, \mathrm{M}_\theta)}{H_{N_P}(\mathcal{S}_p^r, \mathrm{M}_\theta)}, \tag{17}$$

with the convergence property holds

$$\widehat{\mathbf{CER}} \xrightarrow{p} \mathbf{CER} \text{ for } N_P \longrightarrow \infty. \tag{18}$$

3.3 Discussions

Indeed the CER metric can effectively evaluate the performance of a deep learning model for profiling attacks since: (1) A small $\mathrm{CE}(k^*)$ indicates the model is close to the true joint distribution of the traces and labels; (2) A big $\mathop{\mathbb{E}}_{k\neq k^*}[\mathrm{CE}(k)]$ indicates the model is less possible to mix up the right key with the wrong keys. Consequently, a smaller CER implies the model learns more information with less ambiguity, which means the model has a good performance for an attack.

By the way, CER metric in Equation 12 can also be defined in another way by substituting Cross Entropy to Kullback-Leibler (K-L) divergence. However, we don't do it because in this way we just add the same constant to both the numerator and denominator while the metric has equivalent functionality.

CER has great compatibility with deep learning. There is no obstacle estimating it since most deep learning algorithms provide Application Programming Interface (API) to compute the estimation of Cross Entropy. Given the profiling set $\mathcal{S}_P$ and the model M_θ, $H_{N_P}(\mathcal{S}_P, \mathrm{M}_\theta)$, the numerator of $\widehat{\mathbf{CER}}$, can be easily and quickly calculated by calling corresponding API. To compute $H_{N_P}(\mathcal{S}_p^r, \mathrm{M}_\theta)$, a little trick can be used. First, we can generate $\mathcal{S}_P^r$ by shuffling the labels in $\mathcal{S}_P$ while keeping the traces unchanged and then it can be calculated just like the numerator. To improve the accuracy, we can iterate the process on N shuffled sets $\mathcal{S}_P^{r_1}, \mathcal{S}_P^{r_2}, \ldots, \mathcal{S}_P^{r_N}$, and use the mean value of $H_{N_P}(\mathcal{S}_p^{r_i}, \mathrm{M}_\theta)$ as the estimation of $H_{N_P}(\mathcal{S}_p^r, \mathrm{M}_\theta)$. Besides, since estimating CER does not require to mount practical attacks like GE/SR, it has low computational complexity and can be easily embedded in deep learning algorithms.

If the estimation of CER suffers from overfitting, it is necessary to separate a validation set $\mathcal{S}_v$ independent of the profiling set to do the estimation. If no overfitting problem appears, the better choice would be using the profiling set, since it usually contains more traces and a validation set is not always available. Actually, CER metric suffers less from overfitting than natural deep learning metrics as it is derived from profiling attacks, whose purpose is not maximizing the classification accuracy. Thus it is good to estimate CER using a validation set while still acceptable to use the training set if a validation set is not available.

4 Relationship Between CER and GE/SR

We will give the theoretic relation between CER and GE/SR with concrete proof and discuss their underlying connection in Subsection 4.1. And then we validate it in Subsection 4.2 with experiments on a public dataset. Subsection 4.3 gives a comparison between CER and deep learning metric accuracy for SCA with imbalanced data.

4.1 CER Reflects GE/SR

CER and GE/SR are all good metrics used to evaluate the performance of SCA in different occasions. CER can be easily estimated in deep learning scenario and does not require to mount practical attacks, while GE/SR are suitable for any kinds of SCA at a high computational cost. Actually CER has a close relationship with GE/SR in deep learning-based profiling attacks, as stated in the following propositions.

Proposition 1 (CER and GE)**.** *For any attacking set $\mathcal{S}_A$ drawn uniformly from the joint distribution* $\Pr[\mathbf{X}, P]$ *under key k^* with N_A traces, and an attacker $\mathbf{A}_{ge}$,* $\mathbf{CER} < 1$ *implies the following results:*

(i) $\mathbf{A}_{ge}(\mathcal{S}_A) = 1$ *for $N_A \to \infty$, and*

(ii) $\mathbf{GE} = 1$ *for $N_A \to \infty$.*

Proof. According to the definition of CER in Equation 12, $\mathbf{CER} < 1$ is equivalent to $\mathrm{CE}(k^*) < \underset{k \neq k^*}{\mathbb{E}}[\mathrm{CE}(k)]$. Besides, by Equation 11 and Equation 14, if $k \neq k^*$ we have $\underset{k \neq k^*}{\mathbb{E}}[\mathrm{CE}(k)] = \mathrm{CE}(k)$ since:

$$\begin{aligned} \mathrm{CE}(k) &= \underset{\mathbf{X},P}{\mathbb{E}}[-\log_2 \mathrm{M}_\theta(\mathbf{X})[f \circ G(P,k)]] \\ &= \underset{\mathbf{X},U}{\mathbb{E}}[-\log_2 \mathrm{M}_\theta(\mathbf{X})[f(U)]] \\ &= \underset{k \neq k^*}{\mathbb{E}}[\mathrm{CE}(k)], \end{aligned} \tag{19}$$

where U is randomly drawn from $\mathcal{V}$. Then we have $\mathrm{CE}(k^*) < \mathrm{CE}(k)$ for $k \neq k^*$.

