A Guessing Entropy-based Framework for Deep Learning-Assisted Side-Channel Analysis

Ziyue Zhang, A. Adam Ding, and Yunsi Fei Member, IEEE

Abstract—Recently deep-learning (DL) techniques have been widely adopted in side-channel power analysis. A DL-assisted SCA generally consists of two phases: a deep neural network (DNN) training phase and a follow-on attack phase using the trained DNN. However, currently the two phases are not well aligned, as there is no conclusion on what metric used in the training can result in the most effective attack in the second phase. When traditional loss functions such as negative log-likelihood (NLL) are used in training a DNN, the trained model does not yield optimal follow-on attack. Recently some information theoretical SCA leakage metrics are proposed, either as the validation metric to stop the DNN training with traditional loss functions, or as both the validation metric and the training loss function. None of those proposed metrics, however, directly measures the SCA effectiveness. We propose to conduct DNN training directly with a common SCA effectiveness metric, Guessing Entropy (GE). We overcome the prior practical difficulty of using GE in DNN training by utilizing the GEEA estimation algorithm introduced in CHES 2020. We show that using GEEA as either the validation metric or the loss function produces DNN models that lead to much more effective follow-on attacks. Our work consolidates the DL-assisted SCA framework with a consistent metric, which shows great potential to be adopted as the universal SCA-oriented DNN training framework.

Index Terms—Side-channel Analysis, Deep learning, Guessing Entropy, Evaluation metric.

I. Introduction

Over the past several years, deep learning (DL) has been adopted to assist power side-channel attacks (SCAs). DL-assisted SCAs can break devices protected by countermeasures such as masking and hiding [MPP16], [CDP17]. Secure implementations of different ciphers are all cracked by DL-assisted SCAs [CCC+19], [WPB19]. The common DL-assisted SCAs generally consist of two phases. In the training phase, side-channel leakage traces from a device with a known key are used to train a deep neural network (DNN) classifier whose output labels reflect key-dependent intermediate values in the cipher. In the attack phase, traces from the device with an unknown key are fed as inputs to the pre-trained DNN to predict the labels, and the key is derived as the one among all key candidates that most agrees with the predicted result.

However, how to train the DNN so as to yield an optimal model for SCA is still an unresolved issue. DNN is traditionally trained to minimize a loss function, such as negative log-likelihood (NLL), mean squared error (MSE), and cross-entropy (CE). Experimental results demonstrate that the optimal DNN trained under such a loss function often does not deliver the best SCA as measured by SCA effectiveness metrics such as guessing entropy (GE) and success rate (SR) [CDP17], [PHJ+19].

In deep learning practices, to prevent model overfitting, a common technique is to stop the learning via monitoring a validation metric. The data set is split into a training set S_{Tr} and a validation set S_{Val}. While the learning algorithm optimizes the training loss on S_{Tr}, a validation metric is evaluated on S_{Val} to check overfitting. Typically, the validation metric is the same loss function. We call such procedurehomogeneous train-ing. The procedure using a validation metric different from the loss function is called heterogeneous training. Heterogeneous training is often used when the desired optimization quantity is not differentiable and therefore not viable for numerical optimization in model training. For example, for a classifier, the common optimization metrics include accuracy, precision and recall, all dependent on true positive rates and true negative rates etc. These metrics can not be used directly in loss functions but can be used as the validation metric to monitor learning based on traditional loss functions, such as NLL, MES and CE.

Neither optimizing for these classification metrics with heterogeneous training nor optimizing traditional loss functions with homogeneous training will lead to optimal DL-assisted SCA [PHJ+19]. This phenomenon has prompted many proposals on new SCA related information-theoretic metrics for DNN learning. For heterogeneous training, metrics such as empirical mutual information and $d_{train;val}$ [PBP20], [RZC+20] have been proposed as the validation metric. For homogeneous training, Cross Entropy Ratio (CER) [ZZN+20] and Ranking loss(RkL) [ZBD+20] are proposed for both the loss function and the validation metric. While these proposals resulted in SCA effectiveness improvement over traditional DNN training in experiments on some example datasets, none ensures the optimality of the follow-on SCA.

A universally accepted optimal training procedure for DL-assisted SCAs is still missing. As in common deep learning practices, to achieve optimal SCA effectiveness, conceptually we should use a SCA effectiveness metric such as GE either as the validation metric in heterogeneous training or as the loss function in homogeneous training. Using the average rank of the correct key, which is empirical GE, in heterogeneous DNN training has been suggested before [PHJ+19], [WVdHG+20]. However, using such an empirical GE as the validation metric suffers due to the instability and inaccuracy of the empirical GE estimation. Since the empirical GE is not differentiable, it

This work was supported in part by the National Science Foundation under Grant No. CNS-2212010, SaTC-1929300, and IUCRC-1916762.

Ziyue Zhang, Aidong Adam Ding and Yunsi Fei are with Northeastern University, Boston, MA 02115, USA. (emails: zhang.ziyue@northeastern.edu; a.ding@northeastern.edu; y.fei@northeastern.edu)

cannot be used as the training loss function directly in homogeneous training. In this paper, we explore a practical procedure of DNN training to directly optimize the corresponding SCA's GE by using the recently proposed Guessing Entropy Estimation Algorithm (GEEA) estimator in CHES2020 [ZDF20]. Through an adjustment to batch training, we ensure efficient and accurate practical GE estimation for first time during the training process. We demonstrate that GEEA can be used in either heterogeneous training or homogeneous training. It finds the DNN for the follow-on attack which minimizes the SCA metric GE.

The rest of the paper is organized as follows. In Section II, we first describe background information on DNN structures and the framework of DL-assisted SCAs. In Section III, we elaborate our GEEA-based DNN training framework for SCAs. We implement the proposed GEEA-based training framework and compare with existing training frameworks over public datasets in Sections IV and V. We first present results of the heterogeneous training framework and demonstrate the advantage of using the GEEA function as the validation metric. We then use the GEEA function directly as the loss function and demonstrate the advantage of our proposed GEEA-based homogeneous training framework over other existing homogeneous training frameworks. Finally we conclude the paper in Section VI with some discussions.

II. Background

In this section, we provide the background of common DNN structures used in SCA and the procedure of profiled DL-assisted SCA.

A. Deep Neural Networks for Side-channel Attacks

A neural network is composed of the first input layer, the last output layer (which outputs predictions), and hidden layers in between. Within each layer, computation neurons perform operations on the input to the layer and a set of trainable parameters. Different types of hidden layers are designed with unique operation so that a neural network can achieve specific functionality with proper combinations of layers. Generally, the architecture of a neutral network is determined by the type and number of its layers as well as the number of neurons contained in each layer. To find the optimized parameters, a neural network has to be trained with a training set S_{Tr}. A loss function that measures the classification error of f over the training set is minimized using an optimizer (e.g., Stochastic Gra-dient Descent [Rob07] [KW52] [BCN18], RMSprop, and Adam [KB17]).

Two model families are most commonly used for SCAs: Multi-Layer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs).

1) MLP: An MLP is efficiently computable, composed of fully-connected layers and activation functions. A fully-connected layer performs an affine transformation $y = w x + b$ of its input x with a weight matrix w and a bias vector b. The activation function is a non-linear function, such as sigmoid, RELU, and SELU function, that will follow the affine transformations and be applied to each coordinate of the feature map. To solve a classification problem, the last fully-connected layer is commonly followed by a special activation function, softmax, $s(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$. The role of s is to normalize the output scores in such a way that they define a probability distribution.

2) CNN: A CNN is composed of two parts: a convolutional part for feature selection and a fully-connected part for classification. The convolutional part is composed by several convolutional blocks, each of which includes a series of one convolutional layer followed by an activation layer and ends with a pooling layer. The fully-connected part has the same architecture as an MLP model.

A convolutional layer performs convolutional operations on the input with multiple convolutional filters. The filter kernel can be configured with some hyperparameters - width and height of the filter, and parameters - weights in the filter are trainable. A pooling layer is a non-linear layer that reduces the dimension of the input while preserves the most relevant information. The hyperparameter is the pooling window size. The window tiles through the input to select elements where the pooling function will be applied onto. Max pooling and Average pooling are the most commonly used pool functions.

B. Deep Learning-Assisted Profiled Attack

The most common type of DL-assisted SCAs is the profiled attack. In a profiled attack, a dataset $S_P = (x_i; z_i)_{1 \le i \le N_P}$ is collected to train the network, where the traces x_i are acquired on the victim device with the key known. The target label $z_i = G(k_c; p_i)$ is the sensitive information leakage, and the select function $G(;)$ is over the key and public variable p_i. Commonly used select functions include Hamming Weight (HW), Hamming Distance (HD), and SBox Output, etc.

In the first training phase, a DNN is trained (with its internal parameters updated) to match the model output $y^{S_P}() = (f(x_i))_{1 \le i \le N_P}$ with the target label $z^{S_P} = (z_i)_{1 \le i \le N_P}$ on the profiled dataset. Then in the attack phase, the attacker collects an attack dataset S_A containing N_A measurement traces of the victim device, with unknown cor-rect key k_c and the known variable $(p_i)_{1 \le i \le N_A}$. For each key candidate $k_g \in K$, the attacker calculates all the labels $z^{S_A}_{k_g} = (z_{i;k_g})_{1 \le i \le N_A} = (G(k_g; p_i))_{1 \le i \le N_A}$ over the attack dataset. The attacker then chooses the key candidate k_g that yields the best matching labels $z^{S_A}_{k_g}$ with the DNN model outputs $y^{S_A}()$ on the attack dataset S_A, measured by a score function $d_{S^A} = d(z_{S^A}; y^{S_A}())$.

