Focus is Key to Success: A Focal Loss Function for Deep Learning-Based Side-Channel Analysis

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

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

[2] Radboud University, Nijmegen, The Netherlands

Abstract. The deep learning-based side-channel analysis represents one of the most powerful side-channel attack approaches. Thanks to its capability in dealing with raw features and countermeasures, it becomes the de facto standard approach for the SCA community. The recent works significantly improved the deep learning-based attacks from various perspectives, like hyperparameter tuning, design guidelines, or custom neural network architecture elements. Still, insufficient attention has been given to the core of the learning process - the loss function.

This paper analyzes the limitations of the existing loss functions and then proposes a novel side-channel analysis-optimized loss function: Focal Loss Ratio (FLR), to cope with the identified drawbacks observed in other loss functions. To validate our design, we 1) conduct a thorough experimental study considering various scenarios (datasets, leakage models, neural network architectures) and 2) compare with other loss functions used in the deep learning-based side-channel analysis (both "traditional" ones and those designed for side-channel analysis). Our results show that FLR loss outperforms other loss functions in various conditions while not having computational overhead like some recent loss function proposals.

Keywords: Deep learning · Focal loss · Loss function · Side-channel analysis

1 Introduction

Side-channel analysis (SCA) is one of the most popular tools to exploit the implementation weaknesses of an algorithm [11]. Commonly, SCA attacks can be divided into two categories: direct attacks and profiling attacks. The first attack method analyzes the leakages from the target device directly, while the second one requires a copy of the target device. There, an attacker would first learn the characteristic of the copied device (profiling phase) and then launch an attack on the target device (attack phase).

With stronger attack assumptions, profiling attacks are considered more powerful than their counterpart. In recent years, the rise of deep learning further

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
J. Balasch and C. O'Flynn (Eds.): COSADE 2022, LNCS 13211, pp. 29–48, 2022.
https://doi.org/10.1007/978-3-030-99766-3_2

increased the profiling attacks' capability. Specifically, such attacks can break targets protected with countermeasures by relaxing the assumptions of knowledge from target implementations [7,25,29]. Moreover, compared with the conventional profiling attack (i.e., template attack [22]) that relies upon points of interest selection, deep learning-based SCA has softer restrictions on data preprocessing/feature engineering.

Deep learning-based SCA still requires a significant effort to reach its full potential. There are many open questions when applying such attacks, such as network architecture design [18,24,29], evaluation metric design [30], as well as the interpretability and explainability of models [26]. Unfortunately, those issues represent only a part of the problem. A perspective that cannot be neglected is that classical machine learning metrics/loss functions do not necessarily give an accurate representation of the performance of an SCA model [16,30]. On the other hand, launching practical attacks and averaging the key rank to estimate the guessing entropy is computationally costly (especially if also done during the training phase, see, e.g., [14,19]). Consequently, the SCA community put a significant attention on developing SCA-specific metrics and loss functions, such as Cross-Entropy Ratio loss (CER) [30] and ranking loss (RKL) [28].

The CER loss is one of the recently developed methods to improve the performance of deep learning models in the SCA domain. When comparing the CER loss and the conventional categorical cross-entropy (CCE) loss, the CER loss introduces a denominator to the CCE loss that calculates the correlation between multiple traces and incorrect labels so that the difference between the target cluster and other clusters can be maximized. Similar implementations are proved to be efficient in the machine learning domain as well [31]. However, CER loss has two limitations. The CER's denominator calculation can be a challenging task since even for the traces that belong to the same cluster, the classification difficulties of each trace can be different. The easily classified traces can significantly increase the denominator's value for CER loss, thus reducing the overall loss. From a higher level, when performing the classification tasks, learning from easy samples is not helpful but could easily trigger model overfitting. The hard samples, on the contrary, help the classifiers learn the underlying data's properties, thus could lead to better classification performance.

Returning to the CER loss, although one can include more traces to increase the possibility of picking up hard samples 1) since the samples are randomly selected, the samples' difficulties are uncertain; 2) including too many samples would significantly slow down the training process. Using, e.g., ten traces for the denominator calculation significantly slows down the training time compared to the CCE loss.

To overcome the limitations mentioned before and inspired by the focal loss [9], we propose a novel loss function for SCA: Focal Loss Ratio (FLR). The main contributions of this paper are:

- We design a novel loss function that enables deep learning models to learn from noisy or imbalanced data efficiently.
- As FLR requires tuning of additional hyperparameters, we discuss the appropriate hyperparameter tuning strategies.

– We perform systematic evaluation and benchmark on commonly used and recently proposed SCA-based loss functions.

We provide the source code in a Github repository: https://github.com/AISyLab/focal_loss.

2 Background

This section provides an introduction to deep learning-based profiling side-channel attacks. Afterward, we discuss various loss functions and the datasets used in our experiments.

2.1 Deep Learning-Based Side-Channel Analysis

Supervised machine learning aims to learn a function f mapping an input to the output based on examples of input-output pairs. The function f is parameterized by $\boldsymbol{\theta} \in \mathbb{R}^n$, where n denotes the number of trainable parameters. Supervised learning consists of two phases: training and test. Moving to the profiling side-channel attacks, those two phases are commonly denoted as the profiling and attack phase. In the deep learning-based SCA, the function f is a deep neural network with the Softmax output layer. We encode classes in one-hot encoding, where each class is represented as a vector of c values (that depends on the leakage model and the considered cipher), with zero on all the places, except one, denoting the membership of that class. Our work considers two commonly used deep learning models: multilayer perceptron and convolutional neural networks.

The **multilayer perceptron** (MLP) is a feed-forward neural network that maps sets of inputs onto sets of appropriate outputs. MLP consists of multiple layers (at least three: an input layer, an output layer, and hidden layer(s)) of nodes in a directed graph, where each layer is fully connected to the next one, and training of the network is done with the backpropagation algorithm [4].

Convolutional neural networks (CNNs) commonly consist of three types of layers: convolutional layers, pooling layers, and fully connected layers. The convolution layer computes the output of neurons connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. Pooling decrease the number of extracted samples by performing a down-sampling operation along the spatial dimensions. The fully connected layer computes either the hidden activations or the class scores.