Additionally, for the log likelihood score in Equation 4 we have:

$$\begin{aligned} \underset{\mathcal{S}_A}{\mathbb{E}}[-d_{\mathcal{S}_A}[k]] &= \mathbb{E}[-\frac{1}{N_A}\sum_{i=1}^{N_A} \log(\mathbf{y}_i[f \circ G(p_i,k)])] \\ &= \frac{1}{N_A}\sum_{i=1}^{N_A} \mathbb{E}[-\log(\mathbf{y}_i[f \circ G(p_i,k)])] \\ &= \frac{1}{N_A}\sum_{i=1}^{N_A} \underset{\mathbf{x}_i,p_i \sim \Pr[\mathbf{X},P]}{\mathbb{E}}[-\log_2 \mathrm{M}_\theta(\mathbf{x}_i)[f \circ G(p_i,k)]] \\ &= \frac{1}{N_A}\sum_{i=1}^{N_A} \mathrm{CE}(k) \\ &= \mathrm{CE}(k), \end{aligned} \tag{20}$$

which implies the opposite of the log likelihood score i.e., $-d_{\mathcal{S}_A}[k]$ converges to $\mathrm{CE}(k)$ in probability for $N_A \to \infty$ by **LLN**. As $\mathrm{CE}(k^*) < \mathrm{CE}(k)$ for $k \neq k^*$, we have $d_{\mathcal{S}_A}[k^*] > d_{\mathcal{S}_A}[k]$ for $k \neq k^*$ and $N_A \to \infty$. Hence the guess vector output by the attacker $\mathbf{A}_{ge}$ will be $\mathbf{g} = [k^*, g[2], \ldots, g[|\mathcal{K}|]]$, i.e. $\mathbf{A}_{ge}(\mathcal{S}_A) = 1$ for $N_A \to \infty$. Since $\mathbf{GE} = \underset{\mathcal{S}_A}{\mathbb{E}}[\mathbf{A}(\mathcal{S}_A)]$, we also have $\mathbf{GE} = 1$ for $N_A \to \infty$. □

Proposition 2 (CER and SR)**.** *For any attacking set $\mathcal{S}_A$ drawn uniformly from the joint distribution* $\Pr[\mathbf{X}, P]$ *under key k^* with N_A traces, and an attacker $\mathbf{A}_{sr}$,* $\mathbf{CER} < 1$ *implies the following results:*

(i) $\mathbf{A}_{sr}(\mathcal{S}_A) = 1$ *for $N_A \to \infty$, and*

(ii) $\mathbf{SR} = 1$ *for* $N_A \to \infty$.

Proof. Similar to Proposition 1, we know $\mathbf{CER} < 1$ implies $\mathrm{CE}(k^*) < \mathrm{CE}(k)$ for $k \neq k^*$, and thus $d_{\mathcal{S}_A}[k^*] > d_{\mathcal{S}_A}[k]$ for $k \neq k^*$ and $N_A \to \infty$. The guess vector output by the attacker $\mathbf{A}_{sr}$ will be $\mathbf{g} = [k^*, g[2], \ldots, g[|\mathcal{K}|]]$, i.e. $\mathbf{A}_{sr}(\mathcal{S}_A) = 1$ for $N_A \to \infty$. Since $\mathbf{SR} = \Pr[\mathbf{A}_{sr}(\mathcal{S}_A) = 1]$, we also have $\mathbf{SR} = 1$ for $N_A \to \infty$. □

Proposition 1 and Proposition 2 states the fact that the attacker will surely disclose the secret key once CER < 1 if enough traces are available.

From the propositions, we know $\mathrm{CE}(k)$ is the expected value of $-d_{\mathcal{S}_A}[k]$ and $\underset{k \neq k^*}{\mathbb{E}}[\mathrm{CE}(k)] = \mathrm{CE}(k)$. To further explore the relation between CER and GE/SR, we rewrite CER as

$$\begin{aligned}\mathbf{CER} &= \frac{\mathrm{CE}(k^*)}{\mathrm{CE}(k)} \\ &= \frac{\underset{\mathcal{S}_A}{\mathbb{E}}[-d_{\mathcal{S}_A}[k^*]]}{\underset{\mathcal{S}_A}{\mathbb{E}}[-d_{\mathcal{S}_A}[k]]},\end{aligned} \tag{21}$$

where $k \neq k^*$. Let $\mathbf{CER} = \gamma$, for $k \neq k^*$ we have

$$\begin{aligned}\underset{\mathcal{S}_A}{\mathbb{E}}[d_{\mathcal{S}_A}[k^*]] - \underset{\mathcal{S}_A}{\mathbb{E}}[d_{\mathcal{S}_A}[k]] &= \underset{\mathcal{S}_A}{\mathbb{E}}[-d_{\mathcal{S}_A}[k]] - \underset{\mathcal{S}_A}{\mathbb{E}}[-d_{\mathcal{S}_A}[k^*]] \\ &= (1-\gamma)\underset{\mathcal{S}_A}{\mathbb{E}}[-d_{\mathcal{S}_A}[k]]\end{aligned} \tag{22}$$

According to Equation 4, $\underset{\mathcal{S}_A}{\mathbb{E}}[-d_{\mathcal{S}_A}[k]] \geq 0$. Actually, we can assume $\underset{\mathcal{S}_A}{\mathbb{E}}[-d_{\mathcal{S}_A}[k]] > 0$ when $k \neq k^*$ without loss of generality. Equation 22 indicates that on average sense, the score of the right key will be greater than the score of the wrong keys when CER < 1 and the difference of the scores gets bigger when CER becomes smaller. Hopefully with a smaller CER, the attacker might need less traces to mount a successful attack, or with the same amount of traces he might get a lower GE and a higher SR. However this haven't been proved yet but intuitively it is the case since random variables will be around the expected value in a small range with high probability according to Chebyshev' inequality.