In literature, there are two types of score functions used in the attack phase to measure the discrepancy between the DNN model predicted outputs and the assumed labels $z^{S_A}_{k_g}$ given a key candidate.

- NLL Attack: Most existing attacks are of this type. Here the model output $y_i()$ is a probability-like score vector for the input sample x_i, reflecting the likelihood of each possible label class. For the assumed class label $z_{i;k_g}$, denote $y_{i;k_g} = y_i(; z_{i;k_g})$ as the component of $y_i()$ corresponding to the $z_{i;k_g}$-th class, i.e. the likelihood for the $z_{i;k_g}$-th class outputted by the DNN model for the

i-th trace x_i. The key is distinguished using the whole dataset log-likelihood which is an additive score.

$$d_{k_g}^{S_A} = \frac{1}{N_A} \sum_{i=1}^{N_A} \log_2[y_{i;k_g}]. \quad (1)$$

COR Attack: In a COR Attack [RQL19], a trained DNN model outputs a one-dimension encoding y_i which represents a continuous prediction value for the class labels (e.g., HW of a single-byte SBox output), given the i-th input sample x_i. In the follow-on SCA key recovery, the correlation

$$d_{k_g}^{S_A} = (y; z_{k_g}^{S_A}) = \frac{\frac{1}{N_A}\sum_{i=1}^{N_A}(y_i - \bar{y})(z_{i;k_g} - \bar{z}_{k_g})}{[\frac{1}{N_A}\sum_{i=1}^{N_A}(y_i - \bar{y})^2]^{\frac{1}{2}}[\frac{1}{N_A}\sum_{i=1}^{N_A}(z_{i;k_g} - \bar{z}_{k_g})^2]^{\frac{1}{2}}} \quad (2)$$

between the encodings and assumed labels under each candidate $k_g \in \mathcal{K}$ is computed, and the key with the highest correlation is deemed to be the correct key. This attack requires to also minimize the COR loss function during the training.

For one byte-key, the attacker can work with the 256 classes (one for each possible key values) or 9 classes of the HW/HD values. The COR attack can only work with continuous labels thus it requires a correct specification of leakage model such as HW in the 9 classes attack. The NLL attack do not require a continuous label and can work as either the 256 classes attack or the 9 classes attack.

An unresolved problem is what is the best training procedure so that the trained DNN will result in the most effective SCA (for either score function). We discuss the training procedure in the next section.

III. Training Frameworks of DNN in DL-assisted SCAs

Traditionally, in view of the second-phase of NLL/COR attack, the first-phase of training a DNN in DL-assisted SCAs aims to optimize the score function on the profiling dataset. This follows the usual DL training procedure to minimize a loss function $\mathcal{L}$ evaluated on the profiling dataset $\mathcal{S}_P$. For a DNN model f, a loss function $\mathcal{L}$ measures the discrepancy between the model output $y^{\mathcal{S}_P}(\theta)$ and the target label z. The model parameters, θ, are updated iteratively so as to minimize the loss function value $\mathcal{L}(y^{\mathcal{S}_P}(\theta); z^{\mathcal{S}_P})$, with some algorithms such as Gradient Descent algorithm.

Simply training DNN to minimize the loss $\mathcal{L}(y^{\mathcal{S}_P}(\theta); z^{\mathcal{S}_P})$ often leads to overfitting. To avoid overfitting, a common machine learning technique splits the profiling dataset into a training set $\mathcal{S}_{Tr}$ and validation set $\mathcal{S}_{Val}$. While the train-ing algorithm follows the gradient direction for the loss $\mathcal{L}^{Tr}(\theta) = \mathcal{L}(y^{\mathcal{S}_{Tr}}(\theta); z^{\mathcal{S}_{Tr}})$ on the training dataset $\mathcal{S}_{Tr}$ to update the parameters, the evaluation metric $\mathcal{M}^{Val}(\theta) = \mathcal{M}(y^{\mathcal{S}_{Val}}(\theta); z^{\mathcal{S}_{Val}})$ on the validation dataset $\mathcal{S}_{Val}$ is moni-tored to decide when the training process should be terminated. The DNN parameters, θ, are trained to converge to values that optimize $\mathcal{M}^{Val}$ through certain stopping rule rather than values that minimize $\mathcal{L}^{Tr}$. That is, the current DNN training practice can be formulated as:

Scheme 1: (Current Training Practice) The attacker monitors the evaluation metric $\mathcal{M}^{Val}(\theta)$ on the validation dataset $\mathcal{S}_{Val}$ and aims to select the model $f_{\hat{\theta}}$ such that

$$\hat{\theta} = \arg\operatorname{opt}_{\theta} \mathcal{M}^{Val}(\theta)$$

while updating $\hat{\theta}$ using gradient of $\mathcal{L}^{Tr}(\theta)$.

Ideally the loss function should be consistent with the validation metric. However, for the classification problems which most machine learning methods aimed to solve, validation metrics (e.g., classification accuracy) often are not differentiable and therefore cannot be directly used as a loss function, and instead some surrogate loss function has to be used. We classify the training framework into two types: Homogeneous Training Framework where the loss function on the training dataset is the same as the evaluation metric on the validation dataset, i.e. $\mathcal{M}^{Val}(\theta) = \mathcal{L}^{Val}(\theta)$; Heterogeneous Training Framework where the loss function differs from the validation metric.

When applying deep learning for SCAs, a research question is what training loss function and what validation evaluation metric should be used in DNN training for SCAs. Earliest DL-assisted SCAs optimized the attack score function under the true key k_c. Since most existing attacks use the likelihood for distinguishing scores, this becomes homogeneous training of DNN with the widely used machine learning loss function Negative log-likelihood (NLL), i.e., $d_{k_c}^{S_A}$ in (1). [MDP20] theoretically relates minimizing the NLL to maximizing the perceived leakage information.

However, the performance of follow-on SCAs involves the comparison of the scores under the true key k_c versus other key candidates k_g, and is usually measured by Gaussian Entropy (GE) or Success Rate (SR). Maximizing the NLL under true key k_c does not always lead to optimal GE or optimal SR of follow-on SCAs, and the deficiency is particularly obvious when there are imbalances among the classes labels [PHJ+19]. The root-cause of the mismatch is that NLL, as the training loss function and the validation evaluation, does not directly relate to GE or SR. A natural way of curing this mismatch is to focus the DNN training to optimize GE or SR. However, there has not been a practical way to do this up to now. Previously, the GE is estimated empirically: averaging ranks of the correct key k_c from SCA over multiple independent subsets partitioned from the validation dataset. Using this empirical GE as the validation was attempted in [WVdHG+20]. However, the empirical GE is not reliable nor efficient for DNN training to accurately optimize the GE. We will discuss in detail the issues about its performance in the later section. The empirical estimation of the SR is similarly slow and inaccurate for DNN training. Others proposed to use empirical quantities indirectly related to GE or SR: GEBVD [vdVP19], Mutual Information [PBP20] and $d_{train;val}$ [RZC+20] as the validation metric in the heterogeneous training framework. For homogeneous training, loss functions such as Cross Entropy Ratio (CER) [ZZN+20] and Ranking loss (RkL) [ZBD+20] were similarly proposed as related metrics. The RkL is shown as related to a bound of SR so that minimizing RkL can approximately maximize SR. However, no consensus has been

reached on what are the most appropriate validation metrics and loss functions to use in DNN-assisted SCA.

We propose to conduct the DNN training with a specific goal of optimizing the GE of the follow-on SCA, and show that this is achievable practically using GEEA estimator.

A. GE minimization for DNN-Assisted SCA

The DNN training should be cognizant of the follow-on SCA, since the goal of training is to discover a good DNN model suitable for key distinguishing by the SCA. A common performance metric, Guessing entropy (GE), is the expected value of the true key ranked by attackers, which reflects the average computing cost for a SCA.

Definition 1: (Guessing Entropy) Given a randomly collected attack dataset S_A from the victim device with size $|S_A| = q$, the attack DNN model f gives a score $d^{S_A}_{k_g}$ for each key candidate $k_g \in \mathcal{K}$. The key candidates are ranked according to the scores, and Guessing Entropy, $GE_q(f)$, the expected rank of the correct key k_c, can be computed as follows:

$$GE_q(f) = \mathbb{E}_{S_A : |S_A| = q}[\mathrm{rank}(k_c | S_A; f)] \tag{3a}$$

$$= 1 + \mathbb{E}_{S_A : |S_A| = q}\Big[\sum_{k_g \in \mathcal{K} = k_c} \mathbb{1}_{d^{S_A}_{k_g} > d^{S_A}_{k_c}}\Big] \tag{3b}$$

where $\mathrm{rank}(k_c | S_A; f) = 1 + \sum_{k_g \in \mathcal{K} = k_c} \mathbb{1}_{d^{S_A}_{k_g} > d^{S_A}_{k_c}}$ is one plus the number of wrong key guesses that have a higher score than the correct key.