A dataset is a collection of side-channel traces (measurements), where each trace $\mathbf{t}_i$ is associated with an input value (plaintext or ciphertext) $\mathbf{d}_i$ and a key value $\mathbf{k}_i$. Similar to the conventional machine learning process, the dataset is divided into disjoint subsets where the training set has M traces, the validation set has V traces, and the attack set has Q traces.

1. The goal of the profiling phase is to learn $\boldsymbol{\theta}$ (vector of parameters) that minimizes the empirical risk represented by a loss function L on a training dataset of size M.

2. In the attack phase (also known as test or inference), predictions are made for the classes
$$y(x_1, k^*), \ldots, y(x_Q, k^*),$$
where x denotes leakage traces and k^* represents the secret (unknown) key on the device under the attack. The outcome of predictions 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 candidate k is then used as a maximum log-likelihood distinguisher:
$$S(k) = \sum_{i=1}^{Q} \log(\mathbf{p}_{i,y}). \tag{1}$$
The value $\mathbf{p}_{i,y}$ is the probability that for a key k being used to generate leakage d_i, we obtain the class y. A specific class y is obtained from the key and input through a cryptographic function and a leakage model.

In SCA, an adversary aims at revealing the secret key k^*. More specifically, given 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. Thus, g_1 is the most likely and $g_{|\mathcal{K}|}$ the least likely key candidate. The attack performance is evaluated by standard performance metrics such as success rate (SR) and guessing entropy (GE) [21]. Guessing entropy is the average position of k^* in $\mathbf{g}$. The success rate is the average empirical probability that g_1 is equal to the secret key k^*. In this work, both metrics are considered.

2.2 Loss Functions

In machine learning, the loss indicates the difference between the predicted outputs of the model and the ground truth labels belonging to the input. The result of a loss function L is used to update the weights in the network with gradient descent, finally reducing the deviation between the predicted and true labels.

For classification, the common loss function is the categorical cross-entropy (CCE), and it has been used in various classification tasks [5,8,27]. Since side-channel attack can be considered as a classification task as well, CCE is also usually adopted in SCA [1,7,10]. Cross-entropy is a measure of the difference between two distributions. Minimizing the cross-entropy between the distribution modeled by the deep learning model and the true distribution of the classes would improve the predictions of the neural network:
$$CCE(y, \hat{y}) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{c} y_{i,j} \cdot \log(\widehat{y_{i,j}}), \tag{2}$$
where c is the number of classes, y is the true value, and $\hat{y}$ is the predicted value.

Categorical cross-entropy loss has several variants depending on usage cases. Focal loss is one of the popular ones in dealing with class imbalance problems as well as improving learning speed [9]:

$$FOCAL(y, \hat{y}) = -\alpha(1 - \hat{y})^{\gamma} CCE(y, \hat{y}), \tag{3}$$

where CCE is the categorical cross-entropy function, α is a vector of weights for each class, and γ is the parameter that increases the loss for correctly classified examples with low confidence.

More recently, two SCA-specific loss functions have been proposed. One of them is the ranking loss (RKL) proposed by Zaid et al. [28]. The ranking loss uses both the output score of the model and the probabilities produced by applying the Softmax activation function to these scores. The idea behind the ranking loss is to compare the rank of the correct key byte and the other key bytes in the score vector before the Softmax function is applied:

$$RKL(s) = \sum_{\substack{k \in \mathcal{K} \\ k \neq k^*}} \left(\log_2 \left(1 + e^{-\alpha(s(k^*) - s(k))} \right) \right), \tag{4}$$

where s is the predicted vector with scores for each key hypothesis, $\mathcal{K}$ is the set of all possible key values, k^* is the correct key, and $s(k)$ is the score for key guess k, calculated by looking at the rank of k in the list of all possible keys. Finally, α is a parameter that needs to be set dependent on the size of the used profiling set. The implementation of the ranking loss function is provided by the authors [28] on Github[1].

Zhang et al. proposed the cross-entropy ratio (CER) [30]. CER can be used as a metric to estimate the performance of a deep learning model in the context of SCA, which can be further extended as a loss function:

$$cer(y, \hat{y}) = \frac{CE(y, \hat{y})}{\frac{1}{n}\sum_{i=1}^{n} CE(y_{r_i}, \hat{y})}, \tag{5}$$

where CE is the categorical cross-entropy, and y_{r_i} denotes the one-hot encoded vector with the incorrect labels. Here, the variable n denotes the number of incorrect sets to use. The authors do not provide a value for n, but state that increasing n should increase the accuracy of the metric. We use $n = 10$ to balance computational complexity and attack performance in our experiments.

2.3 Datasets

Our experiments consider three publicly available datasets representing a typical sample of commonly encountered scenarios. These datasets are a common choice for evaluating the performance of deep learning-based SCA. For all datasets, we consider the Hamming weight (HW) and Identity (ID) leakage models.

ASCAD Datasets. The ASCAD datasets are generated by taking measurements from an ATMega8515 running masked AES-128 and are proposed as the benchmark datasets for SCA [1]. There are two versions of the dataset. The first

[1] https://github.com/gabzai/Ranking-Loss-SCA.

version consists of 50 000 profiling traces, and 10 000 attack traces, each trace consisting of 700 features. The profiling and attacking sets use both the same fixed key. We denote this dataset as ASCAD_fixed.

The second version of the ASCAD dataset uses random keys to build the profiling traces. The dataset consists of 200 000 profiling, and 100 000 attack traces, each consisting of 1 400 features. We denote this dataset as ASCAD_variable. For our experiments on the ASCAD datasets (both versions), 50 000 profiling traces are used. 2 000 traces for the ASCAD_fixed dataset and up to 3 000 traces for the ASCAD_variable are used in the attack phase. For both versions, we attack the first masked key byte, which is key byte 3. The ASCAD datasets are available on the ASCAD GitHub repository[2].

