On the Evaluation of Deep Learning-Based Side-Channel Analysis

Wu, Lichao; Perin, Guilherme; Picek, Stjepan

DOI
10.1007/978-3-030-99766-3_3

Publication date
2022

Document Version
Final published version

Published in
Constructive Side-Channel Analysis and Secure Design - 13th International Workshop, COSADE 2022, Proceedings

Citation (APA)
Wu, L., Perin, G., & Picek, S. (2022). On the Evaluation of Deep Learning-Based Side-Channel Analysis. In J. Balasch, & C. O'Flynn (Eds.), *Constructive Side-Channel Analysis and Secure Design - 13th International Workshop, COSADE 2022, Proceedings* (pp. 49-71). (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Vol. 13211 LNCS). Springer. https://doi.org/10.1007/978-3-030-99766-3_3

On the Evaluation of Deep Learning-Based Side-Channel Analysis

Lichao Wu[1], Guilherme Perin[1], and Stjepan Picek[1,2](✉)

[1] Delft University of Technology, Delft, The Netherlands
[2] Radboud University, Nijmegen, The Netherlands
stjepan@computer.org

Abstract. Deep learning-based side-channel analysis is rapidly positioning itself as a de-facto standard for the most powerful profiling side-channel analysis.The results from the last few years show that deep learning techniques can efficiently break targets that are even protected with countermeasures. While there are constant improvements in making the deep learning-based attacks more powerful, little is done on evaluating the attacks' performance. Indeed, how the evaluation process is done today is not different from what was done more than a decade ago from the perspective of evaluation metrics.

This paper considers how to evaluate deep learning-based side-channel analysis and whether the commonly used approaches give the best results. To that end, we consider different summary statistics and the influence of algorithmic randomness on the stability of profiling models. Our results show that besides commonly used metrics like guessing entropy, one should also show the standard deviation results to assess the attack performance properly. Even more importantly, using the arithmetic mean for guessing entropy does not yield the best results, and instead, a median value should be used.

Keywords: Side-channel Analysis · Deep Learning · Guessing Entropy · Median

1 Introduction

Side-channel analysis (SCA) encompasses techniques aiming at exploiting weaknesses of algorithms' implementations [11]. One standard division of SCA is into direct attacks and profiling attacks. Profiling attacks (two-stage attacks) are more powerful but require a stronger attacker who can access a copy of a device to be attacked. The attacker uses that copy to build a model of a device to be used to attack another similar (identical) device. In the last few years, the most explored profiling attacks have been based on machine learning (especially deep learning). Such attacks are very powerful as they can break targets protected with countermeasures [3,6] but are also somewhat "easier" to deploy as they do not necessarily require pre-processing/feature engineering stages [9,15]. Still, many open questions are usually connected with how to find machine learning architectures that perform well [27,30]. Unfortunately, this is just one side of the problem. A perspective that cannot be neglected is how to assess the performance of such a profiling model. While the state-of-the-art in deep learning SCA progressed tremendously in the last few years, no results consider how to evaluate the performance of such attacks and if commonly used techniques are the most appropriate ones.

It is common to use metrics like key rank, success rate, and guessing entropy to evaluate the attack performance in SCA [1,6,27,30]. While the first metric requires one experiment run, the latter two are run multiple times to counteract the effect of dataset/measurements selection. For direct attacks or simpler profiling attacks like the template attack, this repetition is sufficient as the algorithms are deterministic, so running them multiple times gives the same results (if the measurements and selected features do not change). On the other hand, deep learning techniques (i.e., artificial neural networks) have multiple sources of randomness (due to the initialization, regularization, and optimization procedure), making those algorithms stochastic. The randomly initialized weights and biases with selected initialization methods make the models perform differently before training, which may lead to performance variation after training as well. Regularization techniques like dropout randomly 'switch-off' some neurons, leading to unpredictable model behaviors. Optimization algorithms, such as stochastic gradient descent (SGD) and Limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm (L-BFGS), can lead to significant performance variation as well due to their different working principles. Thus, it is intuitive to expect different results when training deep learning models (and including the above-mentioned random sources multiple times), making the evaluation of the attack performance not straightforward. This problem becomes even more challenging when considering the differences among various neural network architectures.

To the best of our knowledge, there are not many works assessing the evaluation performance of side-channel attacks. For instance, Martin et al. investigated how to estimate key rank distribution for SCA [12]. Whitnall and Oswald considered robust profiling setting [26], which can also be connected with the stability of a profiling model, as intuitively, a more robust profiling model provides more stability. Picek et al. considered the robustness through the expectation estimation problem and provided theoretical foundations to assess the robustness of deep learning-based SCA [20]. The authors concluded that deep learning algorithms are robust, but they did not consider improving the evaluation process.

This paper investigates how to evaluate the attack performance of deep learning-based SCA. Our main contributions are:

- We investigate the influence of algorithmic randomness on the attack performance. More precisely, we use the standard deviation to showcase that running experiments multiple times can result in a significantly different assessment of the attack performance. This difference in the attack performance is confirmed for scenarios that use 1) different random models, and 2) the same profiling model and train it independently several times (where the randomness comes from the algorithmic settings).

- We investigate the most appropriate summary statistic for evaluating the attack performance. We consider the arithmetic mean, geometric mean, and median and show that the median works the best (fastest convergence). Our results indicate that deep learning-based SCA often results in skewed distributions of the attack performance, so the arithmetic mean is not appropriate statistics, which is relevant as it is commonly used in the SCA domain.
- We investigate how a different number of independent experiments (key rank evaluations) in the attack phase influences attack performance. Our results show that this value does not significantly influence the results, so much smaller values can be safely used.

We conduct an extensive experimental evaluation including three datasets, two leakage models, and different types of neural networks (multilayer perceptrons and convolutional neural networks) to confirm our observations.

2 Machine Learning-Based Side-Channel Analysis

We concentrate on supervised machine learning and the multi-class classification task (with c classes), as commonly done in related works (see Sect. 3). Supervised machine (deep) learning classification represents the task of learning a function f that maps an input to the discrete output ($f : \mathcal{X} \rightarrow Y$)) based on examples of input-output pairs. The function f is parameterized by n parameters learned in the profiling model: $\boldsymbol{\theta} \in \mathbb{R}^n$.

Hyperparameters are the variables determining the network structure (e.g., the number of neurons and layers) and the variables determining how the network is trained (e.g., learning rate). The parameters are the coefficients chosen through learning (e.g., weights).

Training. In the training phase, the goal is that the algorithm learns the parameters $\boldsymbol{\theta}$ minimizing the empirical risk represented by a loss function on a training dataset of size N.

Validation. When training a profiling model, it should generalize well to previously unseen data, i.e., the profiling model shows stability. To this end, it is common to use cross-validation techniques. Cross-validation is a statistical validation technique used to assess the performance of a machine learning model. Two commonly employed cross-validation techniques in the machine learning-based SCA are 1) validation and 2) k-fold cross-validation.

With the validation technique, we divide the dataset into training (size N), validation (size V), and test dataset (size Q) and use the validation dataset to assess the performance of a model trained on the training dataset. Finally, we use the best-obtained model to attack the test dataset. This technique is commonly used with deep learning-based SCA (due to computational simplicity) [22,28,30].