Using GE as the attack performance measure, we can formulate the goal of the DNN training for SCA as:

Scheme 2: (Attack Optimization) Training the DNN model f is to find the parameter θ such that, for a specified attack dataset of size q

$$\hat{\theta} = \arg\min_{\theta} GE_q(f).$$

Considering the different training frameworks, the attack optimization can be achieved either by using an GE estimator as the validation evaluation metric in the heterogeneous training framework, or using the GE estimator as both the loss function and the validation evaluation metric in the homogeneous training framework.

Empirically, GE can be estimated by the average rank of the true key over N independent attack datasets $(S_{A;i})_{1 \le i \le N}$ each of which is at the same size q. That is the empirical GE:

$$\overline{GE}_q(f) = \frac{1}{N}\sum_{i=1}^{N} \mathrm{rank}(k_c | S_{A;i}; f)] \tag{4a}$$

$$= 1 + \frac{1}{N}\sum_{i=1}^{N}\Big[\sum_{k_g \in \mathcal{K} = k_c} \mathbb{1}_{d^{S_{A;i}}_{k_g} > d^{S_{A;i}}_{k_c}}\Big] \tag{4b}$$

The empirical GE $\overline{GE}_q$, also call Average Rank, has been commonly used to evaluate the SCA effectiveness. Using the $\overline{GE}_q$ as a validation metric in the heterogeneous training has been suggested before [PHJ+19], [WVdHG+20]. However, as we will show in Section IV-B, the empirical GE $\overline{GE}_q$ is very inaccurate. Thus the heterogeneous DNN training using empirical GE $\overline{GE}_q$ is unreliable. The calculation of empirical GE is also inefficient [ZDF20].

Instead, we propose a practical implement of DNN train-ing using the GEEA estimator [ZDF20], $\widehat{GE}_q(\theta)$, given in Equation (6). Within the heterogeneous training framework, we monitor the $\widehat{GE}^{Val}_q(\theta)$ over the validation dataset while the DNN model is being updated in training with the gradient descent algorithm for any loss function. This allows to find the optimal DNN model that yields the best $\widehat{GE}^{Val}_q(\theta)$ value. Furthermore, unlike the empirical GE $\overline{GE}_q$, the GEEA estimator $\widehat{GE}_q(\theta)$ is differentiable with respect to θ. Hence the GEEA can also be used in the homogeneous training framework: we can evaluate $\widehat{GE}^{Tr}_q(\theta)$ on the training dataset S_{Tr}, and use its gradient (with respect to θ) to guide the parameter updating.

B. GEEA Estimator

The GEEA estimator is built on the inherent relationship between GE and pairwise success rates, given in the prior work [ZDF20]:

$$GE_q(f) = 1 + \sum_{k_g \in \mathcal{K} = k_c} P_{S_A : |S_A| = q}(d^{S_A}_{k_g} > d^{S_A}_{k_c})].$$

where $P_{S_A : |S_A| = q}(d^{S_A}_{k_g} > d^{S_A}_{k_c})$ is the probability that, over all possible attack datasets S_A at the size of q, a wrong key candidate k_g is mistakenly chosen over the correct key value k_c according to the score function $d^{S_A}_k$. Most of the SCAs use an additive distinguisher, that is, $d^{S_A}_k = \frac{1}{q}\sum_{i=1}^{q} d_{i;k}$ where $d_{i;k}$ is the score of key value k based on the i-th side-channel measurement trace in the dataset S_A. For an additive distinguisher, $\Delta_k = d^{S_A}_{k_g} - d^{S_A}_{k_c}$ asymptotically follows a univariate Gaussian distribution. Given a dataset S of side-channel measurement traces and an attack model f which computes the additive ranking scores, the mean and the standard deviation of this univariate Gaussian distribution can be estimated as

$$\hat{\mu}^{S}_{k_g} = \frac{1}{|S|}(d^{S}_{k_g} - d^{S}_{k_c}) = \frac{1}{|S|}\sum_{i=1}^{|S|}(d_{i;k_g} - d_{i;k_c}),$$
$$\hat{\sigma}^{S}_{k_g} = \sqrt{\frac{1}{|S|}\sum_{i=1}^{|S|}(d_{i;k_g} - d_{i;k_c} - \hat{\mu}^{S}_{k_g})^2}. \tag{5}$$

Using the estimated means and standard deviations, GEEA estimator becomes:

$$\widehat{GE}_q(f; S) = 1 + \sum_{k_g = k_c} \Phi\Big(\frac{\sqrt{q}\,\hat{\mu}^{S}_{k_g}}{\hat{\sigma}^{S}_{k_g}}\Big), \tag{6}$$

where $\Phi()$ denotes the cumulative distribution function (CDF) of the standard Gaussian distribution $\mathcal{N}(0; 1)$.

The formula (6) can be applied on any dataset. As needed in our training framework, we can apply the formula (6) on the training set S_{Tr} and on the validation data set S_{Val}. To evaluate the effectiveness of the trained DNN for SCA, we can also apply it on the attack dataset S_A.

Particularly, we use the GEEA formula on the two types of DNN-assisted attacks described in Section II-B with different score functions:

GEEA estimation in NLL attack: The NLL attack is an additive distinguisher, with its score function $d_{i;k_g} = \log_2(y_{i;k_g})$ given in Equation (1). Therefore, given a DNN model f, the NLL attack GE is estimated using (6) with the mean and the variance computed in (5).

GEEA estimation in COR attack: The COR attack is not an additive distinguisher. However, it is known that the COR attack is asymptotically an additive distinguisher with uniformly distributed public variable p_i. In such a case, $\frac{1}{q}\sum_{i=1}^{q}(z_{i;k_g} - \bar{z}_k^g)^2$ where $z_{i;k_g}$ is the label function converges to a constant, $\mathrm{var}(z)$, for all $k_g \in K$. The expected model output, $\frac{1}{q}\sum_{i=1}^{q}(y_i - \bar{y})^2$, is independent of k_g. Hence the COR attack using the correlation score $(y; z_{k_g})$ as shown in Equation (2) is asymptotically equivalent to an attack using the additive score of $\mathrm{Cov}(y; z_{k_g}) = \frac{1}{q}\sum_{i=1}^{q}(y_i - \bar{y})(z_{i;k_g} - \bar{z}_{k_g})$. Hence the COR attack GE is estimated using (6) with the mean and the variance in (5) computed as:

$$\hat{\mu}^S_{k_g} = \frac{1}{|S|}\sum_{i=1}^{|S|}(d^{cor}_{i;k_g} - d^{cor}_{i;k_c}); \tag{7a}$$

$$\hat{\sigma}^S_{k_g} = \sqrt{\frac{1}{|S|}\sum_{i=1}^{|S|}(d^{cor}_{i;k_g} - d^{cor}_{i;k_c} - \hat{\mu}^S_{k_g})^2} \tag{7b}$$

where $d^{cor}_{i;k} = (y_i - \bar{y})(z_{i;k} - \bar{z}_k)$

C. GEEA Based Implementation of Training Framework

We now discuss two practical issues in the implementation of training frameworks: selecting the size q of attack dataset and the batch size.

Note that the GE_q varies with q value and the attacker has to decide for what q value to optimize the GE. Due to the instability of the empirical GE discussed in detail of Section IV-B, the selection of q value would affect the trained model greatly. However, the GEEA $\widehat{GE}_q()$ is a smooth decreasing function of q, the selection of q does not affect the final convergent state of except for numerical accuracy. $GE_q()$ is a nonlinear bounded function, and the changes would be very small at both ends - when q is very small or q is very large. In the heterogeneous training, to observe the changes in $\widehat{GE}^{Val}_q()$ clearly, a q value needs to be found experimentally so that $\widehat{GE}^{Val}_q()$ is sensitive enough for changes in . Our experience is that the change is indeed sensitive enough to be observed over a wide range of q values. In homogeneous training, the gradient of $\widehat{GE}^{Tr}_q()$ is also used. Choosing different q values do not change the gradient direction but only the magnitude. So changing the q value essentially changes the learning rate. In practice, the value q should be selected together with the learning rate to allow convergence of training.

Another important practical issue involves the adaption to batched training. Nowadays the standard DNN training is conducted in batches: the training dataset is separated into batches (subsets) and the DNN weights are updated based on the gradient of the loss function computed on each batch. An epoch of training is finished when the weight updates iterate through all batches once. The batched training is necessary for DNN training over large data sets due to resources (such as memory) restriction, and also speeds up the convergency of weights.

However, the correctness of the batched training is based on the addictiveness of the loss function while GEEA is not additive. The standard loss function such as the NLL (1) is additive: averaging the loss calculated on each traces over all traces results in the loss of the whole dataset. This ensures that averaging batch loss over all batches equals the loss of the whole data set. DNN training is based on the gradient of the loss function. Additive loss function also insures that averaging batch loss gradients over all batches equals the loss gradient of the whole data set, thus no bias is introduced in the batched training. On the other hand, GEEA is not additive since the averaged quantity is further passed through nonlinear operations in equations (5) and (6), thus more care are needed.