Besides, Equation 22 also implies that on average sense the right key and the wrong keys will be more distinguishable with a big $\underset{\mathcal{S}_A}{\mathbb{E}}[-d_{\mathcal{S}_A}[k]]$, i.e., a big $\underset{k \neq k^*}{\mathbb{E}}[\mathrm{CE}(k)]$, which in turn verifies the definition of CER is reasonable.

4.2 Experimental Validation

To validate the relation between CER and GE/SR, we conduct some experiments on a public dataset.

4.2.1 ASCAD Dataset

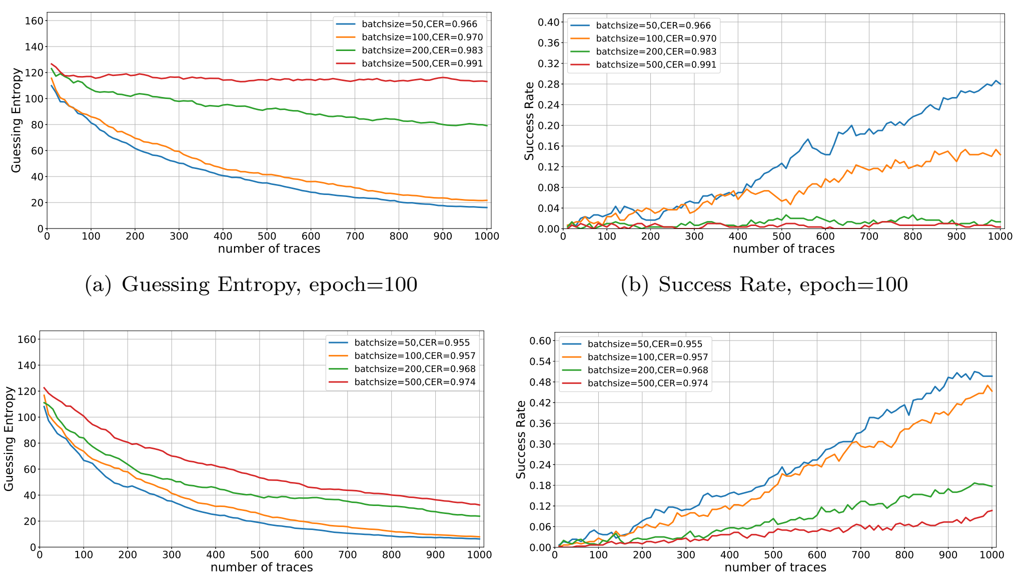
ASCAD dataset is introduced in [PSB+18] by Prouff et al. aiming at providing a standard dataset specifically for SCA. ASCAD dataset targets a software protected AES implementation running over an 8-bit AVR architecture, namely ATMega8515[3]. In total, 60000 power traces are provided in which 50000 traces are divided into the profiling set and the remaining traces are used as the attacking set. Each raw trace contains 100000 time samples and for convenience, 700 time samples concerning with the operations of the third masked Sbox in the first round are selected. These traces have no first order leakage unless the masks are leaked, which is verified in the original paper. Two sets of desynchronized traces are also provided with the jitter window ranging maximally to 50 and 100 time samples correspondingly.

[3]This dataset is publicly available at https://github.com/ANSSI-FR/ASCAD.

4.2.2 Settings and Results

We use synchronized traces with hamming weight and identity labels as the training data for the MLP models in the profiling phase. The structures of the MLP models used in our experiments are similar to the ones proposed in [PSB+18] and we vary the epoch and batch size to explore whether CER reflects GE/SR under different conditions.

The MLP models are trained using a NVIDIA TITAN Xp graphic card on ubuntu with driver version 418.39 and CUDA version 10.1. Tensorflow 2.0 [AAB+15] with GPU support is used to accelerate the training process. To estimate GE and SR after the training, we iterate the attacking process 300 times and each time 1000 traces are randomly chosen from the attacking set to mount a maximum likelihood attack. Meanwhile, CER is estimated in every iteration and the mean value is used to improve accuracy.



(a) Guessing Entropy, epoch=100

(b) Success Rate, epoch=100

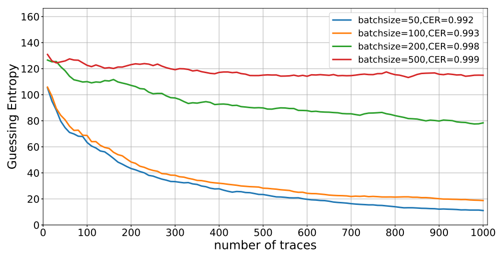
(c) Guessing Entropy, epoch=200

(d) Success Rate, epoch=200

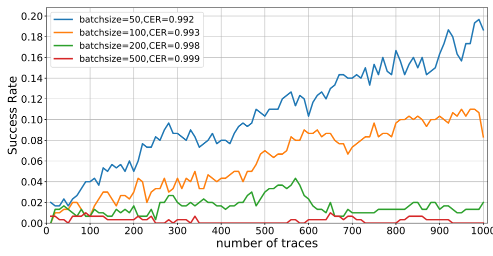
Figure 3: Relation between CER and GE/SR for hamming weight labels with different epochs and batch sizes.

First, for models using hamming weight labels, i.e. for imbalanced labels, the results are encouraging. As shown in Figure 3, with different epochs and batch sizes, CER is closely related to GE/SR. A small CER is usually a indication for a small GE and a high SR. For model using identity labels, corresponding to balanced data, the results are similar. As Figure 4 shows, CER also reflects GE/SR.