In the k-fold cross-validation, a dataset is divided into k parts. Then, a model is built on $k-1$ folds and evaluated on the k-th fold. This process is repeated until each fold serves as the k-th fold (every combination of $k-1$ folds serves to train the model). This technique is commonly used with computationally simpler machine learning techniques [20].

Test. In the test phase, the goal is to predict classes (or probabilities that a specific class would be predicted) y based on the previously unseen traces $\mathbf{x}$ (the number of traces equals Q), and the trained model f.

Evaluating the Attack Performance. The outcome of predicting with a model f on the attack set is a two-dimensional matrix P with dimensions equal to $Q \times c$. The cumulative sum $S(k)$ for any key byte candidate k is a valid SCA distinguisher, where it is common to use the maximum log-likelihood distinguisher $S(k) = \sum_{i=1}^{Q} \log(\mathbf{p}_{i,y})$. The value $\mathbf{p}_{i,y}$ denotes the probability that for a key k and a specific input, we obtain the class y (derived from the key and input through a cryptographic function and a leakage model).

It is common to estimate the effort to obtain the correct key k^* with metrics like success rate (SR) and guessing entropy (GE) [25].

With Q traces in the attack phase, an attack outputs a key guessing vector $\mathbf{g} = [g_1, g_2, \ldots, g_{|\mathcal{K}|}]$ in decreasing order of probability where g_1 denotes the most likely and $g_{|\mathcal{K}|}$ the least likely key candidate. Then, guessing entropy is the average position of the correct key in $\mathbf{g}$[1]. The success rate of order o is the average empirical probability that the correct key k^* is located within the first o elements of the key guessing vector $\mathbf{g}$.

Sources of Randomness in Deep Learning-Based SCA. When considering deep learning, several common sources of randomness will influence the obtained results. Informally, the random sources are connected with the dataset (dataset randomness) and the machine learning algorithm (algorithmic randomness). Dataset randomness is caused by the random selection of the traces included in the training/attack dataset. Averaging multiple results is a common way to reduce the effect of any specific traces. While the choice of traces can significantly influence the results, we consider it out of scope for this paper, as it influences any side-channel attack and not only the deep learning ones. For more results about attack performance when selecting different traces, see [29].

In terms of algorithmic randomness, we can obtain different results even if training/evaluating a neural network on the same set of traces (for experiments, see Sect. 5.2, Fig. 1). Indeed, the setting of the random seeds introduces randomness to the machine learning algorithm, where the common sources are:

– Initialization of weights and biases. Initialization of weights provides the first model that is then improved with the backpropagation algorithm. If the weights are chosen poorly (e.g., all the weights are the same value), the training process will not be efficient. The initialization of weight analysis in the context of SCA is done in [8].

[1] Averaging is commonly done over 100 independent experiments (attacks) to obtain statistically significant results.

- Regularization techniques, such as dropout. Regularization represents techniques used to reduce the error by fitting a function f appropriately on the training set. Regularization is used to prevent overfitting (when the model does not generalize to previously unseen data). Dropout is a regularization technique where during the training, some layer outputs are randomly ignored ("dropped out"). Dropout is used to approximate training many neural networks with different architectures in parallel.
- Optimization techniques used to minimize the loss function. Optimizers change the parameters (e.g., weights) of machine learning algorithms (e.g., neural networks) to reduce the loss. They can also change the hyperparameters like learning rate. The analysis of various optimization algorithms and their behavior in SCA is done in [14].

3 Related Works

The variety of techniques and choices one could take when using machine learning (even more deep learning) brought the need for much more detailed analyses, resulting in an abundance of results and papers. Indeed, since 2016, more than 120 papers have examined deep learning and SCA [21].

The first profiling SCA techniques like template attack [4] or stochastic models [23] have no tunable hyperparameters[2], making the analysis simpler and deterministic. Then, running the experiments multiple times results in the same solutions, provided that the same measurements are used. However, the data randomness is introduced by traces and feature selection and will affect the final attack performance. Note that the feature selection/engineering step is also a common one for simpler machine learning techniques[3].

Afterward, machine learning techniques like support vector machines [19], random forest [7], or Naive Bayes [5,17] started to attract more attention in the SCA community as the results were in general favorable compared to the template attack. Those techniques have different hyperparameters (except Naive Bayes, which has no hyperparameters) one needs to tune to reach their full potential. Still, the evaluation of the attack performance did not account for the algorithmic sources of randomness, and the SCA community continued to report the results in the same fashion as for the template attack (e.g., the average key rank for a specific number of attack traces).

Finally, in the last few years, we notice a general direction of using deep learning for profiling SCA. The first significant step was done by Maghrebi et al. as they showed that convolutional neural networks (CNNs) could efficiently break different targets [10]. Additionally, they showed that deep learning works well with raw traces, removing (or, at least reducing) the need for various feature selection techniques [16]. Cagli et al. demonstrated that deep learning could also break implementations protected with jitter countermeasures and introduced

[2] Besides the selection of features (points of interest).

[3] Even deep learning techniques are commonly used with a pre-selected window of features.

the concept of data augmentation in the profiling SCA [3]. Picek et al. evaluated several machine learning metrics and showed a discrepancy between those and the side-channel metrics [18]. The authors showed that the metrics problems also happen for deep learning techniques. Kim et al. designed a deep learning architecture capable of achieving excellent results on several datasets and regularized input with Gaussian noise to further improve the attack performance [6].

Benadjila et al. provided the first more detailed investigation into the importance of hyperparameter tuning [1]. Zaid et al. proposed the first methodology to select hyperparameters related to the size (number of learnable parameters, i.e., weights and biases) of layers in CNNs [30]. Wouters et al. [27] improved upon the work from Zaid et al. [30] where they showed how to reach similar attack performance with significantly smaller neural network architectures. Perin et al. investigated deep learning model generalization and showed that output class probabilities represent a strong SCA metric [13]. Wu et al. introduced Bayesian optimization for hyperparameter tuning [28]. With this approach, the authors managed to find small neural network architectures that perform well (surpassing the architectures' performance obtained by the previous methodologies). Rijsdijk et al. used reinforcement learning to find small convolutional neural networks surpassing the previous state-of-the-art results [22]. Unfortunately, these works showed the importance of hyperparameter tuning but did not consider the influence of algorithmic randomness. What is more, the ever-increasing number of hyperparameters to test resulted in a simpler (faster) validation process but also a larger variance in the results. Li et al. investigated the influence of randomness caused by the weight initialization for multilayer perceptron and convolutional neural network architectures and showed that, depending on the choice of the weight initialization method, SCA attack performance could vary significantly [8]. Perin and Picek explored the impact of the optimizer choice for deep learning-based SCA [14]. Their results indicated that some commonly used optimizers could easily overfit, requiring more effort during the training process.

4 Summary Statistics

Once we obtained the information about key rank from z independent experiments over space $\mathcal{S}$, we need to find the most appropriate estimator for the expected value of $\mathcal{S}$. A common way to do this is to use the **arithmetic mean**, where the arithmetic mean of z examples equals $\overline{x} = \frac{1}{z}\sum_{i=1}^{z} x_i$. While a common way to calculate guessing entropy, arithmetic mean has a drawback as it is dominated by numbers on a larger scale. This happens due to a simple additive relationship between numbers where scales do not play a role.

