The Need for Speed: A Fast Guessing Entropy Calculation for Deep Learning-based SCA

Guilherme Perin
Delft University of Technology
The Netherlands

Lichao Wu
Delft University of Technology
The Netherlands

Stjepan Picek
Radboud University
The Netherlands

***Abstract*—In recent years, the adoption of deep learning drastically improved profiling side-channel attacks (SCA). Although guessing entropy is a highly informative metric for profiling SCA, it is time-consuming, especially if computed for all epochs during training. This paper shows that guessing entropy can be efficiently computed during training by reducing the number of validation traces. Our solution significantly speeds up the process, impacting hyperparameter search and profiling attack performances. Our fast guessing entropy calculation is up to 16× faster and results in more hyperparameter tuning experiments, allowing us to find more efficient deep learning models.**

***Index Terms*—Side-channel Analysis, Deep learning, Guessing entropy, Validation phase, Fast Guessing Entropy**

I. Introduction

Side-channel attacks (SCA) explore the unintentional leakages (such as power consumption, time, and electromagnetic emissions) from electronic devices running secret-sensitive operations such as cryptographic algorithms. Profiling SCA, one of the most popular attack methods, is widely considered by developers and manufacturers to assess worst-case security under the strongest adversary assumptions. This type of attack assumes an adversary has a clone (open) device to build the strongest possible probabilistic model from collected side-channel information. This way, the adversary applies the model to the victim's devices to recover the secret. This attack phase usually requires fewer side-channel measurements.

Template attacks are the most classic form of profiling SCA [1]. Template attacks represent the strongest profiling model theoretically because of their typical underlying statistical distribution of side-channel leakages following multivariate Gaussian (or normal) distributions. Machine learning methods (e.g., random forest or support vector machines) also provide strong profiling models, while their statistical parameters are learned from side-channel measurements rather than directly computed. Both Gaussian templates and machine learning models require feature selection, which is usually only possible by skipping black-box assumptions. Indeed, an effective feature selection requires a strong correlation between the leakages and processed intermediate data. For instance, for protected cryptographic designs with masking countermeasures, the selected features would be valid by knowing the secret random masks. Then, one can deploy worst-case security evaluations to emulate the strongest adversary. Additionally, template and machine learning-based models are susceptible to desynchronization effects in side-channel measurements, thus bringing additional challenges.

In recent years, the adoption of deep neural networks (DNNs) for profiling SCA provided superior results compared to template attacks and classical machine learning-based methods, especially against AES implementations [2], [3]. Without feature selection (which implies considering a weaker adversary), deep learning-based SCA can break cryptographic implementations protected with different countermeasures, such as Boolean masking and timing desynchronization. Their high complexity follows the high learning capacity of DNNs; the expensive hyperparameter tuning becomes a limitation to fully explore the real potential of deep learning to find vulnerabilities in software and hardware implementations. Indeed, smaller DNNs may limit the learning capacity of a model, underfitting the profiling side-channel traces and providing poor attack performance. On the other hand, adding too many layers results in larger models but can easily overfit, thus reducing the possibility of obtaining the secret information. One straightforward way to avoid this problem is by allowing larger models to be trained with regularization, restricting the model's capacity during training. Dropout, weight decay, and data augmentation are well-known methods for regularization, but their indirect influence on the attack performance makes newly introduced hyperparameters difficult to tune. Early stopping is a very efficient regularization mechanism that returns the model when the training reaches the best generalization moment. This is done by monitoring validation metrics, such as accuracy or loss. Unfortunately, in profiling SCA, collected leakages are normally extremely noisy because of environmental noise and implemented countermeasures. Therefore, as demonstrated in [4], validation accuracy and loss are inconsistent with SCA performance. Although the model can be optimized through gradient descent by minimizing generic loss functions (such as categorical cross-entropy or negative log-likelihood) and not an SCA metric, the calculation of guessing entropy (GE) from a set of validation traces is consistent and highly informative concerning the profiling model generalization in SCA. The application of empirical GE as an early stopping metric provides significant overheads, rendering the hyperparameter tuning process very slow and, in some cases, impractical.

We propose a faster guessing entropy calculation by reducing the number of validation traces and still achieving

highly efficient results for early stopping. We compare our fast GE method (denoted FGE) with state-of-the-art metrics for early stopping and guessing entropy. We show that FGE estimation is highly competitive and provides superior results with a neglecting time overhead in all scenarios. With FGE, hyperparameter tuning becomes more efficient, increasing the chances to select an optimal model with less time.