For heterogeneous training, at the end of an epoch, the validation metric can be calculated over the whole data set. Thus using GEEA as a validation metric in batched training is not an issue. However, for homogeneous training, the parameters are updated based on loss gradient calculated on each batch. A bias in the loss gradient may be introduced to the training process when a small batch size is used in evaluating the non-additive GEEA loss $\widehat{GE}_q$. If the bias is big, the training can end up with lower efficiency (more training epochs are required), worse quality end model or even failure to converge.

We study the effect of batch sizes on the bias introduced by comparing $\widehat{GE}_q$ evaluated on the whole data set versus $\widehat{GE}_{q;batch}$, the average of GEEA estimates evaluated on each batch over all batches. As the batch size increases, $\widehat{GE}_{q;batch}$ converges to $\widehat{GE}_q$. However, for small batch size, the bias can be big. It turns out that the bias is smaller for the more effective DNN model (thus smaller GE for the follow-on SCA). As an illustration, we compare $\widehat{GE}_{q;batch}$ to $\widehat{GE}_q$ for three DNN models during different stages in a successful training for $\mathrm{ASCAD}_{\mathrm{Rand}}$ data set, an example described in the later section IV. We saved the three DNN models on the 15-th (early stage), 35-th (middle stage) and 50-th (late stage) epoch during the training phase. For each model, we then calculate the $\widehat{GE}_{q;batch}$ for various batch sizes where q equals to 20.

Figure 1 plots $\widehat{GE}_{q;batch}$ for the three models versus various batch sizes. The dotted line shows the $\widehat{GE}_q$ calculated on the whole data set. The bias is the largest for the least predictive model at the early stage (15-th epoch). The bias, as expected, decreases when batch size increases. Since a small batch size introduces very large bias, it can cause serious degradation of training process. We showed the validation GE during the training progress of this example using different batch sizes in Figure 2.

From Figure 2, a small batch size of 10 leads to the failure of training due to the large bias it introduced. A batch size of 50 does lead to a successful training in this case, but requires much more epoches than the training with batch size 200. With batch size 200, the validation GE reaches its minimum at the 7-

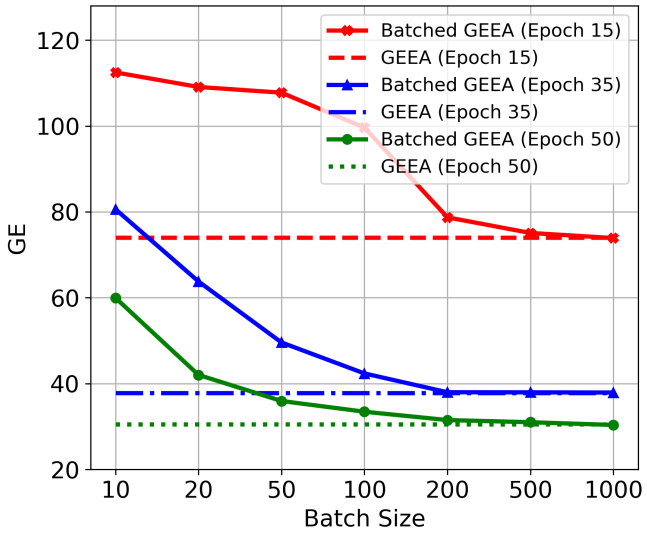


Fig. 1: Batched GEEA versus GEEA on whole training dataset of $\text{ASCAD}_{\text{Rand}}$

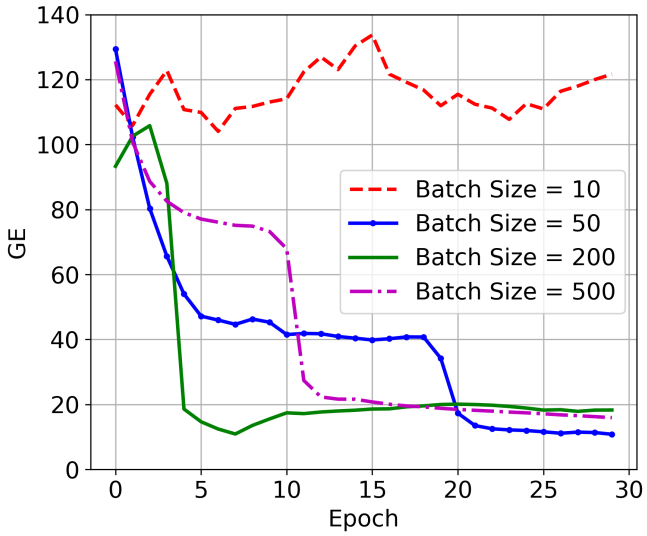


Fig. 2: Training phase on $\text{ASCAD}_{\text{Rand}}$ with different batch size

th epoch, the training will stop according to the usual stopping rule since the rebound in validation GE indicating overfitting after the 8-th epoch. When using an even larger batch size of 500, a similar quality DNN model can be reached after more than 20 epoches. Although a larger batch size ensures less bias thus the correctness of training, the advantage of batch training is lost when the batch size becomes too big.

To see directly the effect of batch size on the loss gradient, the quantity used in parameters update, we show the relative difference $\frac{\|G_{full} - G_{batch}\|_1}{\|G_{full}\|_1}$ during training in Figure 3. Here $\|\cdot\|_1$ denotes the L_1 norm of the vector. The relative difference between the batched average gradient, $\overline{G}_{batch}$, of $\widehat{GE}_{q;batch}$ and the whole data set gradient, G_{full}, of $\widehat{GE}_q$ is large when the batch size is small. As batch size exceeds 200, the relative gradient differences are greatly reduced and accurate training can be conducted.

In general, we propose to select a batch size of at least 200 traces for the GEEA-based homogeneous training. For

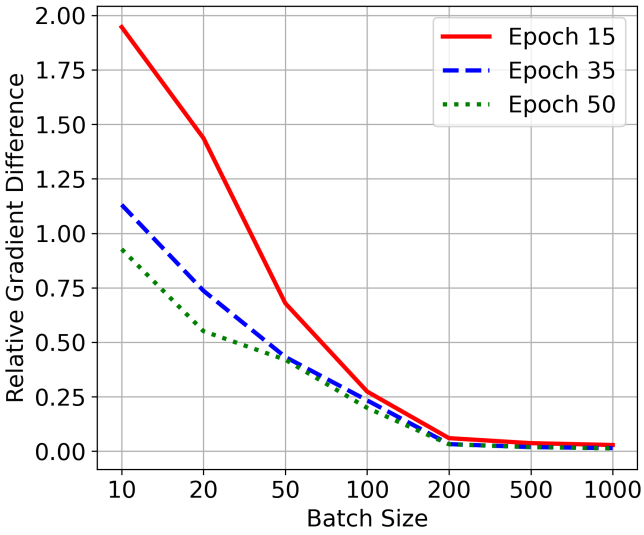


Fig. 3: Relative difference between G_{full} and $\overline{G}_{batch}$ when using GEEA loss function

the heterogeneous training, validation $\widehat{GE}_q^{Val}()$ is calculated on the whole validation dataset S_{val} at the end of an epoch.

In the next two sections, we will compare our proposed GEEA-based training framework with other existing training frameworks. In Section IV, we use GEEA as a validation metric and compare it with other heterogeneous training validation metrics. The NLL and CER loss functions are used as the loss functions respectively for 256 classes and 9 classes attacks as they achieve the start-of-art SCA effectiveness of prior literatures in each setting. In the comparison with other homogeneous training frameworks in Section V, we use GEEA as both training loss function and metric function. Table I summarizes the training frameworks that we compared in following experiments.

TABLE I: Experiment summary of (hetero)homo-geneous framework comparison

Attack types	Sbox(256 classes)		HW/HD(9 classes)	
Hetero-geneous Frame-work	Loss function	Metrics	Loss function	Metrics
	NLL	GEEA	CER	GEEA
		EMP-MI		EMP-MI
		$d_{train;val}$		$d_{train;val}$
Homo-geneous Frame-work	Loss & Metrics		Loss & Metrics	
	GEEA		GEEA	
	NLL		NLL	
	CER		CER	
	RkL		RkL	
			COR	

IV. Framework Comparison on Public Datasets

In this section, we illustrate the implementation of heterogeneous training framework using GEEA as a validation metric on several public datasets. We first illustrate why empirical GE $\overline{GE}_q$ and its variant GEBVD [vdVP19] are not suitable validation metrics for heterogeneous training. Compared to the empirical GE $\overline{GE}_q$, the GEEA estimator is much more stable and will be used in the rest of comparison studies. We then compare GEEA versus other validation evaluation metrics

in literature for heterogeneous training. The resulting DNN model under the GEEA validation metric leads to follow-on SCAs with much lower GE.

A. Public Datasets

We use two public open databases of side-channels measurement traces, which both are for AES-128 implementations on different platforms and have been widely used in SCA literature.

- AES_HD[1] dataset is introduced in [PHJ+19] which provides EM measurements of an unprotected implementation of AES-128 on FPGA. A total of 100,000 EM traces were measured and 1,250 features are included in each trace.
- ASCAD database[2] is widely used as a benchmark for side-channel power analysis. It targets a software protected AES implementation on an 8-bit AVR ATMega85153 microcontroller, and the base dataset contains 60,000 power traces in total. Each raw trace contains 100,000 time samples and within which 700 time samples are related to the operations of the third masked SBox in the first round. These traces have no first-order key leakage unless the masks are leaked, as verified in [BPS+20]. We use two version of ASCAD dataset, i.e. ASCAD_{Fix} and ASCAD_{Rand}, which are generated by a fixed key value and random key values respectively. Two sets of desynchronized traces are also provided with the jitter window maximally of 50 and 100 time samples, respectively.