An alternative to arithmetic mean that takes into account the proportions is the **geometric mean** $\breve{x} = (\prod_{i=1}^{z} x_i)^{\frac{1}{z}}$.

We can also consider the middle value of the dataset, which is called **median** $\tilde{x} = \frac{x_{\frac{z}{2}} + x_{\frac{z}{2}+1}}{2}$. The median is less affected by outliers and skewed data than the arithmetic mean.

The **standard deviation** is a measure of the amount of variation or dispersion of a set of values $\sigma_x = \sqrt{\frac{1}{z}\sum_{i=1}^{z}(x_i - \overline{x})^2}$. In the SCA context, a large standard deviation means that the adversary will have a high probability to be "lucky" (or "unlucky") in the choice of traces or hyperparameters.

5 Experimental Evaluation

5.1 Settings

We investigate two scenarios in our experiments: random profiling models and state-of-the-art profiling models from related works. We experiment with multi-layer perceptron (MLP) and convolutional neural networks (CNNs) in the Hamming weight (HW) and Identity (ID) leakage models. Finally, we consider the ASCAD fixed key (ASCAD_F), ASCAD random keys (ASCAD_R)[4], and CHES 2018 Capture-The-Flag (CHES_CTF) datasets)[5]. For both ASCAD versions, we attack key byte 3 (the first masked key byte) and use 50 000 traces for profiling and 5 000 traces for the attack. For CHES_CTF, we use 45 000 traces with 2 200 features each for profiling and 5 000 traces for the attack, and we attack the first key byte. We opted for these settings to make our experiments aligned with related works. Additionally, it is common to attack only one key byte as it is expected that the attack difficulty should be similar for the other key bytes, see, e.g., [22,27,30].

The machine learning model was implemented in python version 3.6, using TensorFlow library version 2.0. The model training algorithms were run on a cluster of Nvidia GTX 1080 and GTX 2080 graphics processing units (GPUs), managed by Slurm workload manager version 19.05.4. The number of random profiling models is set to 100 for all experiments. We set the maximum sizes (in terms of the number of training parameters) for architectures for the random model generation to the ones from the ASCAD paper [1], which we denote as 'MLP_best' and 'CNN_best'. Since more recent state-of-the-art models are even smaller, we can assume we do not need bigger models for the dataset under investigation. The detailed model implementations are listed in Table 1. Aligned with the settings provided by the ASCAD paper [1], we use RMSProb as the optimizer with a learning rate of 1e–5. The number of training epochs is set to 75. To generate the random models from the baseline models (MLP_best and CNN_best), for CNN models, we randomized the kernel size of the convolution layer and the number of neurons in the dense layer. The latter one is also randomized for MLP models. Specifically, the range is from the $half$ of the original parameter to the original parameter. For instance, the kernel values of the first convolution layer in the CNN model range from 32 to 64. For MLP, the range of the neurons is from 100 to 200. We use diverse architectures to provide general conclusions, but they should still perform relatively well (break the target) since they are based on well-performing architectures that we do not change radically.

[4] https://github.com/ANSSI-FR/ASCAD/tree/master/ATMEGA_AES_v1.

[5] http://aisylabdatasets.ewi.tudelft.nl/ches_ctf.h5.

Table 1. Baseline MLP and CNN architectures used in the experiments.

Test models	Convolution (filter_number, size)	Pooling (size, stride)	Dense layer	Activation
MLP_best	-	-	200*5	ReLU
CNN_best	Conv (64, 128, 256, 512, 512)	avg(2,2)*5	4 096*2	ReLU

In terms of attacks with the state-of-the-art models, we used the MLP models obtained through the Bayesian Optimization [28]. The CNN models we used are developed with the reinforcement learning approach [22]. The details about the architectures are listed in Tables 2 and 3. All of the training hyperparameters are aligned with the original papers [22,28]. Specifically, CNNs use He uniform as the kernel initializer, and the corresponding learning rate is handled by OneCycleLR policy [24] with the maximum learning rate (LR) of 5e–3. For MLPs, Glorot uniform is used as the kernel initializer. Both MLPs and CNNs apply categorical crossentropy as the loss function and mini-batch as the optimization method. While there are other state-of-the-art models we could use (e.g., from [27,30]), we opted for these as the related works did not run experiments for the HW leakage model but only the ID leakage model. We used the selected state-of-the-art models as the authors provided the code for their architectures, making the risk of wrongly interpreting and implementing an architecture impossible. The training effort of each model (i.e., the number of epochs) is set based on the related works [1,22,28]. Specifically, MLP_best and CNN_best are trained with 75 epochs, while the other models are trained with 50 epochs.

Table 2. MLP architectures used in the experiments [28].

Test models	Dense layer	Activation	Learning rate
$ASCAD\_F_{HW}$	1 024, 1 024, 760, 8, 704, 1 016, 560	ReLU	1e–5
$ASCAD\_F_{ID}$	480,480	ELU	5e–3
$ASCAD\_R_{HW}$	448, 448, 512, 168	ELU	5e–4
$ASCAD\_R_{ID}$	664, 664, 624, 816, 624	ELU	5e–4
$CHES\_CTF_{HW}$	192, 192, 616, 248, 440	ELU	1e–3

In all the experiments, we conduct the following steps to obtain the results:

1. To evaluate the general performance of different averaging methods and training settings, we perform multiple independent training phases for state-of-the-art and random models. Based on the preliminary experiments, 20 independent models (thus, independent training phases of a model) are sufficient to assess the performance of the state-of-the-art models, while to evaluate the performance variation of random architectures, we increase the number of the tested models to 100.

Table 3. CNN architectures used in the experiments [22].

Test models	Convolution (filter_number, size)	Pooling (size, stride)	Dense layer	Activation
$ASCAD\_F_{HW}$	Conv(16,100)	avg(25,25)	15+4+4	selu
$ASCAD\_F_{ID}$	Conv(128,25)	avg(25,25)	20+15	selu
$ASCAD\_R_{HW}$	Conv(4, 50)	avg(25, 25)	30+30+30	selu
$ASCAD\_R_{ID}$	Conv(128, 3)	avg(75, 75)	30+2	selu
$CHES\_CTF_{HW}$	Conv(4, 100)	avg(4, 4)	15+10+10	selu

2. For each independent training, we calculate summary statistics (arithmetic mean, geometric mean, and median) for the evaluation metrics (GE, SR) over a number of attacks. Note that an attack represents an individual key rank experiment. For instance, having 100 attacks means running 100 key rank evaluations and providing summary statistics using the evaluation metrics.
3. The arithmetic average and standard deviation of the attack performance metric are plotted. Since the attack performance is averaged over profiling models, the influence of algorithmic randomness is present but not dataset randomness (in that case, we should show standard deviation over different selections of the attack traces).
4. As all of the models effectively retrieve the key or converge to close to zero guessing entropy, we use T_{GE0} (i.e., the number of attack traces to reach GE of zero) to evaluate the attack result. Note that this is still GE metric, but now, with an adjusted number of traces required for a successful attack instead of the fixed number of traces.
5. To conclude which summary statistics is the best, we consider two aspects: the metric that converges to the best value (e.g., GE of 0) and the metric that converges the fastest (with the minimum number of attack traces) to the best value. Since for most experiments provided here, we obtain the best possible value (GE of 0), the main objective is to reach the GE of 0 with the lowest number of attack traces.