CHES CTF Dataset. The CHES CTF dataset was released in 2018 for the Conference on Cryptographic Hardware and Embedded Systems (CHES). The traces consist of masked AES-128 encryption running on a 32-bit STM microcontroller. In our experiments, we use 45 000 traces for the training set, which contains a **fixed key**. The validation and test sets consist of 5 000 traces each, where we used 3 000 traces for the attack phase. Unlike the ASCAD dataset, the key used in the training and validation set is different from the key for the test set. We attack the first key byte. This dataset is available at https://chesctf.riscure.com/2018/news. In our case, we considered a pre-processed version of the dataset where each trace consists of 2 200 features. The pre-processed dataset is available at http://aisylabdatasets.ewi.tudelft.nl/.

3 Related Works

To improve the side-channel attack performance, in recent years, deep learning has received more attention within the SCA community, see, e.g., [2,7,10,12,17,18,23,29]. MLP and CNNs have become the most popular candidates to launch such attacks. The results show that by carefully tuning the model's hyperparameters, the required number of attacks traces can be dramatically reduced to obtain the secret key. For instance, [29] proposed a methodology to find well-performing architectures for SCA, while [7] also researched different architectural choices and the influence of noise. More recently, [15] showed how ensembles of deep learning models can be used for SCA.

Although the model design methodologies have been widely studied, less attention has been put on the loss function. [10] first explored the usage of convolutional neural networks for SCA and mentioned the categorical cross-entropy and the mean squared error loss functions, which is followed by later works on deep learning-based SCA [1,13,15,23,29].

More recently, two novel loss functions optimized for SCA have been proposed [28,30]. More details are presented in Sect. 2.2. Finally, Kerkhof et al.

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

recently conducted a systematic evaluation of several loss functions ("traditional" machine learning ones like categorical cross-entropy and mean squared error) but also the SCA-related ones (CER and ranking loss) [6]. Their analysis showed that CER performs the best for SCA, followed by categorical cross-entropy. Interestingly, the reported results for ranking loss indicate significant issues with that loss function.

4 A Novel Loss Function for SCA

In this section, we introduce our novel loss function. First, we provide a formal problem statement, followed by a discussion about the FLR loss function and how to tune its hyperparameters.

4.1 Problem Statement

Before introducing the Focal Loss Ratio, we first formally define the easy and hard samples [20]. Let a, p, and n denote *anchor* (i.e., ground truth), *positive* (with a label same as the anchor), and *negative* samples (with a label different from the anchor). In general, the anchor can be the data of any label, and the positive and negative samples are based on the anchor's label. We can categorize the positive samples p into two categories based on their similarity S to the anchor sample: 1) easy samples, where $S(a,p) < S(a,n)$; 2) hard samples, where $S(a,n) < S(a,p)$. The way of calculating the similarity depends on the selection of the loss function. Nevertheless, the samples closer to the anchor have higher confidence to be classified to the corresponding clusters. Following this, based on the classification outcomes, we define:

- Easy positives/negatives: samples classified as positive/negative examples.
- Hard positives/negatives: samples misclassified as negative/positive examples.

Recall that the CER loss takes advantage of samples with incorrect labels to increase the attack performance. However, the training would become inefficient if most samples are easy negatives that have limited contribution to the learning process. The bias introduced by easy negatives makes it difficult for a network to learn rich semantic relationships from samples: cumulative easy negatives loss overwhelms the total loss, which degenerates the model. Moreover, one should notice that the class imbalance can be introduced based on the leakage model. For instance, when using the Hamming weight leakage model, information related to middle classes (i.e., $\mathrm{HW} = 4$) in a dataset or mini-batches used in training is over-represented compared to the other classes. Indeed, training a network on an imbalanced dataset will force the network to learn more representations of the data-dominated class than other classes. Unfortunately, besides re-balancing from the dataset level, there are no special measures to address this problem during the training process. Finally, the accurate estimate of CER requires a sufficient number of negative samples (infinite in the ideal case), but it would reduce the training efficiency as a trade-off.

4.2 Focal Loss Ratio

Two actions are essential to address the problems identified in the previous section. First, the hard samples should be prioritized in the training process compared to the easier ones. Second, the weight of each class should be parameterized. Following this, we propose the Focal Loss Ratio (FLR):

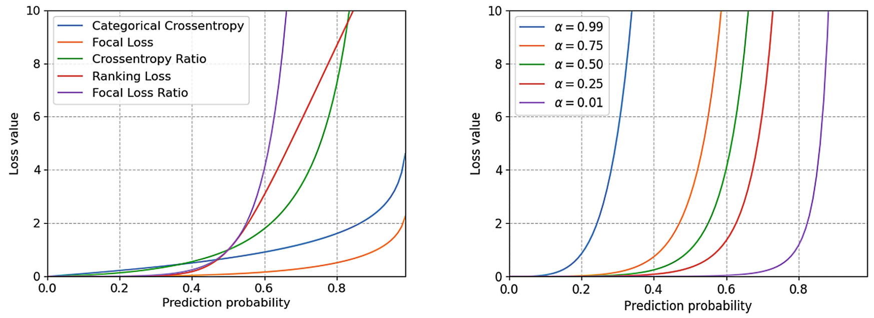
$$FLR(y, \hat{y}) = \frac{-\alpha(1-\hat{y})^{\gamma}CE(y, \hat{y})}{\frac{1}{n}\sum_{i=1}^{n} -\alpha(1-\hat{y})^{\gamma}CE(y_{s_i}, \hat{y})}, \tag{6}$$

where y are the true labels, y_s are the shuffled labels, CE is the categorical cross-entropy, and n is the number of negative samples to use. In Eq. 6, α and γ are introduced to weight the classes and emphasize hard samples for both numerator and denominator, respectively. When looking at the numerator, aligned with the focal loss, the samples with lower prediction probability (hard samples) have a greater impact on the loss function, which is further controlled by the α value. The same statement holds for the denominator as well. Besides, the introduction of the denominator further separates the prediction distribution between the correct cluster and other clusters. Indeed, compared with other loss functions, FLR introduces additional benefits to efficient learning: 1) concentrating on the samples that are difficult to classify (hard samples) and 2) balancing the dataset. Finally, FLR can be seen as an improved version of CER loss, focusing on learning efficiency. Since the theoretical evidence from the CER loss also applies to our FLR loss, we do not repeat it in this work.