In the following experiments, we split each dataset into three subsets: the training dataset S_{Tr}, validation dataset S_{Val} and the testing dataset S_T. During training, a loss function is evaluated on the training dataset S_{Tr} to guide the parameter updating and a validation metric is evaluated on the validation dataset S_{Val}. The resulting DNN is then used to launch SCA on the testing dataset S_T, with the SCA effectiveness measured by GE evaluated on S_T. Table II shows the partition for the two datasets.

TABLE II: Partition of datasets

dataset	total	S_{Tr}	S_{Val}	S_T
AES_HD	100,000	60,000	20,000	20,000
ASCAD_{Fix}	60,000	35,000	15,000	10,000
ASCAD_{Rand}	60,000	35,000	15,000	10,000

We also conducted experiments on two other open datasets: TeSCASE [3] and AES_RD [4]. The results are qualitatively similar. Here we only report the results on the EM traces of AES_HD dataset and the power traces of the ASCAD datasets.

B. Efficiency and stability of GEEA over Empirical GE

The empirical GE based on average ranks, $\overline{GE}_q$, has been widely used for evaluating the effectiveness of DNN-assisted SCAs. There has been consideration of using $\overline{GE}_q^{Val}$ as a validation metric in the heterogeneous training. However, there are two main issues that make it unsuitable for usage in training validation.

- High computational overhead: For the $\overline{GE}_q^{Val}$ to be accurate, it requires a very large validation dataset that can be partitioned into a large number of independent subsets. When such a large validation dataset is used, the empirical GE requires a significant amount of computational overhead [ZDF20], [PBP20]. According to the result of Table 1 in [ZDF20], to reach similar accuracy, $\overline{GE}_q^{Val}$ needs a computational effort that is at least 10^4 times of the effort needed by GEEA $\widehat{GE}_q^{Val}$.
- Local partition dependency: Aside from the high computational overhead, a serious problem of using empirical GE $\overline{GE}_q^{Val}$ as the validation evaluation metric is that the value of empirical GE is highly dependent on the particular partition of the validation dataset S_{Val} into independent subsets. Thus the training results also vary with which partition is taken during the evaluation. In contrast, as an theoretical estimation, GEEA $\widehat{GE}_q$ does not require any empirical data partition and is a fixed constant given the validation dataset S_{Val}.

To understand how dataset partitioning would influence the empirical GE computation and the stopping criteria, we implement the heterogeneous training with empirical GE(EMP-GE) $\overline{GE}_q^{Val}$ and GEEA $\widehat{GE}_q^{Val}$ as the evaluation metric, respectively, to decide when the training should stop.

In this experiment, we target the AES_HD dataset, and study a DNN model proposed by Lennert Wouters et al. [WAGP20]. The model is composed of a specialized pre-processing filter that conducts horizontal standardization and a follow-on MLP with two fully-connected layers. On the validation dataset, we calculate the GEEA $\widehat{GE}_q^{Val}$ and empirical GE $\overline{GE}_q$ for $q = 200$. The validation dataset is randomly partitioned into 50 independent subsets when computing the empirical GE, and the whole validation dataset is used to profile the Gaussian distribution for GEEA computation. We repeat the random partition 100 times. For each partition into 50 independent subsets, we conduct a heterogeneous training using $\overline{GE}_q$ computed based on the partition as the validation metric. NLL is used as the loss function to drive the training process. Thus we have 100 training each using $\overline{GE}_q$ based on one partition. We observe that they do not stop at the same epochs.

For each data partition, the corresponding empirical GE is computed in each of the 100 training epochs, and the training should stop at the epoch where the corresponding empirical GE (under this data partition) reaches its minimum. Figure 4 plots the histogram of the stopping epochs chosen by empirical GEs under varying data partitions. The distribution is widely spread out, over 38 distinct epoch values, showing that such stopping rule based on the empirical GEs is highly unstable across random data partitions. In contrast, when using GEEA as the validation metric, the training stops at a unique epoch value.

We compare the 38 EMP-GE based end models with our

[1] https://github.com/AESHD/AES_HD_Dataset
[2] https://github.com/ANSSI-FR/ASCAD
[3] https://chest.coe.neu.edu/
[4] https://github.com/ikizhvatov/randomdelays-traces

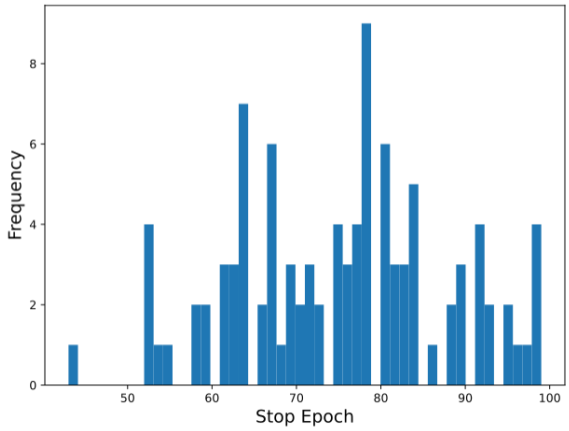


Fig. 4: Histogram of training epochs stopped by EMP-GE

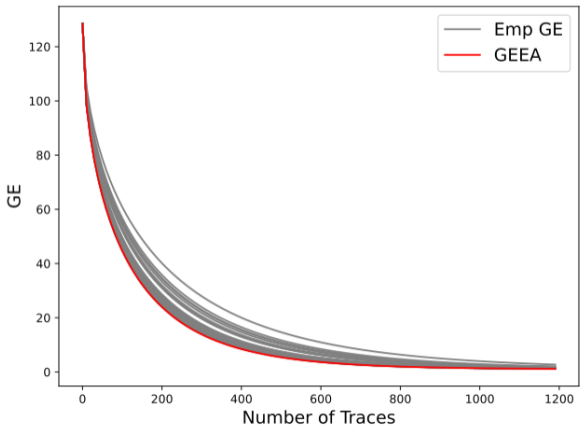


Fig. 5: GEEA based stopping rule versus EMP-GE based stopping rule

proposed GEEA end model and present the result in Figure 5. Each curve depicts how the GE value changes along the size of the attack dataset for each model. It shows that, the effectiveness of attacking models selected by the EMP-GE metric could have large variation. For example, in order to reduce the GE value less than 20, the number of traces required by EMP-GE selected models could range between 253 and 497. Therefore, it is highly possible that a sub-optimal model ends up being selected. However, the GEEA criteria stops the training at a the optimal epoch where the end model leads to most effective follow-on SCA measured by GE.

Another related metric, Guessing Entropy Bias-variance Decomposition (GEBVD) [vdVP19], takes this inherent variation of EMP-GE into consideration. GEBVD separates the bias and the variance of EMP-GE, and tries to minimize the bias as well as the variance. However, GEBVD as a validation metric does not guarantee selection of the DNN model with the minimum GE in the training process, while the GEEA validation metric directly ensures minimization of the GE. In addition, the computational complexity of GEBVD is much higher than even the EMP-GE.

Compared to those EMP-GE related validation metrics, GEEA is much faster and as a validation metric ensures the selection of minimum GE model during heterogeneous training with another loss function. In the following, we further compare GEEA with two other non-EMP-GE related validation metric for heterogeneous training.

C. Comparison between GEEA-based and other metric-based heterogeneous training

Two other non-EMP-GE related validation metrics, namely EMP-MI and $d_{train;val}$, have also been proposed for the heterogeneous training framework on DNN-assisted SCAs. We conduct numerical experiments to compare them with our GEEA-based heterogeneous training framework.

EMP-MI Metric: As Guilherme Perin et al. proposed, the mutual information between internal representations of the DNN model output layer and its labels can be used as another reliable evaluation metric [PBP20]. A histogram based empirical method is used to estimate the mutual information in a lightweight way.

$d_{train;val}$ Metric: In the work of Damien et al [RZC+20], a new metric is used to monitor the effectiveness of the attacking model. $d_{train;val} = |N^d_{val} - N^d_{train}|$, where $N^d_{train}; N^d_{val}$ measure the minimal number of traces that a model needs in order to reach an 90% first-order success rate on the training and validation dataset, respectively.

The comparison of our GEEA-based training framework and the above two training framework are conducted on the AES_HD and ASCAD datasets. The experimental setup is as follows.

Labeling function: We conduct two sets of experiments using 9 classes HW labeling function and 256 classes SBox output labeling function, respectively, to compute the label for model training and evaluation;

Loss function: For the 256 classes and the 9 classes labels, we respectively used the NLL and CER loss functions defined below. Over a given dataset $S = (x_i; z_i)_{1 \le i \le N}$ and corresponding DNN outputs $y^S() = (y^S_i())_{1 \le i \le N}$,

1) Negative log-likelihood (NLL) is defined as

$$NLL = -\frac{1}{N}\sum_{i=1}^{N} \log_2[y^S_i(; z_i)]: \qquad (8)$$

For the 256 classes labels, we use the NLL loss since it has been used to train the model to achieve state-of-art follow-on SCA performance [WAGP20], [ZBHV20]. However, for a dataset which is unbalanced across the classes (which occurs often for the 9-class HD/HW leakage models), the NLL-based training may lead to a non-optimal DNN for the follow-on SCA or even lead to an unsuccessful follow-on SCA [PHJ+19], [ZZN+20].