Naturally, one could argue that the best metric is the one that gives the worst results as it approximates the worst-case security evaluation. However, we believe this somewhat negates the idea of using the most powerful attack approach, which is a common setup for deep learning-based SCA.

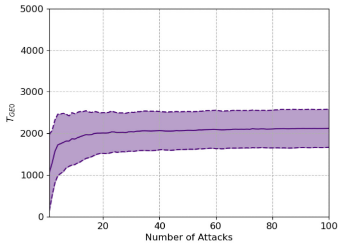
We also investigated the success rate but observed that it commonly does not change regardless of the averaging methods and thus offers limited information. Therefore, we omit these results and only present the success rate results that contain more information. We postulate this happens as success rate considers only the most likely key guess (first-order success rate). At the same time, guessing entropy uses the information from the whole key guessing vector. Thus, if the attack is more difficult, i.e., the probability differences among the best guesses are less pronounced, it will affect the guessing entropy metric more. For success rate, algorithmic randomness is less likely to cause such significant differences in the profiling models so that the most likely guess will change. To conclude, the success rate metric can help avoid the influence of outliers, but that comes with a price of less information about the attack performance.

In the next section, most of the results are plotted with the number of attacks on x-axis (for GE calculation) and T_{GE0} on y-axis. The solid lines represent the average of the T_{GE0} metric (i.e., arithmetic mean, geometric mean, or median of several independent key rank experiments), while the dashed lines of the same color indicate the upper and lower bound of the standard deviation ($\pm\ \sigma$). The spaces between of upper and lower bound are filled with the corresponding but lighter color.

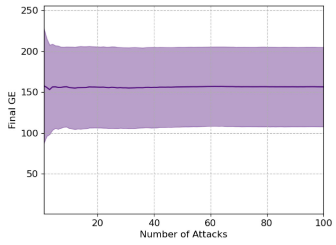
5.2 Results

A Demonstration of Algorithmic Randomness Influence. We showcase the effects of algorithmic randomness in Fig. 1 for the ASCAD fixed key dataset. We select two models from a random hyperparameter tuning: one performs well (GE converges to zero), and the other performs poorly (GE does not converge). For every value of the solid line, we train 100 random models, and for each of those random models, we run the number of attacks as denoted on the x-axis. The influence of the random weight initialization on the poor-performing model is greater than on the well-performing model over 100 independent training experiments. This behavior indicates that a better model suffers less from the random weight initialization, but there will still be differences in performance (recall, finding a model with optimal weights is difficult, and there is no methodology allowing that in the general case). The influence of the dropout layer is limited in this example (cf. Fig. 1a), but still, we can observe slight differences caused by dropout randomization. Finally, two optimization techniques, SGD and L-BFGS, are tested with the same (well-performing) models. In both cases, the attack performance varies more significantly than the original mini-batch optimization method, confirming the impact of the optimizer's randomness on the attack performance. Interestingly, L-BFGS does not reach GE of zero, making a model that performed well into a model that performs poorly.

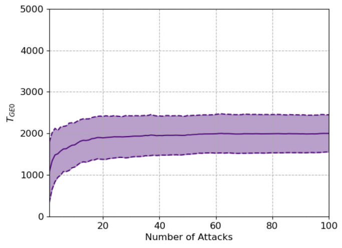
Since most deep learning-based SCAs use random search to find good hyperparameters, from Fig. 1, we can expect (radically) different evaluation results based on the used architectures. While there are already results showing that these sources of randomness introduce instability in deep learning-based SCA (as discussed in Sect. 3), there is no discussion on how to resolve such issues or at least report the results in a more meaningful way. On the other hand, the algorithm randomness is also beneficial as it gives the model a better chance to converge when training networks. For example, stochastic gradient descent uses randomness to give the model the best chance to jump out of local minima and converge to the global minimum for a convex loss function. Correspondingly, algorithm randomness should cause better model convergence and lower standard deviation under the correct settings. This assumes that the training and test data have similar distributions, and optimal hyperparameters are chosen.



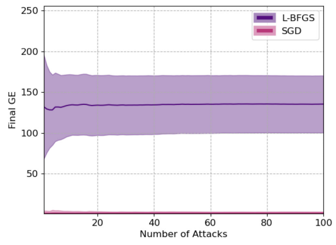
(a) T_{GE0}: Random initialization of weights and biases of a well-performing model.



(b) GE: Random initialization of weights and biases of a bad-performing model.



(c) T_{GE0}: Regularization techniques (dropout on well-performing model.)

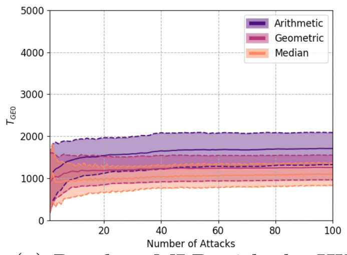


(d) GE: Optimization techniques (SGD, L-BFGS on well-performing model.)

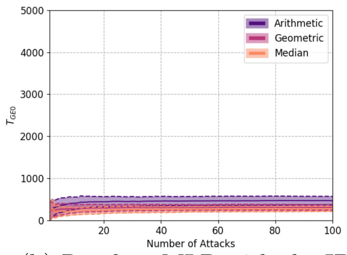
Fig. 1. A demonstration of the algorithm randomness for the Hamming weight (HW) leakage model and the arithmetic mean as summary statistics.

Since those two constraints are not easy to fulfill [2,30], algorithmic randomness can (and will) also have a negative influence on the attack performance.

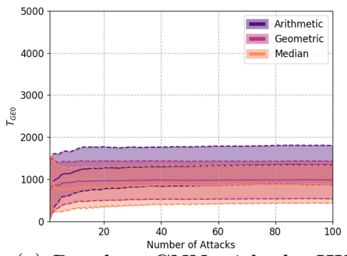
Results for the ASCAD_F Dataset. The results for random models are shown in Fig. 2. All the results indicate relatively stable behavior: when attacking with 100 random models, the median is a statistic indicating the best attack performance while the worst is the arithmetic mean. Interestingly, we can observe that the upper deviation value for the median gives similar results as the lower deviation value for the arithmetic mean, indicating that the median is a significantly better evaluation statistic. The differences in the number of attack traces are also significant: from around 700 to 2 000 attack traces. We analyzed the key rank histogram for all attacks, and outliers (failed attacks) have a significant influence on the arithmetic mean (and to a smaller extent, geometric mean), as they consider all attack results. On the other hand, the median mean is equivalent to the attack performance of a medium-performing model, thus can reliably represent the attack performance. To demonstrate this, Fig. 3 shows the GE histogram of 100 trained models with the smallest and largest averaging performance differences (see Figs. 2b and 2d). Clearly, GE calculated with the arithmetic mean tends to have larger values.



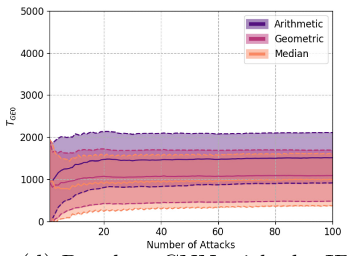
(a) Random MLP with the HW leakage model.



(b) Random MLP with the ID leakage model.