II. Background

A. Deep learning-based SCA

Profiling SCAs consider the strongest adversary with access to a clone device running the target cryptographic algorithm. The adversary can query the clone device with any set of plaintext $P = (p_0, p_1, \ldots, p_{N-1})$ and chosen keys $K = (K_0, K_1, \ldots, K_{N-1})$, and measure side-channel traces $X = (x_0, x_1, \ldots, x_{N-1})$. These traces are used for training the classification algorithm (i.e., to build a machine learning model). This phase is known as the training or profiling phase. Also, during this phase, a validation set containing V traces is selected from the profiling set to validate the model. Next, the adversary takes measurements on the target device, where few traces are captured with known input. The previously trained model is then exploited to recover the secret key used in the target device. This phase is known as the attack or test phase.

The training process has as the main goal the minimization of the selected loss function. In this paper, we consider the *categorical cross-entropy* (CCE) as the loss function. As demonstrated in [4], due to the imbalanced dataset problem, validation loss function values (including CCE) are inconsistent with SCA metrics, which is also the case of SCA-based loss functions, as already proposed in [5], [6]. Therefore, we must select a more efficient validation metric to assess the model's performance for SCA.

Metrics like guessing entropy (GE) [7] are commonly used by an adversary to estimate the required effort to obtain the key. A side-channel attack outputs a key guessing vector $\mathbf{g} = [g_1, g_2, \ldots, g_{|\mathcal{K}|}]$ in decreasing order of probability, i.e., g_1 represents the most likely key candidate and $g_{|\mathcal{K}|}$ the least likely key candidate. Guessing entropy is the average position of k^* in $\mathbf{g}$. Commonly, the averaged value is calculated over multiple independent experiments to obtain statistically significant results. In this paper, this GE method is called empirical GE, and it is evaluated on a set of V validation traces, where the results of multiple key rank executions are averaged and performed on a partition Q from V.

B. Datasets

We evaluate ASCAD datasets [8] which contain electromagnetic side-channel measurements collected from the first-order protected software implementations of AES-128 running on an 8-bit AVR microcontroller [9]. There are two versions of the ASCAD dataset. The first version, ASCADf, has a fixed key and 60 000 traces in total. We split the dataset into 50 000, 5 000, and 5 000 for profiling, validation, and attack sets, respectively. The second version of the ASCAD dataset, ASCADr, has fixed and random keys, and it consists of 300 000 traces. In this case, we consider 200 000 for profiling (with random keys), 10 000 for validation, and 10 000 for the attack set. Both validation and attack sets have a fixed key. For both versions, we attack the third key byte (which is the first masked byte) by using the trimmed intervals already extracted and released in [8]. Thus, we use a pre-selected window of 700 features for ASCADf, while for ASCADr, the window size equals 1 400 features. For all experiments, the datasets are labeled according to the leakage model from the third Sbox output byte in the first AES encryption round, i.e., $S(p_i \oplus k_i)$ and $HW(S(p_i \oplus k_i))$ for the Identity and Hamming weight leakage models, respectively.

III. Related Works

Due to the expensive trial-and-error cost in the profiling phase, enhancing performance in DL-based profiling SCA is a challenging task. In recent years, the SCA community developed two main alternatives to improve the attack efficiency: (1) by defining small neural network models that are faster to train and easier to tune [2], [3] and (2) by reducing the number of the required profiling traces during training [10]. Both solutions can have severe impacts on attack or generalization performance. The first approach may result in models that underfit for more noisy leakages or leakages obtained from other devices (portability problem). The second alternative speeds up the process; still, it may result in limited learnability due to the eventually low number of profiling traces.

Besides the methods mentioned above, a third alternative uses efficient and reliable validation metrics to evaluate training and, consequently, implement faster hyperparameter tuning (which can provide faster convergence) with larger models and larger profiling sets. Empirical GE (described in Section II-A) can be very expensive to compute with larger validation sets, especially if used during training to detect the best training epoch. In a recent publication, Zhang et al. [5] proposed Guessing Entropy Estimation Algorithm (GEEA) to reduce the computational limitation cost of empirical GE for the full attacked key scenarios, which computes faster than empirical GE calculation on separate key bytes. Indeed, empirical GE executes multiple key rank executions over multiple partitions of the dataset V, each partition containing Q measurements. GEEA, on the other hand, only requires one execution over the Q measurements.