2) Cross Entropy Ratio (CER) is recently proposed at CHES 2020 [ZZN+20] to train DNN for SCAs when the training dataset is unbalanced. While NLL only focuses on the correct key (and the corresponding class), CER computes the ratio between the NLL

of the correct key and the average NLL over all other wrong keys.

$$CER = \frac{\frac{1}{N}\sum_{i=1}^{N} \log_2[y_i^S(;z_i)]}{\frac{1}{R}\sum_{r=1}^{R}\frac{1}{N}\sum_{i=1}^{N} \log_2[y_i^S(;z_i^r)]} \quad (9)$$

where $z^r = (z_i^r)_{1 \le i \le N}$ is a random shuffle of z. We use CER as the loss function to train the 9 class HW/HD labeled models in the heterogeneous training simulations here.

Model structures: On each public dataset and for each labels/losses combination, we apply the state-of-art DNN model selection methodology for SCA given in [WAGP20] to search and select the combination of model architectures and hyper-parameters. The detailed results are provided in the supplemental materials.

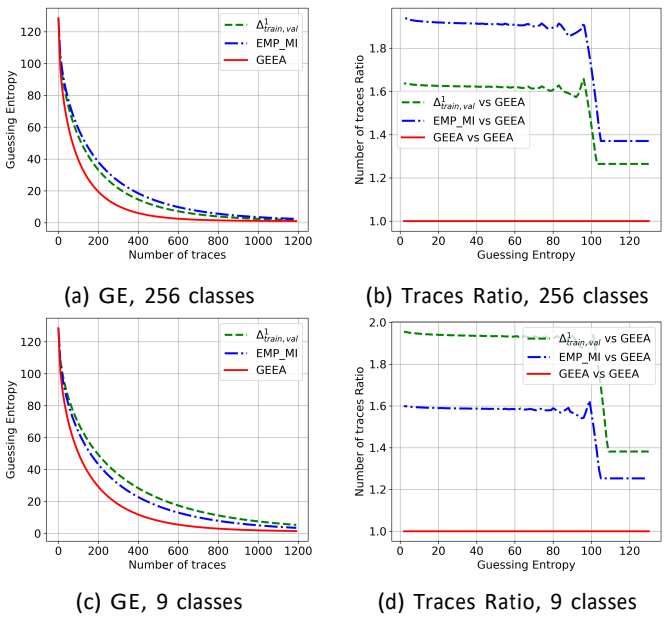
Training and Evaluation: The DNN training is repeated with various combination of hyperparameters described in table III, and the hyperparameters are selected to minimize the validated loss value within 100 epochs using grid search. At each epoch, we evaluate the three different validation metrics EMP-MI, $^{d}_{train;val}$ and GEEA. For each of the validation metrics, an end DNN model is generated with the model parameters values at the epoch for which the validation metric is optimized. The end models are applied onto the testing dataset for SCAs, and we compare the attack results.

TABLE III: Choice of Model hyperparameters for heterogeneous framework comparison

Hyperparameter	Candidates
Optimizer	Adam [KB17], RMSprop, Adamax
Weight Initialization	HeUniform [HZRS15] RandomUniform, RandomNormal
Learning Rate	10^{-4}, $5\,10^{-4}$, 10^{-3}
Batch Size	20, 50, 200, 500

Fig. 6 shows the comparison results on the AES_HD dataset. Fig. 6(a) plots the testing GE for each end-model versus number of attack traces used, for attack based on the 256 classes SBox output. While all models lead to successful SCA with enough attack traces, the most effective SCA comes from the GEEA-based training. To illustrate the magnitude of improvement, we plot in Fig. 6(b) the ratio of the number of traces to achieve the same GE value for each metrics-based training versus the GEEA-based training. Note that GE=128 indicates an attack that is no better than a random guess, and no traces are needed to achieve that. The ratio of number of traces needed to achieve GE close to 128 would get closer to one between any two methods. Also, to achieve GE=1, all methods would need infinite number of traces, so the ratio is not very meaningful there either. Compared with GEEA-based training, EMP-MI-based training needs more than 1.9 times traces to achieve the same GE for most of GE values. The $^{d}_{train;val}$-based training requires more than 1.6 times number of traces than for the GEEA-based training.

Fig. 6(c) and (d) plots the results for comparison for SCAs based on the 9 classes Hamming Weights. Here the



(a) GE, 256 classes (b) Traces Ratio, 256 classes

(c) GE, 9 classes (d) Traces Ratio, 9 classes

Fig. 6: Heterogeneous training frameworks comparison on AES_HD dataset

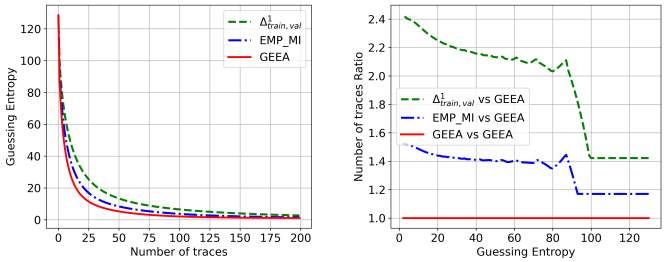
EMP-MI-based training beat the $^{d}_{train;val}$-based training, in contrast to the 256 classes attack. Compared with GEEA-based training, EMP-MI-based and $^{d}_{train;val}$ -based training respectively need more than 1.6 times and 1.9 times traces to achieve the same GE=2.

Figure 7 shows the comparison results on the $ASCAD_{Fix}$ dataset. For the 256 classes attack, compared to that from the GEEA-based heterogeneous training, the EMP-MI based heterogeneous training needs about 1.7 times number of traces and $^{d}_{train;val}$ 2.1 times, to achieve GE=2. For the 9 classes attack, about 1.8 times (EMP-MI) and 2 times ($^{d}_{train;val}$) number of traces as that of GEEA-based heterogeneous training is needed to achieve GE=2.

Figure 8 displays the comparison results on the $ASCAD_{Rand}$ dataset (random key for each encryption). Overall they all require higher ratios of traces to GEEA-based training than the prior synchronized dataset.

V. Comparison of GEEA-based Homogeneous Training Framework and Other Homogeneous Training Frameworks

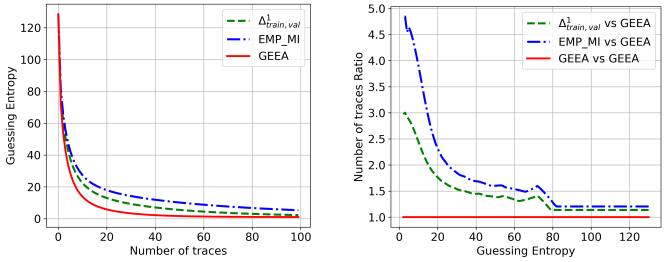
In this section, we compare homogeneous training frameworks for DL-assisted SCA using loss functions in literature with the GEEA-based homogeneous training framework. In the homogeneous training frameworks, while updating the parameter values according to the training loss $L^{Tr}()$, the validation loss $L^{Val}()$ is monitored at the same time and used to decide when to stop the training. Before our work, there are at least four existing loss functions that has been used to train DNNs for conducting SCAs. In addition to the NLL (8) and the CER (9) that we introduced in section IV-C, we lists two more loss functions.



(a) GE, 256 classes (b) Traces Ratio, 256 classes

(c) GE, 9 classes (d) Traces Ratio, 9 classes

Fig. 7: Heterogeneous training frameworks comparison on $\text{ASCAD}_{\text{Fix}}$ dataset



(a) GE, 256 classes (b) Traces Ratio, 256 classes

(c) GE, 9 classes (d) Traces Ratio, 9 classes

Fig. 8: Heterogeneous training frameworks comparison on $\text{ASCAD}_{\text{Rand}}$ dataset

Ranking loss (RkL) is asymptotically related to the success rate of the follow-on SCA, and is proposed by Gabriel Zaid et al. [ZBD+20] as:

$$RkL = \sum_{k \in K; k=k_c} \log_2 \left(1 + e^{\left(\sum_{i=1}^{N_P} y_i(z_{i;k}) - y_i(z_{i;k_c})\right)}\right), \quad (10)$$

where is a constant set by the attacker.

Correlation loss function (COR) is proposed specially for COR-attack [RQL18]. In such an attack, the model output $y_i^S()$ is a scalar (e.g., HW/HD) and

$$COR = -\frac{\sum_{i=1}^{N_P}(y_i^S() - \bar{y}^S())(z_i - \bar{z})}{\sqrt{\sum_{i=1}^{N_P}(y_i^S() - \bar{y}^S())^2 \sum_{i=1}^{N_P}(z_i - \bar{z})^2}} \quad (11)$$

We conduct the homogeneous training of DNN using the above four loss functions on the AES_HD, $\text{ASCAD}_{\text{Fix}}$, and $\text{ASCAD}_{\text{Rand}}$ datasets used in last section. That is, the end DNN model's weights are set at values at the epoch corresponding to the minimum validation loss value. To assess the performance of each training framework, we conduct a grid search on three variation of model structures proposed by Lennert Wouters et. al [WAGP20] in combination of rest hyperparameters described in table III for each homogeneous training framework, and the test GE values of the best end model is used as the assessment. The experiments are conducted for attacks using 9-class HW labeling function and 256-class SBox output labeling functions, respectively. The COR loss function is used to train a DNN for a COR-attack while the other three loss functions are used to train DNNs for an NLL-attack. Since the COR-attack only works for the 9-class attack, only the 9-class labeling is used for the COR-based training, while training with the other loss functions is used for both 9-class and 256-class labeling.