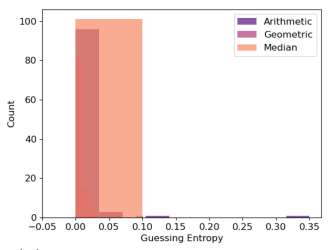


(c) Random CNN with the HW leakage model.

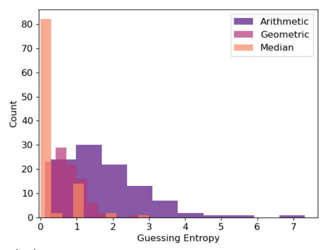


(d) Random CNN with the ID leakage model.

Fig. 2. T_{GE0}: attack on ASCAD_F with random MLP and CNN models.



(a) Random MLP with the ID leakage model.

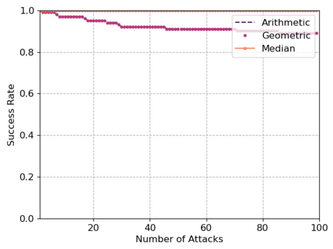


(b) Random CNN with the ID leakage model.

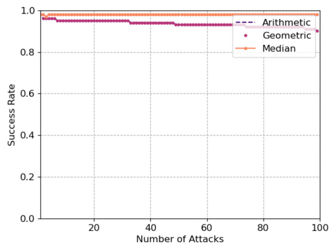
Fig. 3. Histograms of guessing entropy.

The behavior for a different number of attacks remains stable with no differences when using more than 40 attacks. This result indicates that instead of averaging 100 times as commonly done in the literature [13,22], the dataset randomness can be sufficiently countered with less computation effort. Notice how the arithmetic mean can lead to comparable or even better attack performance than its counterparts with a small number of attacks. We hypothesize this happens due to the random shuffling of attack traces and insufficient number of experiments to assess the average behavior properly. Indeed, with more attacks being performed, the increasing number of outliers introduced by data randomness can degrade the attack performance, resulting in less favorable results for the arithmetic mean. With a larger number of attacks, the standard deviation results are comparable regardless of the number of attacks, again confirming that outliers are the main contributors to the reduced attack performance for the arithmetic mean and geometric mean. From a different perspective, this indicates that random models perform well for this dataset and that more elaborate tuning mechanisms are not needed [28]. MLP for the ID leakage model shows the best results and smallest standard deviation. We postulate that this happens as the model's capacity is well aligned with the characteristic of the dataset, so most of the experiments end up with a rather similar attack performance.

We also show averaged success rate results in Fig. 4. Arithmetic mean shares the same tendency with the geometric mean, so the lines are overlapping. The rest of the results are omitted as the success rate results are the same for the three averaging methods. Compared with T_{GE0}, the success rate metric is less sensitive to the variation of the averaging methods since it uses information about the best guess only. We see a drop for both geometric and arithmetic mean with more attack results averaged, while the median remains stable. This behavior indicates that the influence of outliers when considering more attacks becomes more significant, as it skews the distribution.



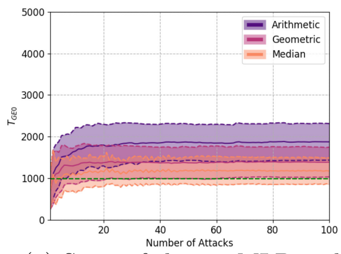
(a) Random MLP with the HW leakage model.



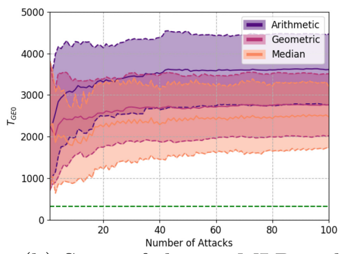
(b) Random CNN with the ID leakage model.

Fig. 4. Success Rate: attack on ASCAD_F with random MLP and CNN models.

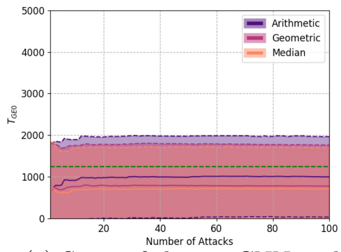
Next, we investigate the performance of four state-of-the-art models. The results are shown in Fig. 5. The green dashed line represents the attack performance reported in the original papers [22,28]. For MLP, the median gives the best results, while the arithmetic mean indicates significantly worse behavior (around twice as many traces required to reach GE of zero). Aligned with previous experiments, the increased number of attacks (i.e., larger than 50) has a limited effect on the performance of each averaging method. In terms of the attack performance of each model, the results reported in related works are better than the averaged performance from multiple models, meaning that obtaining the results on the level of those reported in related works requires a significant number of experiments (until the appropriate weights of a model are found). Large standard deviation values confirm this as many of the found models do not even approach the reported performance. Therefore, we argue that averaging with multiple models initialized with random weights should be mandatory to report their performance reliably.



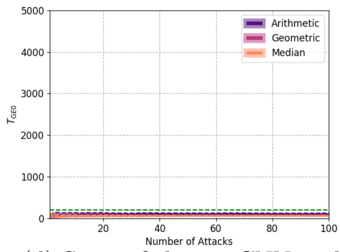
(a) State-of-the-art MLP with the HW leakage model.



(b) State-of-the-art MLP with the ID leakage model.



(c) State-of-the-art CNN with the HW leakage model.



(d) State-of-the-art CNN with the ID leakage model.

Fig. 5. T_{GE0}: attack on state-of-the-art MLP and CNN models with the ASCAD_F dataset.

The median performs the best for CNN results, aligned with the previous results. The number of attacks shows only a marginal influence, and the deviation is large for the HW leakage model while small for the ID leakage model. We hypothesize this happens as with fewer classes scenario (as it is for the HW leakage model), the profiling model has more capacity (recall that these optimized models are already quite small from the perspective of the number of trainable parameters) and more choice to end up with different performing architectures. The model capacity seems better aligned with the task for the ID leakage model, so most of the experiments end up with similar attack performance. Interestingly, we can reach an even better performance than reported in related works. We believe this happens as we (in essence) show results for ensembles of classifiers (recall, we train a single architecture but with different parameters), which is reported to work better than a single classifier [13].

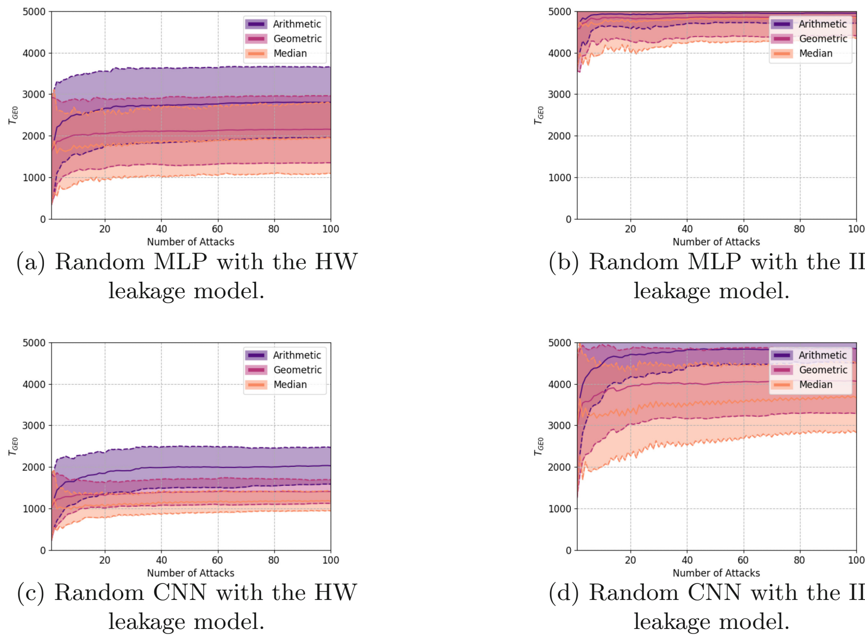
In general, there is a significant deviation even when using a single optimized model, indicating that reporting the attack performance for a single setup can be misleading. On the other hand, our results suggest that the standard deviation correlates with the model's fitness to the dataset. For example, in Fig. 5b, the models had high standard deviation, and the performance was significantly worse than the literature's performance in the green curve. Meanwhile, when looking at Fig. 5d, the standard deviation was very small, and the performance was better than the performance presented in the literature.