Figure 1 demonstrates the above mentioned effects. Given that input in the prediction probability $\hat{y}$ ranges from zero to one and the ground truth y equals zero, as shown in the left graph, FLR ($\alpha = 0.5$) introduces the greatest penalty to the hard samples compared to others. When y_{pred} is getting closer to y_{true}, the FLR value is neglectable, thus reducing the contribution of the easy negatives. The effect of α is shown on the right graph: the influence of the hard samples is reduced when α decreases. Consequently, the FLR loss could be a good candidate when the classes are imbalanced (i.e., the HW leakage model). Moreover, since α can effectively control the hard sample's influence, then the improvement of the model's performance can be realized by different tuning strategies. More discussions are presented in Sect. 6.

4.3 Hyperparameter Tuning

Compared with other loss functions, FLR loss introduces additional hyperparameters. Thus, it requires careful tuning. We consider three strategies for α and γ selection to investigate their influence and reach the top performance in the considered testing scenarios. For the first strategy, we use the values given by [9], namely $\alpha = 0.25$ and $\gamma = 2.0$. Models with these settings are denoted as FLR. The second strategy optimizes both α and γ via random search, denoted as FLR_optimized. The search ranges are defined in Table 1.



(a) Comparison between loss functions. (b) The influence of α towards the FLR.

Fig. 1. Demonstration of different loss functions.

Table 1. Hyperparameter space for FLR_optimized.

α	0.1,	0.25,	0.5,	0.75,	0.9
γ	0,	0.5,	1.0,	2.0,	5.0

Finally, we introduce class re-balancing into our loss function [3]. With class balancing, the weights for each class (α) are set based on the classes' size. For each class, the corresponding weight is calculated as shown in Eq. 7.

$$\alpha_i = \frac{1-\beta}{1-\beta^{n_y}}, \tag{7}$$

where α_i is the weight for class i, n_y is the number of samples in the considered class in the profiling set, and β is a new parameter to be tuned. In this paper, aligned with [3], we set $\beta = 0.999$. Models trained with these settings are referred to as focal_balanced.

We conducted a preliminary search to determine the optimal value of n (ranges from 1 to 20). Our experiments showed the best attack performance when n equals three. This observation also holds when tested on the other datasets. Also, the impact on training time of using $n = 3$ is negligible compared to $n = 1$. Therefore, we set n to three for our experiments with FLR.

5 Experimental Results

In this section, we provide the experimental results for our new loss function. First, we provide details about the experimental setup. Afterward, we provide results for all considered datasets and loss functions.

5.1 Setup

Regarding model architecture tuning, using one or a few optimized models from the literature may introduce bias as they are optimized for a specific dataset-loss function combination. Besides, the model's performance may fluctuate with each training due to the random weight initialization. Therefore, we follow Algorithm 1 to tune the model's hyperparameters for each loss function.

Algorithm 1. Model tuning and the evaluation strategy.

```
1: Generate, train, and test $Z$ models sampled from range $S$ with loss function $L$.
2: Select the **best** performing model $T_b$.
3: Train and test the model $T_b$ $N$ times.
4: Select the **median** performing model $T_{bm}$.
5: Evaluate $T_{bm}$ with evaluation metrics.
```

This paper compares our function against the CER loss, categorical cross-entropy, ranking loss, and focal loss. The selection of "traditional" loss functions is based on the results from [6]. Note that for the RKL's α value, the original paper selected 0.5 for the ASCAD dataset and did not provide values for the other datasets. Since the number of profiling traces we used was almost the same for all datasets, $\alpha = 0.5$ was used for every dataset and model. Although this value can be further optimized, we argue that tuning α for all of the scenarios and architectures is not viable and practical for real-world usages, considering the number of different scenarios/architectures that are relevant.

For each loss function, we set Z to 100 with hyperparameters sampled from Tables 2 and 3. n is set to be 10. We use guessing entropy to evaluate the model's performance during the tuning process (steps 2 and 4). For the evaluation (step 5), we look at the guessing entropy and success rate. In some of the plots in the following sections, the x-axis is reduced to increase visibility.

Table 2. Hyperparameter space for multilayer perceptrons.

Hyperparameter	Options
Dense layers	2 to 8 in a step of 1
Neurons per layer	100 to 1 000 in a step of 100
Learning rate	1e−6 to 1e−3 in a step of 1e−5
Batch size	100 to 1 000 in a step of 100
Activation function (all layers)	ReLU, Tanh, ELU, or SeLU
Loss function	RMSprop, Adam

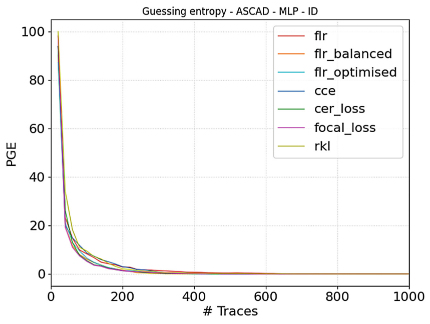
Table 3. Hyperparameter space for convolutional neural networks.