Alternative solutions were proposed as new validation metrics for early stopping, which can stop training sooner and speed up the process. In [11], the authors considered mutual information approach between the output probabilities and validation labels to monitor the best epoch during training. The work of [12] monitors the epoch when the training achieves the minimal difference between the number of profiling and validation traces that are required to achieve 90% of success rate. The authors proposed a routine to abort training if this difference keeps increasing after reaching its minimum value. In our work, we also consider the mutual information metric for comparison. The method proposed in [12] is not considered in our comparative analysis as it is directly adapted

to datasets with fixed keys in the profiling set, which is not the case of ASCADr. The method requires estimating the number of traces to reach a success rate of 90%, which implies obtaining the evolution of success rate concerning the number of validation traces. This means that the success rate is computed Q times for each epoch, adding significant time overhead to the process.

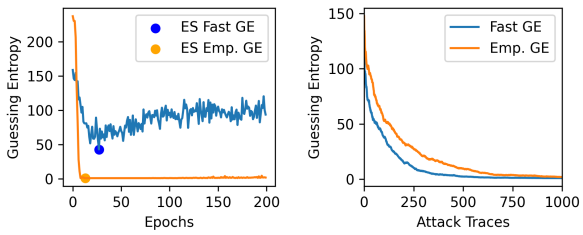
As we can see, none of the mentioned approaches compute GE directly from the validation traces at the end of each training epoch. GEEA was proposed as a fast and more stable GE estimation, but it is not suggested to be used during network training. On the other hand, although GE can be a potential metric candidate, its computation could be very slow if more validation traces are considered (which is required for GE stability), finally providing significant overheads to the training process. Therefore, the SCA community did not consider directly applying GE (including GEEA) as the early stopping metric, especially in the hyperparameter search processes. This work shows that significantly reducing the number of validation traces for GE estimation during training is a reliable and efficient metric for early stopping, benefiting hyperparameter tuning optimization.

IV. Fast GE for Early Stopping

Running hyperparameter search without pre-selecting efficient ranges for each hyperparameter may fail to find powerful attack models. A solution could be searching for small models with restricted search ranges, as proposed in [2], [13] or by setting the objective of the search as being a small model, as proposed by Rijsdijk et al. in [3]. Still, small models suffer from limited fitting capacity, which is particularly problematic for noise and protected targets. An alternative is to allow larger models and add regularization to prevent overfitting [14]. Although regularization tends to improve model generalization, regularized models with increased size require more training epochs, reducing the efficiency in a hyperparameter search process.

We propose a fast GE (FGE) calculation during training that reduces the overhead due to GE up to 16×. When used as an early stopping metric, the FGE metric provides very small overheads to the training process, usually between 1.5% and 3.3%, while, e.g., empirical GE shows overheads between 18.59% and 28.19%, as reported in Section V. Our idea consists in reducing the number of validation traces when using GE as a metric, which has multiple benefits in DL-based SCA.

If the model converges, the attack is successful, and the GE for a reduced number of validation traces can also indicate the best epoch to stop training efficiently. Using too many validation traces for the metric calculation may obscure the real performance of the model: a model that overfits may also slowly decrease guessing entropy to 1 after processing enough validation traces. In contrast, FGE is more sensitive to the model's performance change, thanks to its low usage of the validation traces. This situation is illustrated in Figure 1. As we can see in Figure 1a, when using partitions of $Q = 3\,000$ validation traces for empirical GE, the best epoch will be returned at the moment when GE is equal to 1. If the next epochs indicate a model that requires fewer attack traces to succeed (which means better generalization), this empirical GE will not capture that. On the other hand, using fewer traces allows us to get this convergence and, as a consequence, be able to recover the key with fewer attack traces, as shown in Figure 1b. Of course, the question here is: why would this be a problem if reaching GE of 1 also allows an adversary to recover the key? We can observe two main problems in this scenario: first, empirical GE with more traces provides more overheads and limits the number of hyperparameter search attempts, preventing us from finding a model that eventually breaks the target (which is the example of model found in Figure 1). Second, from the current model, we would select trained parameters before it reaches its best attack performance or generalization capacity, which can also indicate overfitting on the validation set, possibly opening issues in portability scenarios (when the device used for profiling is different from the device used for attack [15]). If a model generalizes, then GE will eventually decrease. Therefore, the final GE value with reduced validation traces should be correlated to GE with more traces.