Results for the ASCAD_R Dataset. Recall that the profiling traces for this dataset contain random keys while the attack set contains a fixed but unknown key. This setting is closer to the real attack scenario as it increases the difficulty of retrieving the correct key from the attack set. Figure 6 presents the attack results for 100 random models. Compared with ASCAD_F, we see performance degradation, especially when attacking in the ID leakage model. For instance, when attacking with random MLP for the ID leakage model, 74% of the models failed to converge GE to zero within 5 000 attack traces. Still, even in the worst attack cases, the median reliably represents the attack result and requires the smallest number of attack traces to obtain the correct key. Aligned with the previous results, there is a limited influence of the number of attacks, while standard deviation is large for all cases except one (MLP with the ID leakage model). This result indicates that several randomly selected models perform poorly and need to be optimized.

Aligned with the previous experiment, in Fig. 7, we observe a drop in success rate for the arithmetic and geometric means when the number of attacks increases, indicating the influence of outliers. The median reaches the highest success rate of all tested averaging methods in all scenarios. We also observe a slight increase in SR for the ID leakage model with the increase in the number of attacks, suggesting significant differences among specific attacks and requiring more experiments to stabilize them. We omit other results for SR as they are similar to the presented ones.

Moving to the results for the state-of-the-art models (Fig. 8), the attack performance is significantly improved compared to the previous result on random models. This means that using random models will not suffice to reach the top attack performance due to a more difficult dataset. Again, the median performs the best, consistently indicating the superiority of this averaging method. When comparing our results with the one reported in the original papers [22,28] (green dashed line), we again see a slight mismatch between them. Specifically, the reported results for CNN with the HW leakage model act as an outlier in Fig. 8c, again emphasizing the influence of the random weight initialization and the need to provide averaged results over a number of profiling models.

The number of attacks has a small influence, but there is no reason to use more than 50 attacks in the experiments. We see a very large standard deviation for the CNN architecture and the ID leakage model, indicating that the profiling model is not stable, so multiple experiments should be done to assess the attack performance properly. Finally, for CNNs, there is the synergistic effect of using



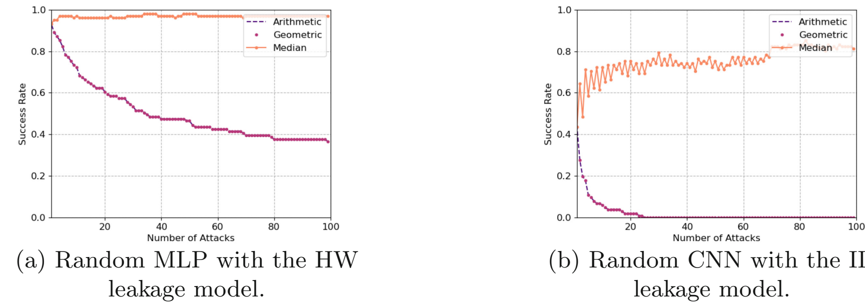
(a) Random MLP with the HW leakage model.

(b) Random MLP with the ID leakage model.

(c) Random CNN with the HW leakage model.

(d) Random CNN with the ID leakage model.

Fig. 6. T_{GE0}: attack on ASCAD_R with random MLP and CNN models.



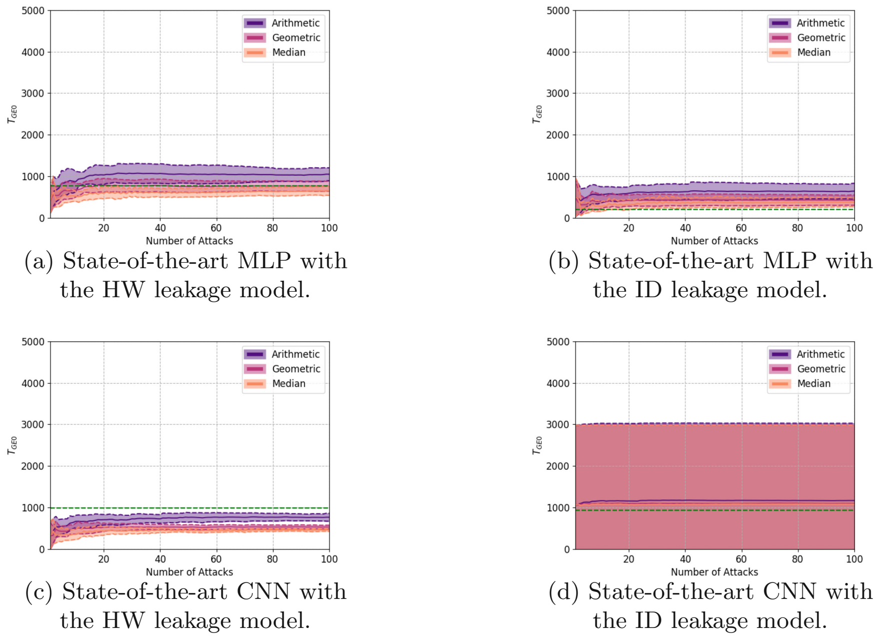
(a) Random MLP with the HW leakage model.

(b) Random CNN with the ID leakage model.

Fig. 7. Success Rate: attack on ASCAD_R with random MLP and CNN models.

multiple profiling models as we effectively develop an ensemble. An interesting perspective is that we can improve state-of-the-art architectures' results by making ensembles of the same architectures with different trainable parameters. We consider this relevant as it allows easy constructions of ensembles based on the available architectures from the literature.

Results for the CHES_CTF Dataset. Note that CHES_CTF with the ID leakage model results in attack failure according to [22,28], so we consider only the HW



(a) State-of-the-art MLP with the HW leakage model.

(b) State-of-the-art MLP with the ID leakage model.

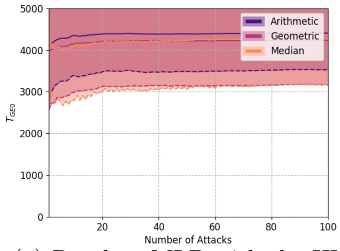
(c) State-of-the-art CNN with the HW leakage model.

(d) State-of-the-art CNN with the ID leakage model.