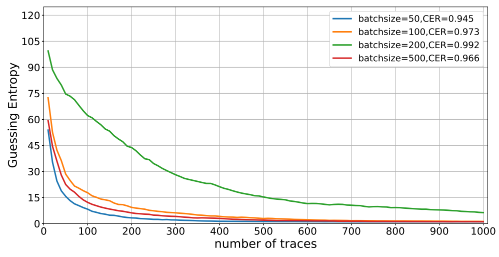
Putting them together, ranking CER in ascending order with different epochs and batch sizes, as shown in Figure 5, it is clear that CER is consistent with GE/SR. In most of time, a smaller CER lead to a lower GE and a higher SR, indicating a better attacking result. Sometimes the consistency might also be violated. For example, in 5(a), GE for CER = 0.968 is higher than GE for CER = 0.970 with 1000 attacking traces. This slight inconsistency is inevitable as CER is irrelevant with the number of attacking traces and thus less accurate than GE/SR. However, it is acceptable since CER generally show good performance in foreseeing the attacking results and has great potential thanks to its great compatibility with deep learning algorithms. What' more, estimating CER does not require to mount piratical attacks like GE/SR and thus has reasonable computational complexity.



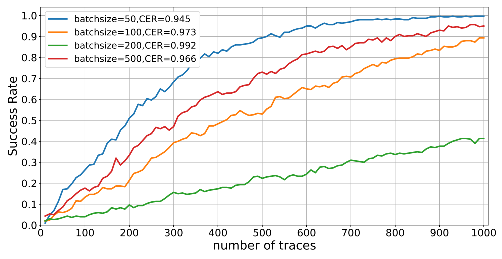
(a) Guessing Entropy, epoch=100



(b) Success Rate, epoch=100

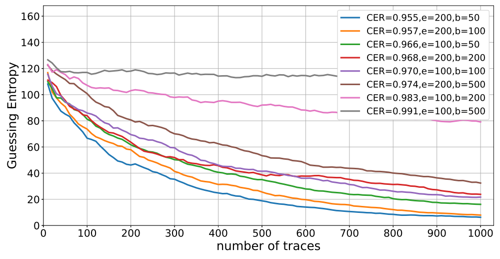


(c) Guessing Entropy, epoch=200

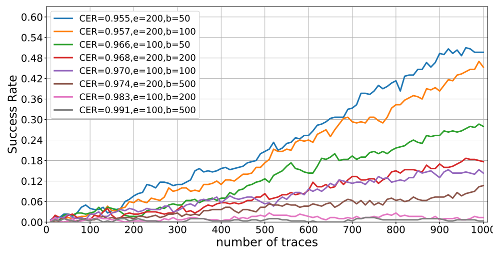


(d) Success Rate, epoch=200

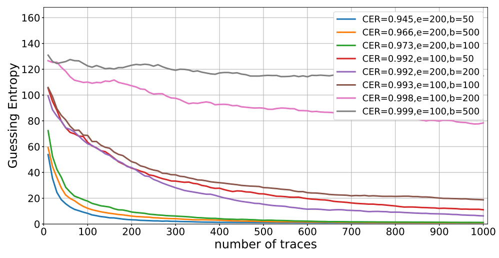
Figure 4: Relation between CER and GE/SR for identity (Sbox Out) labels with different epochs and batch sizes.



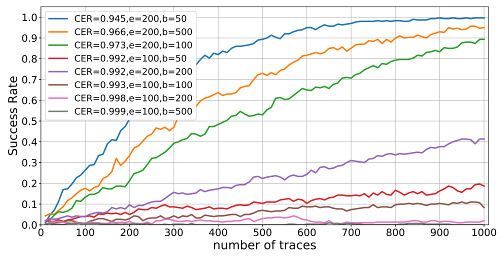
(a) Guessing Entropy, HW



(b) Success Rate, HW



(c) Guessing Entropy, Sbox Out



(d) Success Rate, Sbox Out

Figure 5: Relation between CER and GE/SR for hamming weight (HW) and identity (Sbox Out) labels with different epochs and batch sizes.

4.3 CER v.s. Accuracy

To further investigate the effectiveness of CER with imbalanced data and show why deep learning metrics such as accuracy are not suitable in some occasions, we also conduct several experiments with different levels of imbalanced training data. For simplicity, we use the least significant bit of the output of Sbox as our labels (i.e. two classes, $label = 1$ and $label = 0$), and vary the proposition of $label = 1$ from 50% to 90%. The total number of the training traces is fixed to 24000, selected from original ASCAD dataset according to the level of imbalance and we use MLP models similar to previous ones.

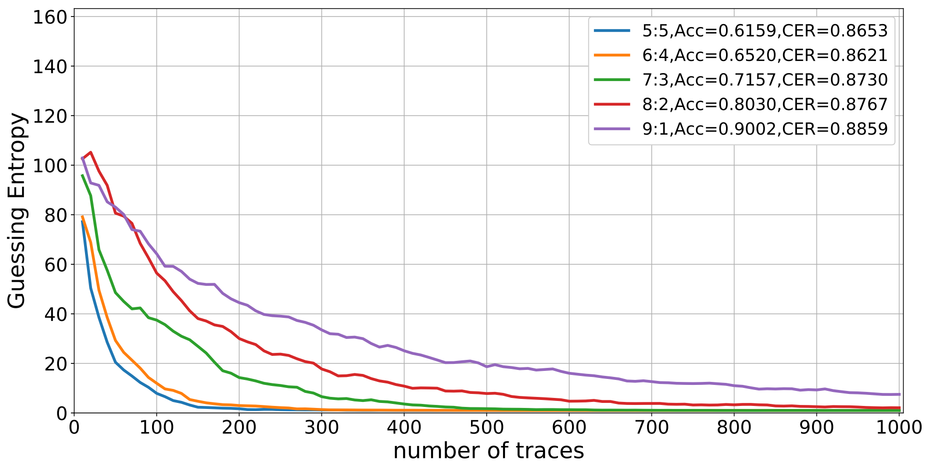


Figure 6: Accuracy (Acc) and CER with imbalanced data in different levels. The ratio of $label = 1$ to $label = 0$ ranges from 5:5 to 9:1.