(a) GE vs epochs. (b) GE vs attack traces.

Fig. 1: Fast GE vs Empirical GE (ES = Early Stopping).

V. Experimental Results

A. Hyperparameter Search Ranges

In this work, we only consider convolutional neural networks (CNNs) as they contain a large number of hyperparameters to tune and, therefore, it becomes more challenging to find good hyperparameter combinations in comparison to, e.g., multilayer perceptrons. Table I provides the selected ranges for the hyperparameter tuning processes. These selected ranges result in a search space containing 2.7×10^9 possible combinations. As we can see, we allow CNNs to contain up to eight hidden layers, combining convolution and dense layers. Each convolution layer is always followed by a pooling layer. As the ASCADf and ASCADr datasets contain 50 000 and 200 000 profiling traces, respectively, larger models would run out of GPU memory, and in most of the cases, become too complex for the considered problem.

TABLE I: Hyperparameter search space for CNNs (c in convolution filters indicates the convolution layer index).

Hyperparameter	Ranges		
	Min	**Max**	**Step**
Batch Size	100	1 000	100
Convolution Layers	1	4	1
Convolution Filters	$2 \times 2^{c-1}$	$16 \times 2^{c-1}$	2
Kernel Size	4	20	1
Stride	1	4	1
Pooling Size	1	4	1
Pooling Stride	1	4	1
Dense layers	1	4	1
	Options		
Neurons	10, 20, 30, 40, 50, 100, 200, 300, 400, 500		
Activation function	ReLU, ELU, or SELU		
Learning Rate	5e-3, 1e-3, 5e-4, 1e-4, 5e-5, 1e-5		
Pooling Type	Average, Max		
Weight Initializer	He, Random Uniform, Glorot Uniform		
Optimizer	Adam, RMSprop		

B. Random Hyperparameter Search with Different Validation Metrics

We compare different early stopping in a random hyperparameter search process for the two ASCAD datasets and different leakage models. Each randomly selected CNN is trained for 200 epochs, and we save the trained weights at the end of each epoch. At the end of the training, each early stopping metric indicates what the best training epoch was, and we restore the trained weights from that epoch in order to compute GE for the attack set. Note that 200 epochs is a relatively small number for training epochs, and, as shown in this section, stopping the training for 200 epochs also delivers good results.

Table II gives the number of validation traces V considered for each metric, while the partition amount Q is the number of the traces used to calculate each specific metric. Note that we set V greater than Q so that the sampling of each data in Q preserves certain randomness. By doing so, the obtained results would have more generality. For mutual information, we apply V validation traces. FGE estimation considers only 50 traces for Q and 500 traces for V. We tested several numbers for Q and V, and this was the minimum value for Q and V that still preserves the best results for FGE.

TABLE II: Number of validation traces for each early stopping method.

	ASCADf		**ASCADr**	
	V	Q	V	Q
Fast GE (this work)	500	50	500	50
Empirical GE	5 000	3 000	10 000	5 000
GEEA	5 000	3 000	10 000	5 000
Mutual Information	5 000	5 000	10 000	10 000

We execute 500 searches for each dataset and leakage model. Table III provides the average time overhead in percentage for each considered metric. As we can see, the FGE estimation provides a maximum of 3.35% overhead among the four considered scenarios. For the ASCADr dataset, the overhead is only 1.19% and 1.49%, which can be considered negligible for the training time compared with its counterparts. As expected, empirical GE and GEEA methods provide the largest overheads, although GEEA is faster than empirical GE. The mutual information method provides the second-best results, which is related to the more straightforward calculation than guessing entropy.

TABLE III: Average time overhead of different early stopping methods.