Hyperparameter	**Options**
Convolution layers	1 to 2 in a step of 1
Convolution filters	8 to 32 in a step of 4
Kernel size	10 to 20 in a step of 2
Pooling type	Max pooling, Average pooling
Pooling size	2 to 5 in a step of 1
Pooling stride	2 to 10 in a step of 1
Dense layers	2 to 3 in a step of 1
Neurons per layer	100 to 1 000 in a step of 100
Learning rate	1e−6 to 1e−3 in a step of 1e−5
Batch size	100 to 1 000 in a step of 100
Activation function (all layers)	ReLU, Tanh, ELU, or SeLU
Loss function	RMSprop, Adam

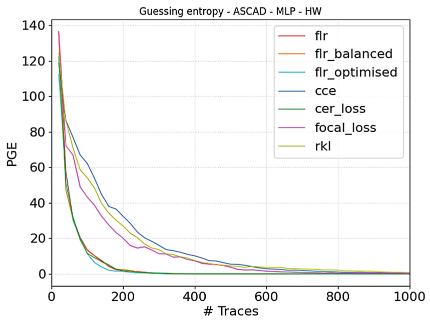
5.2 ASCAD_fixed

Figures 2 and 3 show the guessing entropy and success rate metrics with different attack models and leakage models. From the results, models trained with FLR loss outperform the CCE and focal loss in all test scenarios. Specifically, when the HW leakage model is considered, the FLR model halves the required attack traces compared with categorical cross-entropy or focal loss to reach a GE of 1. Surprisingly, ranking loss performs mediocre in most cases, indicating its low generality towards different deep learning models and test scenarios. Note that we tested on the same datasets as the RKL paper does, and the poor performance mainly comes from the variation of the attack model (recall, we use models created via random search). Unfortunately, although RKL may work well with some specific settings (like the one in [28]), the general applicability of that loss function is relatively poor based on our results.

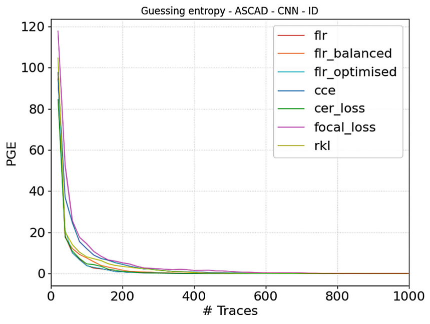
On the other side, FLR loss and CER loss perform comparably. Still, as shown in Table 4, when the median $\overline{N}_{T_{GE}}$ is evaluated, the models trained with FLR outperform the CER loss in three out of four of the test scenarios. Interestingly, all three FLR tuning strategies (for α and γ) work well and lead to successful attacks with a limited number of traces. Although optimal strategy differs per scenario, their variation is limited.



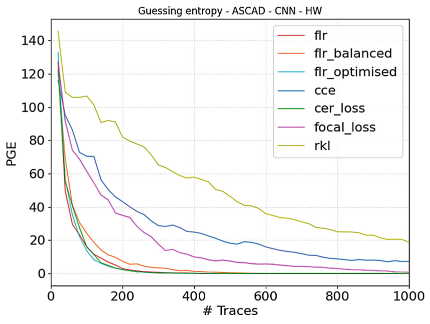
(a) MLP models, ID leakage model.



(b) MLP models, HW leakage model.



(c) CNN models, ID leakage model.



(d) CNN models, HW leakage model.

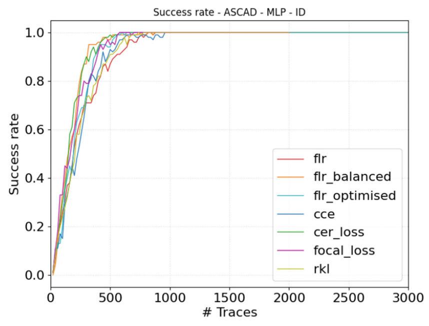
Fig. 2. Guessing entropy of the optimized models for the ASCAD_fixed dataset.

Table 4. Median $\overline{N}_{T_{GE}}$ for the ASCAD_fixed dataset. The lowest $\overline{N}_{T_{GE}}$ for each scenario is marked blue.

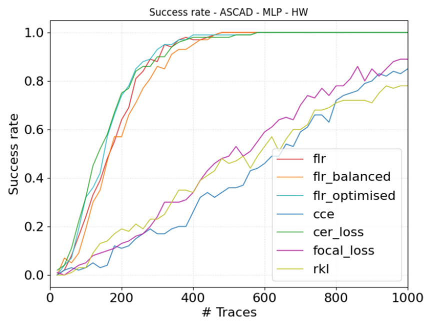
	L_{focal}	CCE	CER loss	RKL	FLR	FLR_balanced	FLR_optimized
MLP ID	580	860	570	900	810	540	680
MLP HW	1480	1560	560	1620	460	570	510
CNN ID	1250	1360	600	1760	610	850	550
CNN HW	1840	>2000	540	>2000	570	790	560

5.3 ASCAD_variable

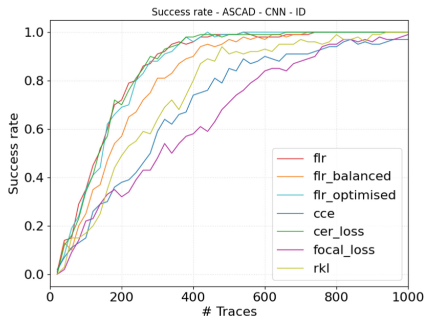
Next, loss functions are tested on the ASCAD_variable dataset. The guessing entropy for each loss function is presented in Fig. 4. For the ID leakage model, neither the MLPs nor CNNs reach a GE of 1 with less than 3 000 traces. Still, the CER loss and FLR perform the best: the CER loss reaches a GE of 1.7 with MLP and 3.13 with CNN, while the models with FLR reach 2.11 and 1.18. When the HW leakage model is considered, as shown in Table 5, the secret key



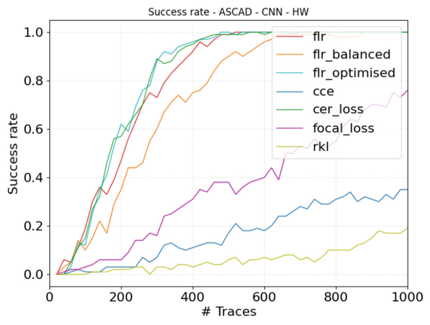
(a) MLP models, ID leakage model.



(b) MLP models, HW leakage model.



(c) CNN models, ID leakage model.



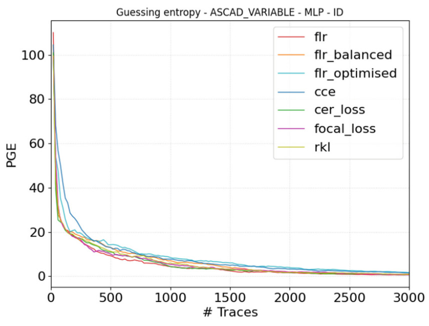
(d) CNN models, HW leakage model.