The results are shown in Figure 6. When the training data becomes imbalanced, the training accuracy increases while the attacking result turns out to be worse. To the contrary, CER stays sound with different levels of imbalanced data and is still a good indicator for the results of the attack. The deceptive performance of accuracy metric won't change even if we separate a validation set since this set is also imbalanced[4].

Estimating CER is easy and has similar computational complexity to the calculation of accuracy. In our experiments, time cost for computing CER and accuracy is close, $7.02ms \pm 150\mu s$ and $6.36ms \pm 496\mu s$ correspondingly, both less than GE/SR which cost much more time (typically more than half an hour). Here we use only one iteration to estimate the denominator of CER. Considering more iterations needs more time but it is still in milliseconds level, which is fast enough in practice. Hence CER metric is qualified for evaluating the performance of deep learning models with a low computational complexity and works for both balanced and imbalanced data.

5 Extended Application to Imbalanced Data

In this section, we adapt CER metric to an effective deep learning loss function specifically for SCA with imbalanced data. Subsection 5.1 gives the definition of CER loss function and Subsection 5.2 shows some experimental results on several public datasets.

[4]The validation set can be made balanced but at the cost of large number of traces belonging to the minority class, which will make the training set even more imbalanced. Normally the validation traces are randomly selected and separated from the training set and thus they have the same distribution with the training traces

5.1 CER Loss Function

Now that CER can be a good metric indicating the attacking performance of deep learning models and has a low computational complexity, it is possible to adapt it to a loss function for the training of deep neural networks. Similar to Cross Entropy loss function, CER loss function can be defined as follows:

$$Loss_{CER} = \frac{H_{N_P}(\mathcal{S}_P, \mathrm{M}_\theta)}{\frac{1}{N}\sum_{i=1}^{N} H_{N_P}(\mathcal{S}_P^{r_i}, \mathrm{M}_\theta)}, \tag{23}$$

where $\mathcal{S}_P^{r_1}, \ldots, \mathcal{S}_P^{r_N}$ are shuffled sets generated from $\mathcal{S}_P$ by shuffling the labels randomly, where N is a constant defined by user to determine how many shuffled sets are needed.

Actually, Equation 23 is noting but an estimation of CER on the training set. We already know from Proposition 1 and Proposition 2 that CER < 1 implies the attack will surely succeed if sufficient traces are available and we validate a smaller CER indicates a better attacking result with experiments. According to Equation 21, CER loss minimization is trying to maximize the log likelihood score of the right key and minimize the scores of wrong keys at the same time, which can make the right key and the wrong keys more distinguishable.

Unlike loss functions designed to maximize the classification accuracy, CER loss function is for deep learning in side channel domain. As shown in Subsection 4.3, CER suffers less than accuracy when training data is imbalanced. Hence, using CER loss for training can mitigate the problems posed by imbalanced data. Comparing to balancing techniques introduced in [PHJ+19] to deal with the imbalanced problems, CER loss function has more potential since it works for implementations with/without masks and does not require to modify the training data set. CER loss function can also be applied to balanced cases. However, we focus on imbalanced cases in this paper as we want to solve the imbalanced problems.

Like traditional loss functions, CER loss function may also be influenced by overfitting. How to solve this problem under side channel scenario is important and needs to be deeply explored though beyond this paper. Adding a proper regulation term in the loss function or using cross validation might help. We leave this part to future work and use CER loss function directly in our later experiments.

5.2 Experiments with Imbalanced Data

To show the advantages of CER loss for imbalanced data, we conduct some experiments with hamming weight/distance labels on several public datasets apart from ASCAD. All the results are encouraging, indicating using CER loss function in deep learning algorithms is beneficial to the performance of SCA for imbalanced data.

5.2.1 ASCAD Dataset

For ASCAD dataset we use the hamming weights of the outputs of a masked Sbox as the imbalanced labels. We experiment with the synchronized trace set and two desynchronized trace sets while we use different deep learning models. For synchronized cases, we still use MLP models with six layers and for desynchronized cases, CNN models are used since they can deal with jitters. The structure of the CNN models is VGG-like proposed in [PSB+18] and we set epoch to 75 and batch size to 200 in our training[5].

[5]All MLP and CNN models used later on have similar structures.

5.2.2 DPA Contest v4 Dataset

DPA Contest v4 dataset provides measurements of a masked AES implementation in software on an ATMega-163 smart card[6]. The masked implementation utilizes a lightweight mask scheme AES-256 RSM which is found to have first order leakage lately [MGH14]. Here we assume the masks are known and thus our attacks fall to first order case. In total, we retrieved 10000 traces, each containing 1000 features around the first Sbox part of the algorithm execution. 9000 traces are randomly chosen for the training of MLP models and the rest of traces are preserved for later attack.

5.2.3 Dataset with Random Delay Countermeasure (AES_RD)

Introduced in [CK09], AES_RD dataset is a collection of power traces from a software implementation of AES with random delays[7]. By adding random delays to the normal operations, the traces captured are misaligned and the Signal to Noise Ratio (SNR) is quite low. As a result, MLP models are not powerful enough to mount a successful attack even with large number of traces. Thus we carry out our attacks using CNN models to deal with random delays. In our experiments on this dataset, 40000 traces are fed to CNN models for training and 10000 traces are preserved for later attack.