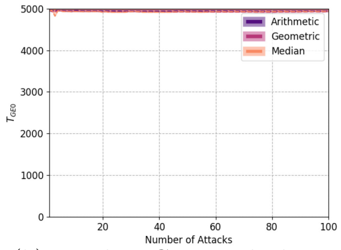
Fig. 8. T_{GE0}: attack on state-of-the-art MLP and CNN models with the ASCAD_R dataset.

leakage model. The results from random model attacks are shown in Fig. 9. The performance of the median and the geometric mean is similar, and both of them outperform the arithmetic mean that is commonly used by researchers and evaluators. The random CNNs show unsuccessful attacks, which means that the random selection of profiling architectures is not appropriate for this dataset. The number of attacks does not show a difference if using more than 40 attacks, and the deviation for MLP is large, as many profiling models do not succeed in breaking the target.

When attacking with state-of-the-art profiling models, the attack efficiency is dramatically improved. As shown in Fig. 10, for both MLP and CNN, the median performs better than the geometric and arithmetic means. Therefore, we can conclude that the median should be the preferred way of calculating GE. Comparing our results and [22,28] (green dashed line), the latter performs significantly better. As mentioned before, since 20-model averaging compensates for the effect of the random weight initialization, we believe that our results reflect the real performance compared to the results reported in related works. A large deviation value additionally confirms those observations. Aligned with all previous cases, we do not see a significant impact of the number of attacks.

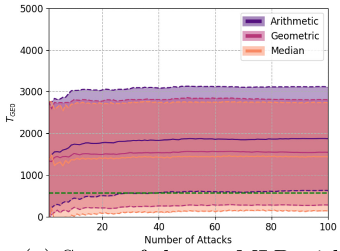


(a) Random MLP with the HW leakage model.

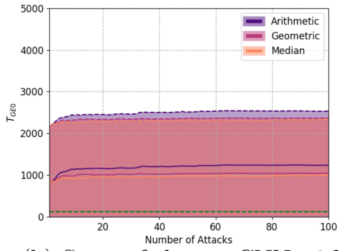


(b) Random CNN with the HW leakage model (most of the attacks failed to converge).

Fig. 9. T_{GE0}: attack on CHES_CTF with random MLP and CNN models.



(a) State-of-the-art MLP with the HW leakage model.



(b) State-of-the-art CNN with the HW leakage model.

Fig. 10. T_{GE0}: attack on state-of-the-art MLP and CNN models with the CHES_CTF dataset.

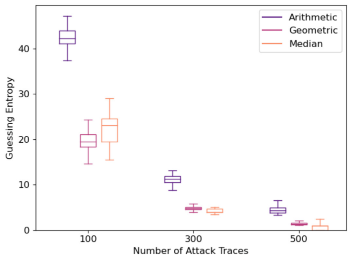
5.3 Discussion

Based on the experimental results, we provide several general observations:

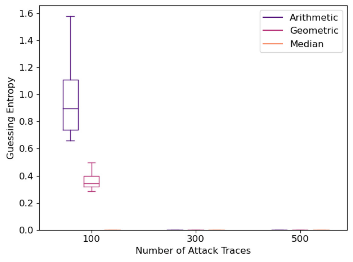
1. Deep learning-based SCA can show different attack results due to algorithmic randomness and skewed distribution of attack results. This, in turn, makes the proper attack assessment potentially difficult, requiring the usage of summary statistics when reporting the attack performance. Naturally, if the number of models that do not converge is significantly larger than the number of converging models, even the median will indicate poor attack performance. Still, we do not consider this a problem as in such cases, the attack is difficult, and the attack performance is generally poor.
2. Arithmetic mean should not be used as the average attack performance estimate as it suffers from a skewed distribution. Our experiments show that the median is the best choice since it is not affected by outliers and thus represents a resistant measure of a center.

3. Large number of independent experiments to average the attack performance does not increase the stability of results, indicating this as a simple option to speed up the evaluation process. According to our results, the averaged results from already 40 attacks are stable and representative in all cases.
4. Large standard deviation with random models is expected as we use (radically) different profiling models. For state-of-the-art models, a large standard deviation indicates the low stability of the model. Thus, the performance of such models could be questionable when facing challenges from the real-world such as devices' portability [2].
5. In many research works, the attack performance is presented for an optimized model (regardless of the technique to achieve it) with specific hyperparameters. However, even for a fixed model, we emphasize the necessity of reporting the averaged performance over a number of profiling models with different weight initialization so that the actual attack performance can be reliably estimated.
6. It is possible to build strong attacks by using ensembles where we use different profiling models (as done in related works) and by using a single model trained a number of times (thus, having different trainable parameters).

We note that the median is well-known to be the preferable metric if a dataset contains outliers or the underlying distribution is skewed. Thus, it could be stated that the results are not surprising. While we agree, we emphasize that related works do not commonly consider or report the media or standard deviation results. Additionally, since the results show that algorithmic randomness plays a significant role, extending the discussion outside of metrics and including appropriate representations is possible. For instance, instead of showing line plots as commonly done in the SCA community, a better option could be to use boxplots. A boxplot provides the minimum, the maximum, the sample median, and the first and third quartiles, allowing better representation for spread and skewness. At the same time, with boxplots, it would be less straightforward to provide results for many values on the x-axis. As a demonstration, we attack ASCAD_F with the HW and ID leakage models 20 times and compare the boxplot of three averaging methods with different numbers of attack traces. As shown in Fig. 11, median averaging performs the best compared to other averaging methods. For Fig. 11b, the results that are not visible indicate the attack reached GE of 0, and there is no variance.



(a) State-of-the-art CNN with the HW leakage model.



(b) State-of-the-art CNN with the ID leakage model.

Fig. 11. Guessing entropy in a boxplot representation: attack on state-of-the-art CNN models with the ASCAD_F dataset.

6 Conclusions and Future Work

This paper investigates the difficulty of assessing the attack performance for deep learning-based side-channel analysis. By doing so, we also provide a way to assess if selected random hyperparameters are well-selected (i.e., they result in models where GE converges). We experimentally show that the most appropriate summary statistics for evaluating deep learning-based SCA is the median and not the arithmetic mean as commonly used. We show that the number of attacks (independent experiments) plays only a marginal role where it is enough to use a small number of attacks (e.g., around 40 independent attacks) to assess the attack performance properly. Naturally, this holds under the assumption that the ranges for random search are optimized. Next, we demonstrate that algorithmic randomness has a significant effect on the results, and to properly assess them, it is necessary to show averaged results and not only a single one (as commonly done). Thus, while it is common to run multiple experiments to account the data randomness (e.g., averaging with guessing entropy), algorithmic randomness also plays an important role (possibly, even being more important), and the results should be reported in such a way to account for it, e.g., using the median over a number of independent training phases.

This paper dealt only with algorithmic randomness. It would be relevant to consider dataset randomness and use more summary statistics. For instance, while reporting average results over multiple experiments is common, no other summary statistics are reported. We consider reporting standard deviation a good option. Indeed, when comparing several deep learning algorithms, one can often see rather similar results. Nevertheless, the question is how stable those results are and if such additional information can help us judge what algorithm performs better. Finally, comparing the results for line plots (as commonly used) and boxplots when depicting the GE results would be interesting.