Fig. 3. Success rate of the optimized models for the ASCAD_fixed dataset.

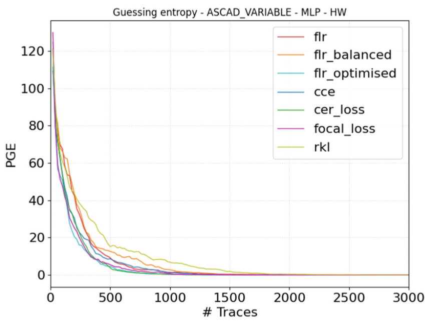
can be retrieved successfully with all considered loss functions. For MLP, FLR loss performs slightly worse than CER ($\overline{N}_{T_{GE}} = 1\,800$ versus $\overline{N}_{T_{GE}} = 1\,340$). For CNN, FLR outperforms CER ($\overline{N}_{T_{GE}} = 800$ versus $\overline{N}_{T_{GE}} = 950$). Ranking loss, unfortunately, performs the worst in most of the test scenarios.

Table 5. Median $\overline{N}_{T_{GE}}$ for the ASCAD_variable dataset. The lowest $\overline{N}_{T_{GE}}$ for each scenario is marked blue.

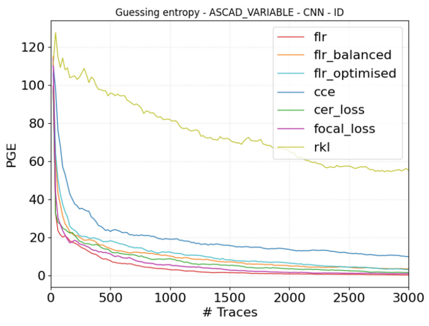
	L_{focal}	CCE	CER loss	RKL	FLR	FLR_balanced	FLR_optimized
MLP ID	>3 000	>3 000	>3 000	>3 000	>3 000	>3 000	>3 000
MLP HW	1 940	2 600	1 340	2 910	2 180	2 460	1 800
CNN ID	>3 000	>3 000	>3 000	>3 000	>3 000	>3 000	>3 000
CNN HW	>3 000	2 840	950	>3 000	880	1 670	1 020



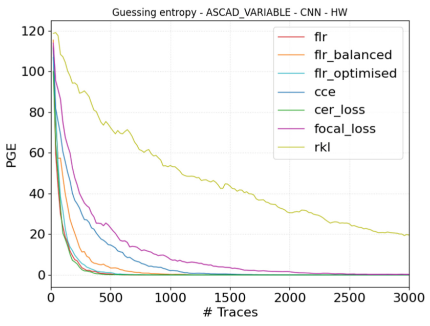
(a) MLP models, ID leakage model.



(b) MLP models, HW leakage model.



(c) CNN models, ID leakage model.



(d) CNN models, HW leakage model.

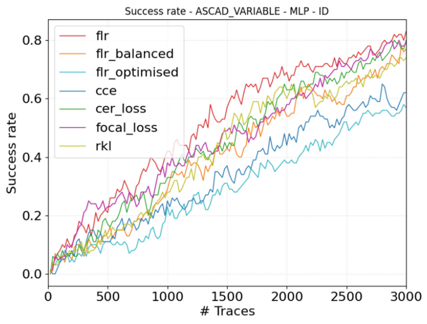
Fig. 4. Guessing entropy of the optimized models for the ASCAD_variable dataset.

Next, the success rates (SR) of each loss function are shown in Fig. 5. Interestingly, the FLR (default version) equipped model reaches a higher SR slightly faster than the other loss functions with the ID leakage model. The FLR and CER loss perform equally well for the HW leakage scenarios. Note that the performance of FLR can fluctuate with different hyperparameter tuning strategies. For the ASCAD_variable dataset, however, FLR with default values ($\alpha = 0.25$, $\gamma = 2.0$) would be a good choice.

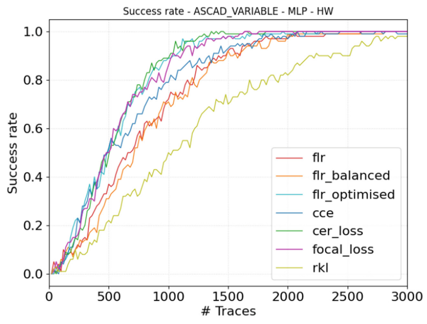
5.4 CHES_CTF

In this section, we discuss the results for the CHES_CTF dataset. Figure 6 shows the guessing entropy in the different scenarios.

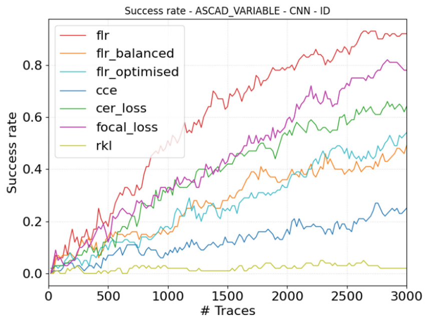
For all considered loss functions, 3 000 attack traces are not sufficient to obtain the correct key for the ID leakage model. Still, from the results, we see a significant performance improvement with the MLP models and the ID leakage when using FLR_balanced. Such an improvement is also visible in some of the CNN models with FLR. However, these models turned out to be less consistent in terms of performance when changing the attack settings. For instance, the



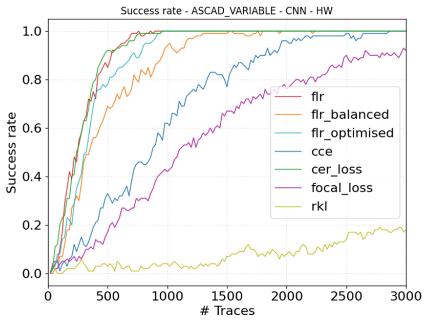
(a) MLP models, ID leakage model.