5.2.4 Unprotected AES-128 on FPGA (AES_HD)

Different from datasets collecting traces from software implementations as described above, AES_HD provides measurements of an unprotected hardware implementation of AES-128 on FPGA[8]. Totally 100000 EM traces were measured and 1250 features are included in each trace. We mount our attacks on the last round of AES with MLP models and use hamming distance leakage of register writing as our imbalanced labels, i.e.,

$$L = HW(\mathrm{Sbox}^{-1}[C_i \oplus k^*] \oplus C_j)$$

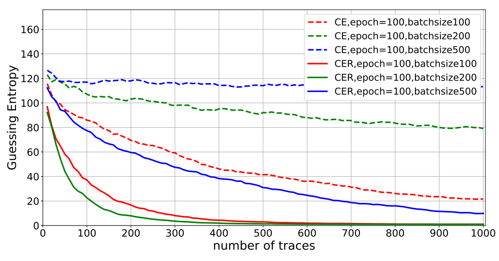
where HW denote hamming weight function, where C_i and C_j are two cipertext bytes. Same as [KPH+19], we choose $i = 12$ resulting in $j = 8$. In our experiments on this dataset, 50000 traces are separated for the training of MLP models and 25000 traces are used to mount our attacks.

5.2.5 Experimental Results

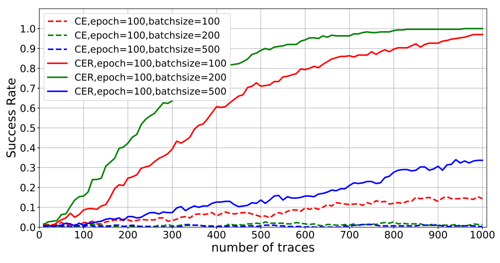
For ASCAD dataset, the attacking results are given in Figure 7, corresponding to the synchronized cases, and Figure 8, corresponding to the desynchronized cases. As the figures show, in both cases, CER loss function performs better than CE loss function. Firstly, in synchronized cases, attacking results for MLP models with CER loss function show excellent improvements and suffer less when batch size is large. For example, using MLP models with CER loss function we can attain about 100% SR with 1000 traces when epoch = 100 and batch size = 200, or epoch = 200 and batch size = 100 or 200, while we can get no more than 50% SR with CE loss function. In all cases, models using CER loss function always have better performance on GE/SR. Secondly, for desynchronized traces using CNN models for the training, the attacking results for CER loss function also outperform the results for CE loss function. All the experimental results on ASCAD dataset show CER loss performs better in both MLP and CNN models with synchronized and desynchronized traces.

For DPA Contest v4 dataset, the secret key can be easily retrieved with about a hundred traces since the masks are assumed to be known. Apparently, as shown in Figure 9, using

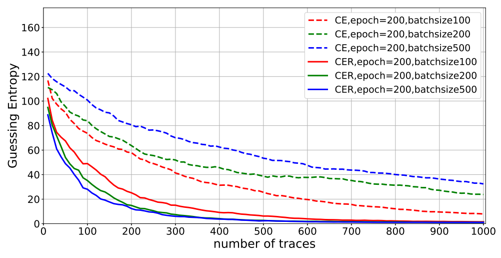
[6]This dataset is publicly available at dpacontest.org.
[7]This dataset is publicly available at https://github.com/ikizhvatov/randomdelays-traces.
[8]This dataset is publicly available at https://github.com/AESHD/AES_HD_Dataset.



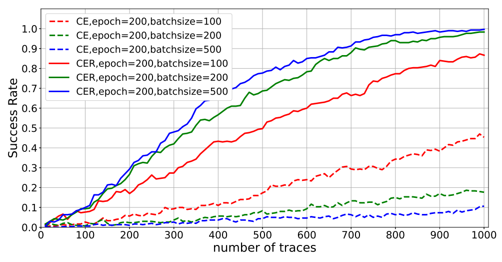
(a) Guessing Entropy, epoch=100



(b) Success Rate, epoch=100

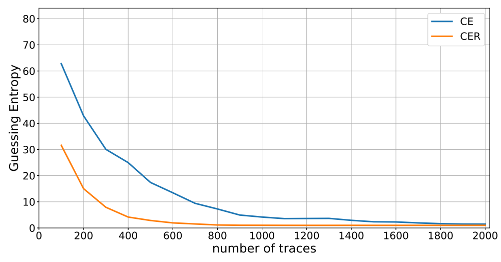


(c) Guessing Entropy, epoch=200

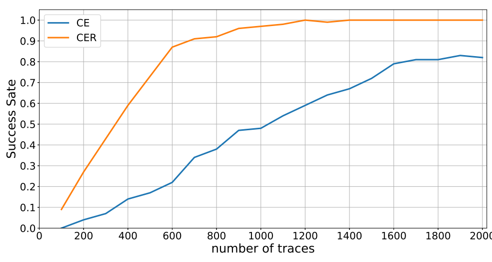


(d) Success Rate, epoch=200

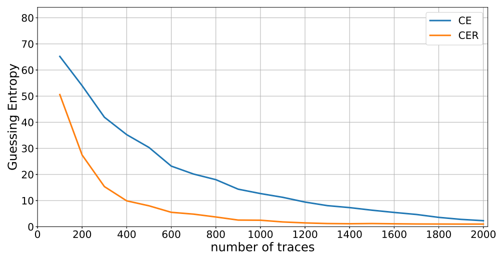
Figure 7: ASCAD: attacking results of MLP models for CER and CE loss functions.