Acknowledgements. This work was supported in part by the Netherlands Organization for Scientific Research NWO project DISTANT (CS.019) and project PROACT

(NWA.1215.18.014). The authors thank our anonymous reviewers and our shepherd, Michael Pehl, for their valuable comments and suggestions.

References

1. Benadjila, R., Prouff, E., Strullu, R., Cagli, E., Dumas, C.: Deep learning for side-channel analysis and introduction to ASCAD database. J. Cryptograph. Eng. **10**(2), 163–188 (2020). 10.1007/s13389-019-00220-8, https://doi.org/10.1007/s13389-019-00220-8
2. Bhasin, S., Chattopadhyay, A., Heuser, A., Jap, D., Picek, S., Shrivastwa, R.R.: Mind the portability: a warriors guide through realistic profiled side-channel analysis. In: 27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, 23–26 February 2020. The Internet Society (2020). https://www.ndss-symposium.org/ndss-paper/mind-the-portability-a-warriors-guide-through-realistic-profiled-side-channel-analysis/
3. Cagli, E., Dumas, C., Prouff, E.: Convolutional neural networks with data augmentation against jitter-based countermeasures. In: Fischer, W., Homma, N. (eds.) CHES 2017. LNCS, vol. 10529, pp. 45–68. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66787-4_3
4. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Kaliski, B.S., Koç, K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36400-5_3
5. Heuser, A., Picek, S., Guilley, S., Mentens, N.: Side-channel analysis of lightweight ciphers: does lightweight equal easy? In: Hancke, G.P., Markantonakis, K. (eds.) Radio Frequency Identification and IoT Security - 12th International Workshop, RFIDSec 2016, Hong Kong, China, November 30–December 2, 2016, Revised Selected Papers, LNCS, vol. 10155, pp. 91–104. Springer, Berlin (2016). https://doi.org/10.1007/978-3-319-62024-4_7
6. Kim, J., Picek, S., Heuser, A., Bhasin, S., Hanjalic, A.: Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. In: IACR Trans. Cryptogr. Hardw. Embed. Syst. **2019**, 148–179 (2019)
7. Lerman, L., Medeiros, S.F., Bontempi, G., Markowitch, O.: A machine learning approach against a masked AES. In: CARDIS, LNCS, Springer, Berlin (2015). https://doi.org/10.1007/s13389-014-0089-3
8. Li, H., Krček, M., Perin, G.: A comparison of weight initializers in deep learning-based side-channel analysis. In: Zhou, I., et al. (eds.) Applied Cryptography and Network Security Workshops, pp. 126–143. Springer International Publishing, Cham (2020)
9. Lu, X., Zhang, C., Cao, P., Gu, D., Lu, H.: Pay attention to raw traces: a deep learning architecture for end-to-end profiling attacks. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2021**(3), 235–274 (2021). 10.46586/tches.v2021.i3.235-274, https://tches.iacr.org/index.php/TCHES/article/view/8974
10. Maghrebi, H., Portigliatti, T., Prouff, E.: Breaking cryptographic implementations using deep learning techniques. In: Carlet, C., Hasan, M.A., Saraswat, V. (eds.) SPACE 2016. LNCS, vol. 10076, pp. 3–26. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49445-6_1
11. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. Springer, Boston (December 2006). https://doi.org/10.1007/978-0-387-38162-6I, SBN 0-387-30857-1, http://www.dpabook.org/

12. Martin, D.P., Mather, L., Oswald, E., Stam, M.: Characterisation and estimation of the key rank distribution in the context of side channel evaluations. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10031, pp. 548–572. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_20
13. Perin, G., Chmielewski, L., Picek, S.: Strength in numbers: Improving generalization with ensembles in machine learning-based profiled side-channel analysis. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**(4), 337–364 (2020). https://doi.org/10.13154/tches.v2020.i4.337-364, https://tches.iacr.org/index.php/TCHES/article/view/8686
14. Perin, G., Picek, S.: On the influence of optimizers in deep learning-based side-channel analysis. In: Dunkelman, O., Jacobson, Jr., M.J., O'Flynn, C. (eds.) SAC 2020. LNCS, vol. 12804, pp. 615–636. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81652-0_24
15. Perin, G., Wu, L., Picek, S.: Exploring feature selection scenarios for deep learning-based side-channel analysis. Cryptology ePrint Archive, Report 2021/1414 (2021). https://ia.cr/2021/1414
16. Picek, S., Heuser, A., Jovic, A., Batina, L.: A systematic evaluation of profiling through focused feature selection. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **27**(12), 2802–2815 (2019)
17. Picek, S., Heuser, A., Guilley, S.: Template attack versus bayes classifier. J. Cryptogr. Eng. **7**(4), 343–351 (2017). https://doi.org/10.1007/s13389-017-0172-7
18. Picek, S., Heuser, A., Jovic, A., Bhasin, S., Regazzoni, F.: 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 (2018). https://doi.org/10.13154/tches.v2019.i1.209-237, https://tches.iacr.org/index.php/TCHES/article/view/7339
19. Picek, S., et al.: Side-channel analysis and machine learning: a practical perspective. In: 2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, 14–19 May 2017, pp. 4095–4102 (2017)
20. Picek, S., Heuser, A., Wu, L., Alippi, C., Regazzoni, F.: When theory meets practice: a framework for robust profiled side-channel analysis. Cryptology ePrint Archive, Report 2018/1123 (2018). https://eprint.iacr.org/2018/1123
21. Picek, S., Perin, G., Mariot, L., Wu, L., Batina, L.: Sok: Deep learning-based physical side-channel analysis. Cryptology ePrint Archive, Report 2021/1092 (2021). https://ia.cr/2021/1092
22. Rijsdijk, J., Wu, L., Perin, G., Picek, S.: Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2021**(3), 677–707 (2021). https://doi.org/10.46586/tches.v2021.i3.677-707, https://tches.iacr.org/index.php/TCHES/article/view/8989
23. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 30–46. Springer, Heidelberg (2005). https://doi.org/10.1007/11545262_3
24. Smith, L.N.: Cyclical learning rates for training neural networks. In: 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 464–472. IEEE (2017)
25. Standaert, F.-X., Malkin, T.G., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 443–461. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01001-9_26

26. Whitnall, C., Oswald, E.: Robust profiling for DPA-style attacks. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 3–21. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48324-4_1
27. Wouters, L., Arribas, V., Gierlichs, B., Preneel, B.: Revisiting a methodology for efficient CNN architectures in profiling attacks. IACR Trans. Cryptogr. Hardw. Embed. Syst. 2020(3), 147–168 (2020). https://doi.org/10.13154/tches.v2020.i3.147-168, https://tches.iacr.org/index.php/TCHES/article/view/8586
28. Wu, L., Perin, G., Picek, S.: I choose you: automated hyperparameter tuning for deep learning-based side-channel analysis. IACR Cryptol. ePrint Arch. **2020**, 1293 (2020)
29. Wu, L., et al.: On the attack evaluation and the generalization ability in profiling side-channel analysis. Cryptology ePrint Archive, Report 2020/899 (2020). https://eprint.iacr.org/2020/899
30. Zaid, G., Bossuet, L., Habrard, A., Venelli, A.: Methodology for efficient CNN architectures in profiling attacks. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**(1), 1–36 (2019). https://doi.org/10.13154/tches.v2020.i1.1-36, https://tches.iacr.org/index.php/TCHES/article/view/8391