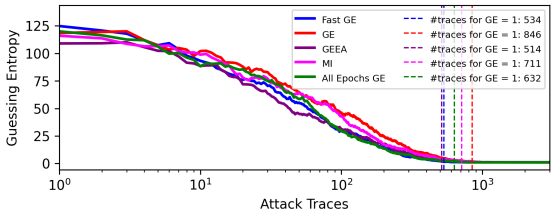
	ASCADf		**ASCADr**	
	HW	**Identity**	**HW**	**Identity**
Fast GE (this work)	2.66%	3.35%	1.19%	1.49%
Empirical GE	20.70%	28.19%	18.59%	24.21%
GEEA	11.48%	23.95%	9.31%	20.17%
Mutual Information	9.28%	7.46%	7.81%	6.30%

Table IV provides the % that each metric can select a generalizing model with early stopping (DNN that reaches GE=1 in the attack phase) from the random search. Together with GEEA, the fast GE is a highly efficient metric (top two performance in all considered scenarios). Most importantly, we successfully verify that FGE is always superior to the situation where no early stopping is used (200 epochs in the table) and with neglecting overhead.

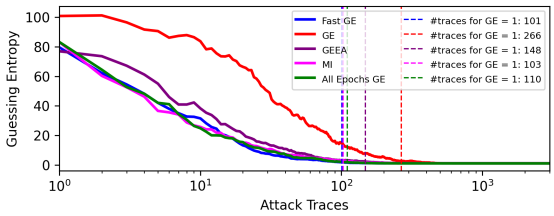
TABLE IV: % of times a generalizing DNN was selected from each metric and from the training with all 200 epochs.

	ASCADf		**ASCADr**	
	HW	**Identity**	**HW**	**Identity**
Fast GE (this work)	56.52%	43.46%	49.63%	34.13%
Empirical GE	59.66%	43.16%	50.00%	33.23%
GEEA	59.66%	43.46%	54.34%	29.30%
Mutual Information	50.74%	37.38%	43.84%	33.53%
200 epochs	49.75%	40.42%	45.83%	32.62%

Figure 2 shows results for the ASCADf dataset. When side-channel traces are labeled according to the Hamming weight leakage model, the correct key is recovered with 514 traces for GEEA metric and 534 traces (the second best) with FGE early stopping metric. In the case of the Identity leakage model, the best results are achieved for the FGE metric, where 101 attack traces are needed to achieve GE equal to 1, which is completely aligned with state-of-the-art results [2], [3], [13]. The good-performing results from the mutual information metric and the GE obtained with 200 epochs indicate the effectiveness of early-stopping metrics in preventing the best-obtained model from overfitting. Again, we confirm that FGE is highly competitive in both leakage models and requires 10× fewer validation traces.

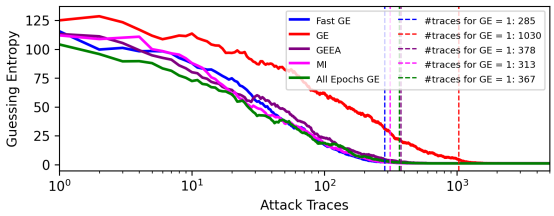


(a) Hamming weight leakage model.

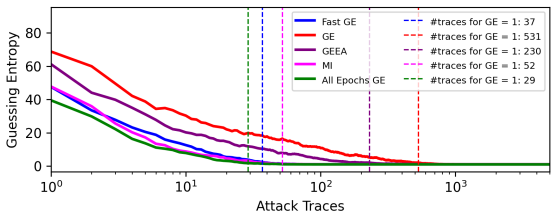


(b) Identity leakage model.

Fig. 2: GE results from best models selected from different early stopping metrics for ASCADf dataset.



(a) Hamming weight leakage model.

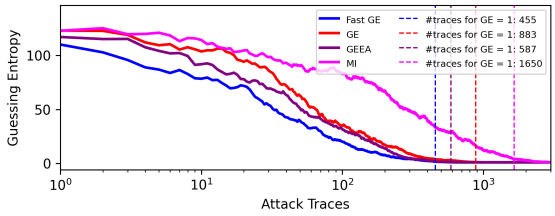


(b) Identity leakage model.

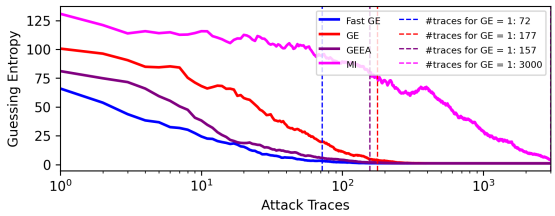
Fig. 3: GE results from best models selected from different early stopping metrics for ASCADr dataset.