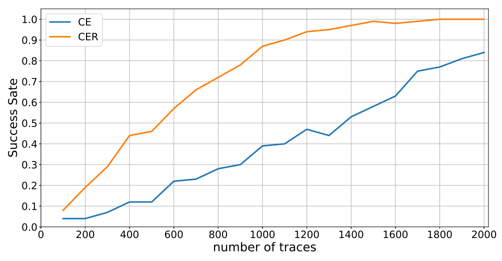
(a) Guessing Entropy, desync=50



(b) Success Rate, desync=50

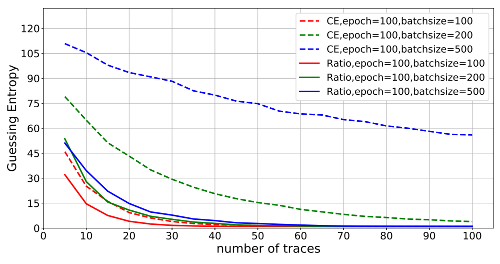


(c) Guessing Entropy, desync=100

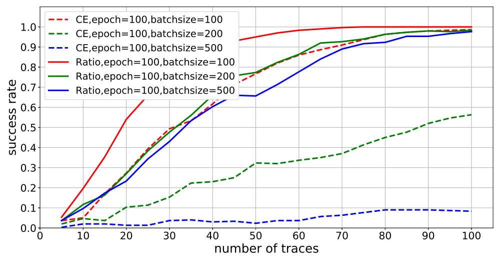


(d) Success Rate, desync=100

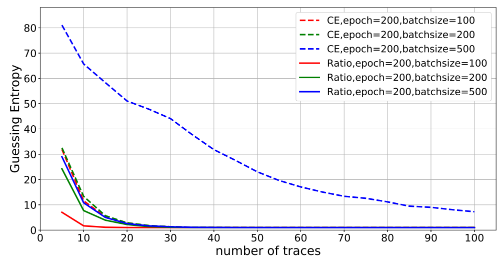
Figure 8: ASCAD: attacking results of CNN models for CER and CE loss functions with desynchronized traces.



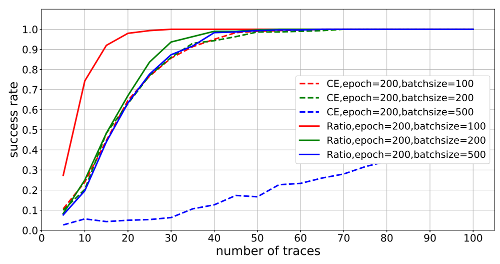
(a) Guessing Entropy, epoch=100



(b) Success Rate, epoch=100

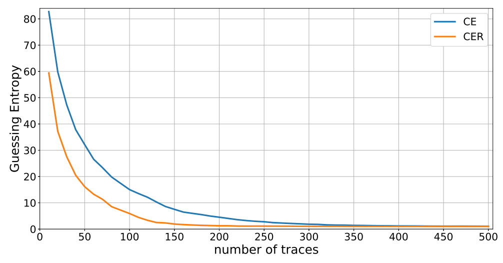


(c) Guessing Entropy, epoch=200

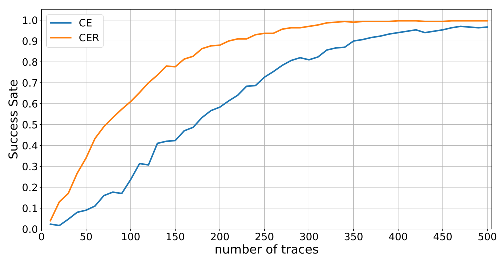


(d) Success Rate, epoch=200

Figure 9: DPA Contest v4: attacking results of MLP models for CER and CE loss functions.

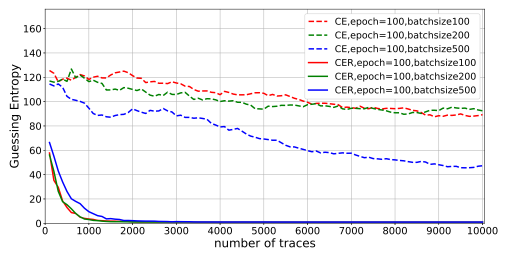


(a) Guessing Entropy, random delay

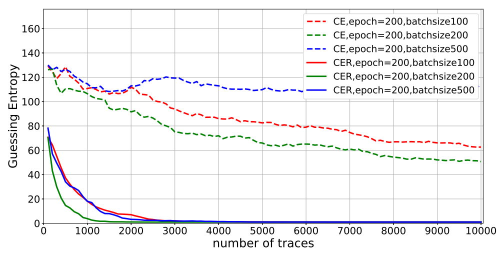


(b) Success Rate, random delay

Figure 10: AES_RD: attacking results of CNN models for CER and CE loss functions with random delay countermeasures.



(a) Guessing Entropy, epoch=100



(b) Guessing Entrpoy, epoch=200

Figure 11: AES_HD: attacking results of MLP models for CER and CE loss functions.

CER loss also brings advantages. For example, models using CER loss function give the best result reaching above 70% SR with 10 traces and 100% with 30 traces, while models using CE loss function can only reach no more than 30% SR with 10 traces and need more than 50 traces to get a 100% SR. Besides, CER loss function show better efficiency than CE loss under the same condition in all experiment settings.