For comparison, we implemented homogeneous training of DNN for NLL-attack using GEEA loss function on both the 9 classes and the 256 classes. The proposed GEEA $\widehat{GE}_q^{Tr}()$ loss function has one distinct difference from other standard loss functions. Usually the loss function is defined for each data point: the loss between a DNN output and the target output. Commonly the DNN training over a large dataset is done in batches: on each batch, the loss can be calculated as the sum of the losses over all data points in the batch. In contrast, for the loss function $\widehat{GE}_q^{Tr}()$, it requires all the data points to participate all together to derive the Gaussian means $\hat{\mu}_{k_g}^{S_{Tr}}$ and standard deviations $\hat{\sigma}_{k_g}^{S_{Tr}}$ (shown in Equation (5)), and then plug them in Equation (6). To enable the batch training with GEEA loss, we calculate the batch GEEA $\widehat{GE}_q^{batch}()$ over the datapoints in each batch. The gradient of $\widehat{GE}_q^{batch}()$ is used in the batch training of Scheme 1.

Figure 9 plots the comparison results on the AES_HD dataset. Fig. 9(a) plots the testing GE for each end-model versus number of attack traces used, for attack based on the 256 classes SBox output. Fig. 9(b) plots the corresponding ratios of the number of traces to achieve the same GE value versus the number of traces needed by GEEA-based training. We can see that GEEA-based homogeneous training results in the most effective DL-assisted SCA. To achieve a GE=2, the other homogeneous training needs more than 3.5 times (RkL) , 2.5 times (CER) and 1.5 times (NLL) number of traces.

In the GEEA-based homogeneous training, the GEEA is used as both the loss function and the validation metric. To further study how much improvement is due to its usage as the loss function versus how much the improvement is due to its usage as the validation metric, we also compare the GEEA-based homogeneous training SCA effectiveness versus the heterogeneous training SCA effectiveness from using the

GEEA validation metric with each of the other three loss functions. Those SCAs from heterogeneous training, compared to that from the GEEA-based homogeneous training, need about 3 times (RkL) , 2 times (CER) and 1.3 times (NLL) number of traces to achieve GE=2. So using GEEA as the loss function, rather than using GEEA as the validation metric, accounts for majority of the improvement.

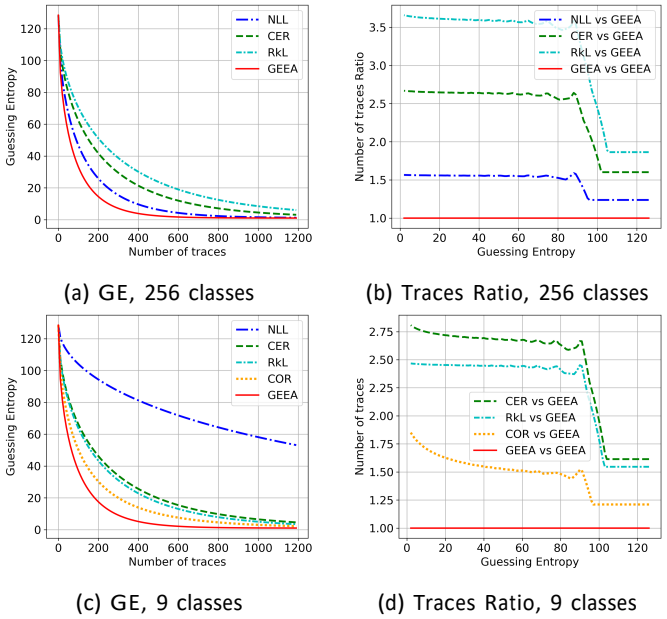
Fig. 9(c) and (d) plots the results for SCAs based on the 9-class HW labeling. For the 9-class HW model, the GEEA can be calculated for both NLL-attack and COR-attack, and we found that the end models when trained with GEEA result in attacks with very similar GEs for either case. The two plotted GE curves virtually overlap with each other. Hence, we only plot the GE curve for the NLL attack with GEEA-loss training. We also plot GE curves for attacks with models trained using the other four loss functions in equations (8) to (11). Note the NLL-loss training leads to an attack much worse compared with the training using other loss functions: the SCA using the NLL-loss trained end model would need about 20 times more traces to achieve the same GE as the SCA using the GEEA-loss trained end model. We have to omit the plot for the ratio of NLL-loss training in (d) as it is at very different scale from others (in the range of [2, 2.8]).

Another finding for the four loss functions is that there is no one consistently outperforming others under different scenarios. In 256-class attacks where there is no class imbalance issue, the traditional NLL loss training works very well, beating the other two loss-functions (CER and RkL) while the COR loss cannot be applied here. However, in 9-class attacks, NLL-based training performs poorly while the COR-based training results in the most effective SCA. The GEEA loss training, however, beats all other loss function trainings for either the 256-class or the 9-class attacks. Since GEEA loss training directly tries to minimize the GE of follow-on SCA, it automatically leads to the best DNN for follow-on SCA in all cases.

Figure 10 shows the comparison results on the ASCAD synchronized dataset. Note that the relative performance among the RkL, CER and NLL does not follow the same pattern as on the AES_HD dataset. The RkL loss training now beats the CER training. For 9-class attacks, the NLL-based training again performs very poorly. The SCAs from training under other three loss functions performs very similarly, and all needs about 40% more number of traces than GEEA-loss training to achieve the same GE.

Figure 11 shows the comparison results on the ASCAD desynchronized N50 dataset (with maximally 50 time samples random shifting amount). The relative performance among the non-GEEA loss training differs from that on the two previous datasets. Now the NLL-based training also results in the worst SCA effectiveness for 256-class attacks. The COR-based training now performs worse than the RkL/CER loss training for 256-class attacks, the opposite pattern as observed on the AES_HD dataset. The GEEA-based training still leads to the most effective SCA.

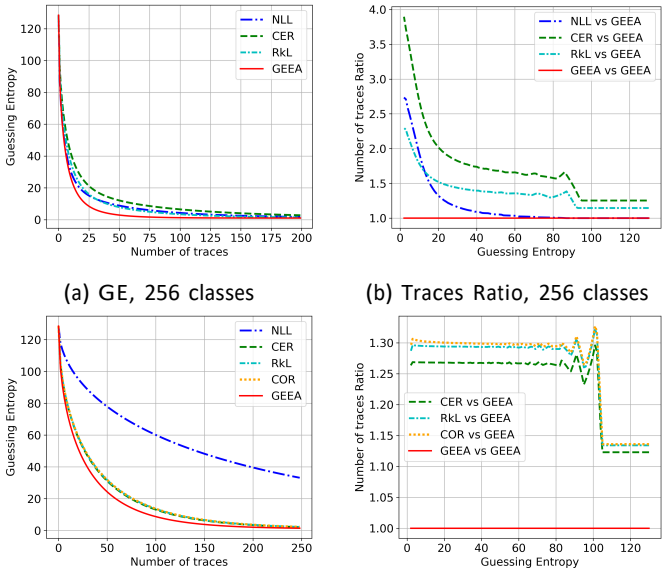
From the experiments, we see that there is no clear pattern among the performance of training under non-GEEA loss functions. The best performing method varies from datasets to



(a) GE, 256 classes (b) Traces Ratio, 256 classes

(c) GE, 9 classes (d) Traces Ratio, 9 classes

Fig. 9: Homogeneous Training frameworks comparison on AES_HD dataset



(a) GE, 256 classes (b) Traces Ratio, 256 classes

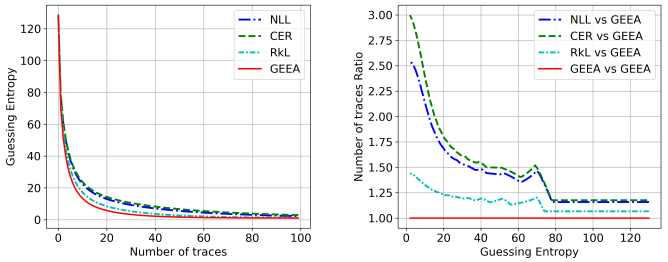
(c) GE, 9 classes (d) Traces Ratio, 9 classes

Fig. 10: Homogeneous Training frameworks comparison on $\text{ASCAD}_{\text{Fix}}$ dataset

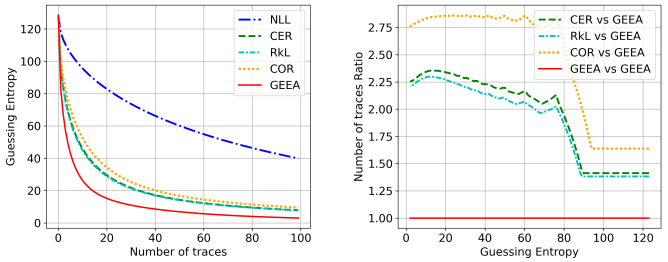
datasets, and varies from 256 classes attack to 9 classes attack. The GEEA-based homogeneous training consistently results in the most effective SCA, and demonstrates great potential to be adopted as the standard training procedure for SCA-related DNN training.