(b) MLP models, HW leakage model.



(c) CNN models, ID leakage model.



(d) CNN models, HW leakage model.

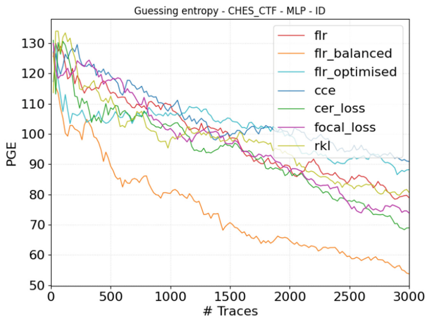
Fig. 5. Success rate of the optimized models for the ASCAD_variable dataset.

FLR_balanced performs the best with MLP, but it performs mediocre with CNN. Similar behavior is also visible for the FLR_optimized.

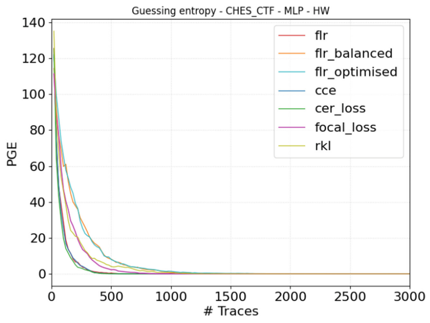
When the HW leakage model is considered, we again see a significant increase in the performance when a CNN is used. As shown in Table 6, the models with FLR and FLR_optimized were the only ones that successfully retrieved the correct key. The median of 10 models with FLR and FLR_optimized were successful with a $\overline{N}_{T_{GE}}$ of 2 740 and 2 000, respectively. When MLPs are used, there is no significant increase in $\overline{N}_{T_{GE}}$, and the performance is approximately equal to the CER loss.

6 Discussion

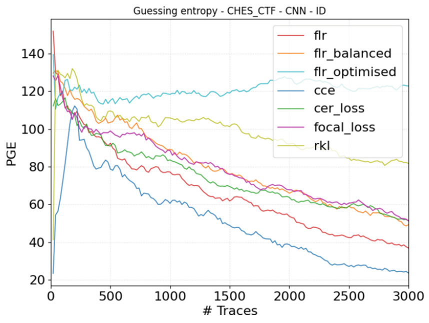
FLR loss performs well in various test scenarios, while the only downside to using FLR as a loss function is the introduction of the α and γ parameters. We used three different strategies: 1) fixed value: $\alpha = 0.25$ and $\gamma = 2.0$; 2) optimized via random search; 3) determined by the frequency of each class.



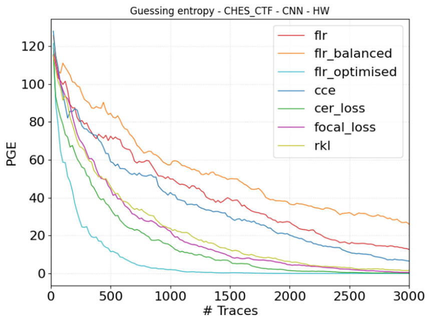
(a) MLP models, ID leakage model.



(b) MLP models, HW leakage model.



(c) CNN models, ID leakage model.



(d) CNN models, HW leakage model.

Fig. 6. Guessing entropy of the optimized models for the CHES_CTF dataset.

Table 6. Median $\overline{N}_{T_{GE}}$ for the CHES_CTF dataset. The lowest $\overline{N}_{T_{GE}}$ for each scenario is marked blue.

	L_{focal}	CCE	CER loss	RKL	FLR	FLR_balanced	FLR_optimized
MLP ID	>3 000	>3 000	>3 000	>3 000	>3 000	>3 000	>3 000
MLP HW	1 220	630	480	1 860	1 080	2 030	2 450
CNN ID	>3 000	>3 000	>3 000	>3 000	>3 000	>3 000	>3 000
CNN HW	>3 000	>3 000	>3 000	>3 000	2 740	>3 000	2 000

Throughout the experiments, there was not a single strategy that worked best for every scenario. Still, the best performing FLR variants have the fixed α values for every class in almost all cases. In some of the scenarios with the ID leakage model, the class re-balance strategy improves the performance. However, using class balancing with the ID leakage model results in almost constant and low values of α. This leads us to conclude that the best strategy is the variant where α is the same for every class and the α and γ parameters are optimized. Optimization via random search can be performed to set the α and γ values. In combination with an increased range of the possible values, e.g., the addition of lower α values, FLR_optimized should outperform the other variants. From Sect. 4.2, one should note that with lower α, the samples that trigger high loss value are the ones misclassified with high confidence (probability).

Compared with other loss functions that require models to be confident about predicting, this FLR configuration softens the restriction for the predictions: only (very) hard negative will be penalized, while the others that are correctly classified, or even misclassified but with low confidence would have limited loss contributions. From the learning perspective, loss functions forcing the model to reach high accuracy/low loss would normally lead to the learning from the major classes/overfitting. FLR with low α allows the models to make mistakes, increasing the model's generality and helping to learn from the imbalanced data.

We performed an additional set of experiments on ASCAD_fixed and ASCAD_variable datasets to test our hypothesis. The search space for α is now extended to $0.005, 0.01, 0.05, 0.1, 0.25, 0.5, 0.75$, and 0.9. We use FLR as the loss function for each test scenario and again optimize hyperparameters via random search. The results of these experiments are listed in Tables 7 and 8.

Table 7. Median $\overline{N}_{T_{GE}}$ for the ASCAD_fixed dataset. The lowest $\overline{N}_{T_{GE}}$ for each scenario is marked blue.

	L_{focal}	CCE	CER loss	RKL	FLR
MLP ID	580	860	570	900	640
MLP HW	1 480	1 560	560	1 630	490
CNN ID	1 250	1 360	600	1 760	520
CNN HW	1 840	>2 000	540	>2 000	500

Table 8. Median $\overline{N}_{T_{GE}}$ for the ASCAD_variable dataset. The lowest $\overline{N}_{T_{GE}}$ for each scenario is marked blue.