Figure 10 shows the results for AES_RD, which are similar to the cases of ASCAD dataset with desynchronized traces. CER loss also helps to improve the attacking results and is again validated to be an effective loss function for CNN models when dealing with traces with random delays.

For AES_HD dataset, results given in Figure 11 show significant improvements. The secret key can be recovered almost surely with about 3000 traces for models using CER loss function, though traditional MLP models with CE loss function are not powerful enough to mount a successful attack with even 10000 traces. The reason for the huge improvements might be that the low SNR of hardware implementations brings obstacles minimizing CE loss in MLP models while minimizing CER loss is kept easy. Consequently, CER loss function is also suitable and efficient for deep learning-based SCA on hardware implementations.

To sum up, our experiments show CER loss function is more efficient than CE loss function with imbalanced data for software/hardware implementations, MLP/CNN models, synchronized/desynchronized traces, and both first order and high order cases. Hence we properly solve the problems of imbalanced data in side channel scenario.

6 Conclusions

In this paper, we propose a novel evaluation metric CER for deep learning-based SCA. CER can be used to evaluate the performance of deep learning models for profiling attacks and it works stably with imbalanced data. Also CER can be easily embedded in deep learning algorithms and has a low computational complexity since it does not require practical attacks like GE/SR. Besides, CER is closely related to GE/SR both in theory and our experiments. A small CER is generally an indication for a low GE and a high SR. Thus CER can be a complement of side channel metrics that fits to deep learning algorithms and we properly solve the open question in [PHJ+19].

Moreover, we explore the possibility of adapting CER metric to an explainable and efficient loss function for side channel attacks with imbalanced data. We conduct numerical experiments on several datasets to investigate the effectiveness and efficiency of our proposed CER loss function. Compared to commonly used CE loss function, our experimental results show CER loss function exhibits sound effectiveness and better efficiency when dealing with hamming weight/distance labels. Thus CER loss function can be an effective alternative of loss functions in deep learning-based SCA for imbalanced data.

Acknowledgments

The authors would like to thank Information Science Laboratory Center of USTC for the hardware/software services. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This work was supported by National Natural Science Foundation of China (Nos. 61972370 and 61632013), Fundamental Research Funds for Central Universities in China (No. WK3480000007) and Anhui Initiative in Quantum Information Technologies under Grant AHY150400.

References

[AAB+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[APSQ06] Cédric Archambeau, Eric Peeters, François-Xavier Standaert, and Jean-Jacques Quisquater. Template attacks in principal subspaces. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2006.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.

[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.

[CDP15] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Enhancing dimensionality reduction methods for side-channel attacks. In Naofumi Homma and Marcel Medwed, editors, *Smart Card Research and Advanced Applications - 14th International Conference, CARDIS 2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers*, volume 9514 of *Lecture Notes in Computer Science*, pages 15–33. Springer, 2015.

[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.

[CK13] Omar Choudary and Markus G. Kuhn. Efficient template attacks. In Aurélien Francillon and Pankaj Rohatgi, editors, *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*, volume 8419 of *Lecture Notes in Computer Science*, pages 253–270. Springer, 2013.

[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.

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

[HCE+17] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, and Kevin W. Wilson. CNN architectures for large-scale audio classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017*, pages 131–135. IEEE, 2017.

[HG09] Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Trans. Knowl. Data Eng.*, 21(9):1263–1284, 2009.

[ISW03] Yuval Ishai, Amit Sahai, and David A. Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

[Kim14] Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[Koc96] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[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. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(3):148–179, 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.

[MDP19] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. A comprehensive study of deep learning for side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1):348–375, Nov. 2019.

[MGH14] Amir Moradi, Sylvain Guilley, and Annelie Heuser. Detecting hidden leakages. In Ioana Boureanu, Philippe Owesarski, and Serge Vaudenay, editors, *Applied Cryptography and Network Security - 12th International Conference, ACNS 2014, Lausanne, Switzerland, June 10-13, 2014. Proceedings*, volume 8479 of *Lecture Notes in Computer Science*, pages 324–342. Springer, 2014.

[Mor16] Ajinkya More. Survey of resampling techniques for improving classification performance in unbalanced datasets. *CoRR*, abs/1608.06048, 2016.

[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.

[PSB$^+$18] Emmanuel Prouff, Rémi Strullu, Ryad Benadjila, Eleonora Cagli, and Cécile Dumas. Study of deep learning techniques for side-channel analysis and introduction to ASCAD database. *IACR Cryptology ePrint Archive*, 2018:53, 2018.

[RO04] Christian Rechberger and Elisabeth Oswald. Practical template attacks. In Chae Hoon Lim and Moti Yung, editors, *Information Security Applications, 5th International Workshop, WISA 2004, Jeju Island, Korea, August 23-25, 2004, Revised Selected Papers*, volume 3325 of *Lecture Notes in Computer Science*, pages 440–456. Springer, 2004.

[RQL19] Pieter Robyns, Peter Quax, and Wim Lamotte. Improving CEMA using correlation optimization. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(1):1–24, 2019.

[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.

[RW17] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation*, 29(9):2352–2449, 2017.

[SA08] François-Xavier Standaert and Cédric Archambeau. Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, volume 5154 of *Lecture Notes in Computer Science*, pages 411–425. Springer, 2008.

[SKWW07] Yanmin Sun, Mohamed S. Kamel, Andrew K. C. Wong, and Yang Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378, 2007.

[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.

[SWK09] Yanmin Sun, Andrew K. C. Wong, and Mohamed S. Kamel. Classification of imbalanced data: a review. *IJPRAI*, 23(4):687–719, 2009.

[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.

[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.