A. Further Evaluations between the Frameworks and on Success Rate

The optimization of GE can be achieved either using GEEA as the validation metric in the heterogeneous training or using

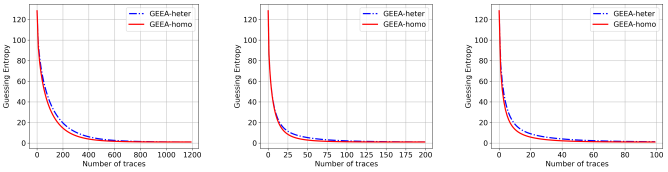


(a) GE, 256 classes (b) Traces Ratio, 256 classes



(c) GE, 9 classes (d) Traces Ratio, 9 classes

Fig. 11: Homogeneous Training frameworks comparison on $\text{ASCAD}_{\text{Rand}}$ dataset

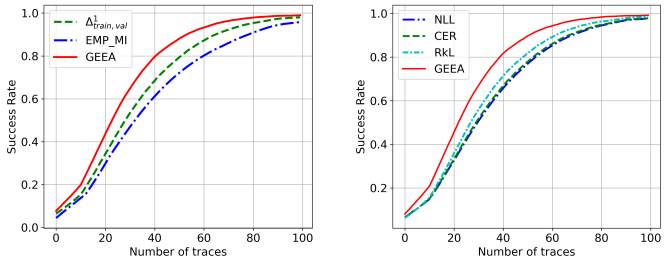


(a) AES_HD (b) $\text{ASCAD}_{\text{Fix}}$ (c) $\text{ASCAD}_{\text{Rand}}$

Fig. 12: GEEA-based Training Frameworks comparison among multiple datasets

GEEA as both the loss function and the validation metric in homogenous training. Intuitively, the homogenous training is more directly aimed at GE optimization and should achieve more effective SCA (i.e., lower GE). We plot in Figure 12 the GE curve of SCAs using heterogeneous training and the GE curve of SCAs using homogenous training from the above studies. The homogenous training does lead to SCAs with lower GE but not by much.

Success rate (SR) is another commonly used performance metric of SCAs. A more effective SCA with lower GE often also has higher SR. We also compared the SR of all DL-SCA in the above experiments, and the GEEA-training based DL-SCA has highest SR for all the comparison settings on all data sets described in Sections IV and V. Figure 13 shows such comparisons on $\text{ASCAD}_{\text{Rand}}$ dataset: the SR curves for DL-SCAs using both heterogeneous training and homogeneous training frameworks. We can see that the SR for the GEEA-trained DL-SCA (the red curve) is significantly higher than all other existing training methods. Noticeably, although optimizing the RkL metric theoretically is equivalent to optimizing the asymptotical SR [ZBD+20], the SCA for model trained with RkL has lower SR than the GEEA-trained SCA. This is likely due to the fact that asymptotical conditions for the



(a) Heterogeneous Training Frameworks (b) Homogeneous Training Frameworks

Fig. 13: Success Rate comparison among training frameworks on $\text{ASCAD}_{\text{Rand}}$ dataset

theoretical equivalency are far from holding on the real data sets.

Although the GEEA-based training do lead to SCAs with SR higher than all existing methods in all our experiments, we do not claim that GEEA-based training optimize the SR of the follow-on SCA since theoretically SR and GE do not always have a monotone relationship. To ensure optimizing SR of the follow-on SCAs, methods aiming directly at SR still need to be developed.

VI. Discussions and Conclusions

DL-assisted SCAs have outperformed traditional SCA in many cases, and have become the state-of-art analysis method used in side-channel security and countermeasure evaluation. However, there lacks a consensus on what are the most appropriate loss function and validation metric when training DNN for usage in SCAs. Training with traditional loss functions and classification validation metrics has been shown to work poorly for the follow-on SCAs, particularly for unbalanced dataset. This has prompted many proposals on SCA-aware loss functions and validation metrics. However, as we demonstrate in the paper by both theoretical analysis and experimental results, the existing loss functions are not robust - the best performing loss function varies with the attack models and datasets.

We propose to establish a standard training framework specifically focusing on the SCA metric - Guessing entropy. Using the recently proposed GEEA estimator for fast online GE evaluation, we can train the DNN to optimally support the follow-on SCA. The proposed GEEA-based homogeneous training framework consistently yields the most effective SCA across various attack models and datasets.

Another commonly used SCA metric is success rate (SR). In principle we can similarly design DNN training methods to focus on optimizing the SR of follow-on SCAs. However, the technical difficulty we face for SR focused training is that there is no analytic formula for SR that is differentiable against the weights in DNN. Literature has given analytic formulas for SR based on a multivariate Gaussian distribution, which involve high-dimensional integral that can not be evaluated easily. Empirical evaluation of that formula is fast but non-differentiable, rendering it infeasible to use as a training loss function. Some approximate SR formulas do not guarantee

maximum SR in training. In contrast, GEEA reduces the GE evaluation from the multivariate Gaussian distribution to a sum of univariate Gaussian probabilities. Thus the GEEA provides a fast differentiable formula, allowing its use as the loss function and the validation metric which can guarantee training for the minimum GE of follow-on SCA.

References

[BCN18] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning, 2018.

[BPS+20] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. Deep learning for side-channel analysis and introduction to ascad database. Journal of cryptographic engineering, 10(2):163–188, 2020.

[CCC+19] Mathieu Carbone, Vincent Conin, Marie-Angela Cornélie, François Dassance, Guillaume Dufresne, Cécile Dumas, Emmanuel Prouff, and Alexandre Venelli. Deep learning to evaluate secure RSA implementations. IACR Transactions on Cryptographic Hardware and Embedded Systems, pages 132–161, 2019.

[CDP17] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures. In International Conference on Cryptographic Hardware and Embedded Systems, pages 45–68. Springer, 2017.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), December 2015.

[KB17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[KW52] J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. Annals of Mathematical Statistics, 23:462–466, 1952.

[MDP20] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. A comprehensive study of deep learning for side-channel analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems, pages 348–375, 2020.

[MPP16] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. In International Conference on Security, Privacy, and Applied Cryptography Engineering, pages 3–26. Springer, 2016.

[PBP20] Guilherme Perin, I. Buhan, and S. Picek. Learning when to stop: a mutual information approach to fight overfitting in profiled side-channel analysis. IACR Cryptol. ePrint Arch., 2020:58, 2020.

[PHJ+19] S 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 Transactions on Cryptographic Hardware and Embedded Systems, 2019(1):1–29, 2019.

[Rob07] H. Robbins. A stochastic approximation method. Annals of Mathematical Statistics, 22:400–407, 2007.

[RQL18] Pieter Robyns, Peter Quax, and Wim Lamotte. Improving cema using correlation optimization. Transactions on Cryptographic Hardware and Embedded Systems, 2019(1), 2018.

[RQL19] Pieter Robyns, Peter Quax, and Wim Lamotte. Improving CEMA using correlation optimization. IACR Trans. Cryptographic Hardware & Embedded Systems, 2019(1):1–24, 2019.

[RZC+20] Damien Robissout, Gabriel Zaid, Brice Colombier, Lilian Bossuet, and Amaury Habrard. Online performance evaluation of deep learning networks for side-channel analysis. Cryptology ePrint Archive, Report 2020/039, 2020. https://ia.cr/2020/039.

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

[WAGP20] Lennert Wouters, Victor Arribas, Benedikt Gierlichs, and Bart Preneel. Revisiting a methodology for efficient cnn architectures in profiling attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020(3):147–168, Jun. 2020.

[WPB19] Léo Weissbart, Stjepan Picek, and Lejla Batina. One trace is all it takes: Machine learning-based side-channel attack on EdDSA. In International Conference on Security, Privacy, and Applied Cryptography Engineering, pages 86–105. Springer, 2019.

[WVdHG+20] Lennert Wouters, Jan Van den Herrewegen, Flavio D. Garcia, David Oswald, Benedikt Gierlichs, and Bart Preneel. Dismantling dst80-based immobiliser systems. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020(2):99–127, Mar. 2020.

[ZBD+20] Gabriel Zaid, Lilian Bossuet, François Dassance, Amaury Habrard, and Alexandre Venelli. Ranking loss: Maximizing the success rate in deep learning side-channel analysis. Transactions on Cryptographic Hardware and Embedded Systems, 2021(1), 2020.

[ZBHV20] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Methodology for efficient cnn architectures in profiling attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020(1):1–36, 2020.

[ZDF20] Ziyue Zhang, A. Adam Ding, and Yunsi Fei. A fast and accurate guessing entropy estimation algorithm for full-key recovery. Transactions on Cryptographic Hardware and Embedded Systems, 2020(2), 2020.

[ZZN+20] Jiajia Zhang, Mengce Zheng, Jiehui Nan, Honggang Hu, and Nenghai Yu. A novel evaluation metric for deep learning-based side channel analysis and its extended application to imbalanced data. Transactions on Cryptographic Hardware and Embedded Systems, 2020(3), 2020.