For ASCADr dataset, results for FGE are also very promising, as shown in Figure 3. For the Hamming weight leakage model, FGE provides the best results, followed by mutual information metric. In the case of the Identity leakage model, the best result is obtained with all 200 epochs, showing that this number of epochs is appropriate for this best model found through random search. When early stopping is considered, the best results are obtained with the FGE metric.

Furthermore, the performance of best models selected from empirical GE as an early stopping metric provided less efficient results. As already mentioned in [5], empirical GE requires a very large validation set N, and a more stable GE estimation can be obtained by the selection of an increased number of partitions Q. This directly implies that empirical GE, although convenient for GE estimation for larger N at the end of the profiling phase, is inefficient as an early stopping metric while also having a significant time overhead.



(a) Hamming weight leakage model.



(b) Identity leakage model.

Fig. 4: GE results from best models found with BO with different early stopping metrics for ASCADf dataset.

C. Hyperparameter Tuning with Different Validation Metrics

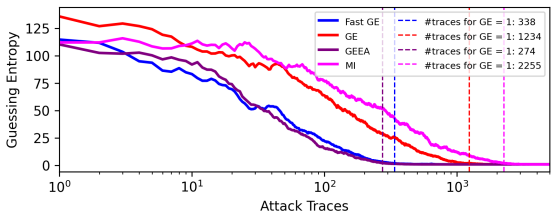
This section analyzes how the evaluated early stopping metrics perform with Bayesian optimization (BO) for hyperparameter search [16]. For that, we consider the open-source **BayesianOptimization** method provided in *keras-tuner* [17] Python package. We run BO for 100 searches with ASCAD datasets and the Hamming weight and Identity leakage models. We repeat each search process five times for each different early stopping metric. Results for all epochs (as indicated in the previous section) are omitted because keras-tuner requires a validation metric computed for each training epoch. The results reported in this section are extracted from the best-found model out of the five search attempts.

Results from BO for ASCADf dataset are shown in Figure 4. The best results are obtained from FGE for both Hamming weight and Identity leakage models. In particular, for the Identity leakage model, as shown in Figure 4b, the best found model achieves GE equal to 1 with less than half of the attack traces needed for GEEA. In these experiments, mutual information provides less efficient results.

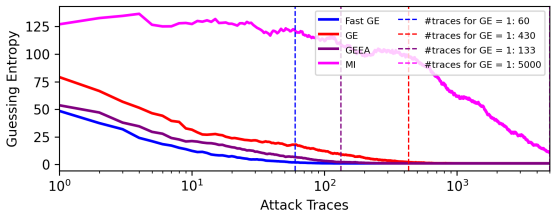
Figure 5 provides BO results for ASCADr dataset. For the Hamming leakage model, GEEA and FGE provide the best results. For the Identity leakage model, results for FGE are superior, and only 60 attack traces are required for key byte recovery, while empirical GE requires $10\times$ more attack traces to succeed. Again, the mutual information metric delivers the worst results.

D. State-Of-The-Art Models with Different Validation Metrics

Here, we provide attack results when applying early stopping to three different CNN architectures considered state-of-the-art for the ASCADf dataset. As the results for these CNN

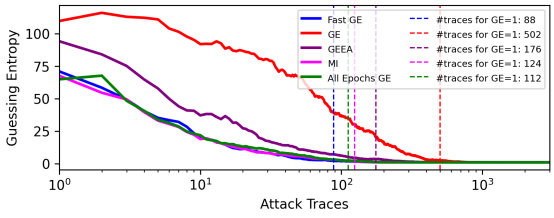


(a) Hamming weight leakage model.

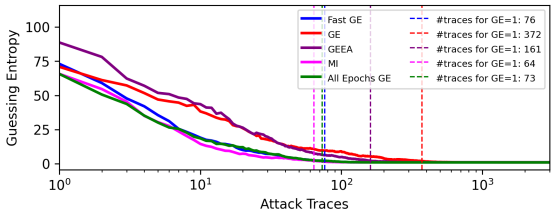


(b) Identity leakage model.

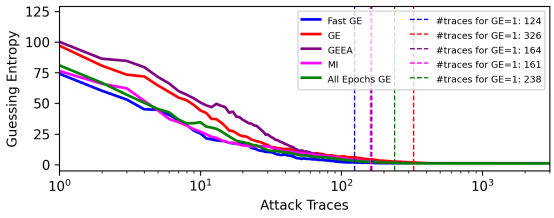
Fig. 5: GE results from best models found with BO with different early stopping metrics for ASCADr dataset.



(a) CNN from [2]



(b) CNN from [13]



(c) Best CNN from [3]

Fig. 6: Performance of different validation metrics on state-of-the-art CNN architectures.

were reported for the Identity leakage model, we only consider this scenario in our analysis. As shown in Figure 6, for CNN models from [2] and [3], our FGE metric provides best results. Results for CNN model from [13] also put FGE among the best-performing metrics.

VI. Conclusions

Hyperparameter tuning is an expensive process in DL-based profiling SCA. To solve this issue, the number of search attempts should be optimized by selecting the best possible validation metric to find a neural network generalizing to different attack sets. We propose using a fast GE metric that significantly reduces the number of validation traces in the GE calculation. Our results indicate that fast GE as a validation metric delivers efficient and competitive early stopping results. Our technique is validated in different scenarios and shows good results with neglecting time overheads. Thus, we consider FGE as the method of choice for practical deep learning-based SCA hyperparameter tuning. In future work, we will explore the efficiency of different validation metrics in portability settings.

References

[1] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Cryptographic Hardware and Embedded Systems - CHES 2002*. Springer Berlin Heidelberg, 2003, pp. 13–28.

[2] G. Zaid, L. Bossuet, A. Habrard, and A. Venelli, "Methodology for efficient CNN architectures in profiling attacks," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2020, no. 1, pp. 1–36, 2020.

[3] J. Rijsdijk, L. Wu, G. Perin, and S. Picek, "Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2021, no. 3, pp. 677–707, 2021.

[4] S. Picek, A. Heuser, A. Jovic, S. Bhasin, and F. Regazzoni, "The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2019, no. 1, pp. 209–237, 2019.

[5] J. Zhang, M. Zheng, J. Nan, H. Hu, and N. Yu, "A novel evaluation metric for deep learning-based side channel analysis and its extended application to imbalanced data," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2020, no. 3, pp. 73–96, 2020.

[6] G. Zaid, L. Bossuet, F. Dassance, A. Habrard, and A. Venelli, "Ranking loss: Maximizing the success rate in deep learning side-channel analysis," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2021, no. 1, pp. 25–55, 2021.

[7] F. X. Standaert, T. G. Malkin, and M. Yung, "A unified framework for the analysis of side-channel key recovery attacks," *Lecture Notes in Computer Science*, vol. 5479 LNCS, pp. 443–461, 2009.

[8] "ASCAD GitHub Repository," Website, 2018, https://github.com/ANSSI-FR/ASCAD.

[9] R. Benadjila, E. Prouff, R. Strullu, E. Cagli, and C. Dumas, "Deep learning for side-channel analysis and introduction to ASCAD database," *J. Cryptographic Engineering*, vol. 10, no. 2, pp. 163–188, 2020.

[10] S. Picek, A. Heuser, G. Perin, and S. Guilley, "Profiling side-channel analysis in the efficient attacker framework," Cryptology ePrint Archive, Report 2019/168, 2019.

[11] G. Perin, I. Buhan, and S. Picek, "Learning when to stop: A mutual information approach to prevent overfitting in profiled side-channel analysis," ser. LNCS, vol. 12910. Springer, 2021, pp. 53–81.

[12] D. Robissout, G. Zaid, B. Colombier, L. Bossuet, and A. Habrard, "Online performance evaluation of deep learning networks for profiled side-channel analysis," ser. Lecture Notes in Computer Science, vol. 12244. Springer, 2020, pp. 200–218.

[13] L. Wouters, V. Arribas, B. Gierlichs, and B. Preneel, "Revisiting a methodology for efficient CNN architectures in profiling attacks," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2020, no. 3, pp. 147–168, 2020.

[14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[15] S. Bhasin, A. Chattopadhyay, A. Heuser, D. Jap, S. Picek, and R. R. Shrivastwa, "Mind the portability: A warriors guide through realistic profiled side-channel analysis," in *27th NDSS*, 2020.

[16] L. Wu, G. Perin, and S. Picek, "I choose you: Automated hyperparameter tuning for deep learning-based side-channel analysis," Cryptology ePrint Archive, Report 2020/1293, 2020.
[17] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi *et al.*, "Kerastuner," https://github.com/keras-team/keras-tuner, 2019.