	L_{focal}	CCE	CER loss	RKL	FLR
MLP ID	>3 000	>3 000	>3 000	>3 000	>3 000
MLP HW	1 940	2 600	1 340	2 910	1 340
CNN ID	>3 000	>3 000	>3 000	>3 000	>3 000
CNN HW	>3 000	2 840	950	>3 000	800

From the results, in the scenarios in which the class balanced FLR was previously best, such as the ASCAD_fixed scenarios, the FLR with our new strategy still performs very well. For instance, when attacking ASCAD_fixed with MLP and the ID leakage model, the best performing model uses a fixed α that equals 0.005. Although it did not perform as well as the CER loss or FLR_balanced in this case, it did perform better than the other strategies. We also see results similar to the previous experiments when using the HW leakage model on the

ASCAD_variable dataset. FLR outperforms the CER loss in most cases. The benefit, however, is that a single strategy can be used for each scenario, namely the same optimized value for α for each class.

7 Conclusions and Future Work

In this paper, we proposed a novel loss function optimized for deep learning-based side-channel analysis. More precisely, we started by discussing the advantages and drawbacks of several loss functions in the context of SCA. Using those characteristics, we constructed a new loss function for deep learning-based SCA denoted as the Focal Loss Ratio (FLR).

We confirmed FLR's outstanding performance by testing it on combinations of datasets, leakage models, and neural network architectures. Finally, we showed that neural network models using FLR work with different parameter optimization strategies and that FLR outperforms the CER loss and other loss functions like the categorical cross-entropy in most of the considered scenarios. We plan to explore the hyperparameter selection for FLR loss when considering datasets with more complex countermeasures for future work.

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).

References

1. Benadjila, R., Prouff, E., Strullu, R., Cagli, E., Dumas, C.: Deep learning for side-channel analysis and introduction to ASCAD database. J. Cryptogr. Eng. **10**(2), 163–188 (2019). https://doi.org/10.1007/s13389-019-00220-8
2. 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
3. Cui, Y., Jia, M., Lin, T.Y., Song, Y., Belongie, S.J.: Class-Balanced Loss Based on Effective Number of Samples. CoRR abs/1901.05555 (2019). http://arxiv.org/abs/1901.05555
4. Goodfellow, I.J., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016). http://www.deeplearningbook.org
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition (2015)
6. Kerkhof, M., Wu, L., Perin, G., Picek, S.: No (good) loss no gain: systematic evaluation of loss functions in deep learning-based side-channel analysis. Cryptology ePrint Archive, Report 2021/1091 (2021). https://ia.cr/2021/1091
7. 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. IACR Trans. Cryptogr. Hardware Embed. Syst. **2019**(3), 148–179 (2019). https://doi.org/10.13154/tches.v2019.i3.148-179. ISSN 2569-2925

8. Kussul, N., Lavreniuk, M., Skakun, S., Shelestov, A.: Deep learning classification of land cover and crop types using remote sensing data. IEEE Geosci. Remote Sens. Lett. **14**(5), 778–782 (2017). https://doi.org/10.1109/LGRS.2017.2681128
9. Lin, T.Y., Goyal, P., Girshick, R.B., He, K., Dollár, P.: Focal Loss for Dense Object Detection. CoRR abs/1708.02002 (2017). http://arxiv.org/abs/1708.02002
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, Cham (2006). http://www.dpabook.org/. ISBN 0-387-30857-1
12. Masure, L., Dumas, C., Prouff, E.: A Comprehensive Study of Deep Learning for Side-Channel Analysis (2019)
13. Moos, T., Wegener, F., Moradi, A.: DL-LA: deep learning leakage assessment: a modern roadmap for SCA evaluations. IACR Trans. Cryptogr. Hardware Embed. Syst. **2021**(3), 552–598 (2021). https://doi.org/10.46586/tches.v2021.i3.552-598. https://tches.iacr.org/index.php/TCHES/article/view/8986
14. Perin, G., Buhan, I., Picek, S.: Learning when to stop: a mutual information approach to fight overfitting in profiled side-channel analysis. Cryptology ePrint Archive, Report 2020/058 (2020). https://ia.cr/2020/058
15. Perin, G., Chmielewski, L., Picek, S.: Strength in numbers: improving generalization with ensembles in machine learning-based profiled side-channel analysis. IACR Trans. Cryptogr. Hardware 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
16. 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. Hardware 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
17. Picek, S., Samiotis, I.P., Kim, J., Heuser, A., Bhasin, S., Legay, A.: On the performance of convolutional neural networks for side-channel analysis. In: Chattopadhyay, A., Rebeiro, C., Yarom, Y. (eds.) SPACE 2018. LNCS, vol. 11348, pp. 157–176. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-05072-6_10
18. Rijsdijk, J., Wu, L., Perin, G., Picek, S.: Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis. IACR Trans. Cryptogr. Hardware 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
19. Robissout, D., Zaid, G., Colombier, B., Bossuet, L., Habrard, A.: Online performance evaluation of deep learning networks for profiled side-channel analysis. In: Bertoni, G.M., Regazzoni, F. (eds.) COSADE 2020. LNCS, vol. 12244, pp. 200–218. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-68773-1_10
20. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: a unified embedding for face recognition and clustering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 815–823 (2015)
21. 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

22. 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
23. Timon, B.: Non-profiled deep learning-based side-channel attacks with sensitivity analysis. IACR Trans. Cryptogr. Hardware Embed. Syst. **2019**(2), 107–131 (2019). https://doi.org/10.13154/tches.v2019.i2.107-131. https://tches.iacr.org/index.php/TCHES/article/view/7387
24. Wouters, L., Arribas, V., Gierlichs, B., Preneel, B.: Revisiting a methodology for efficient CNN architectures in profiling attacks. IACR Trans. Cryptogr. Hardware Embed. Syst. **2020**(3), 147–168 (2020). https://doi.org/10.13154/tches.v2020.i3.147-168. https://github.com/KULeuven-COSIC/TCHES20V3_CNN_SCA
25. Wu, L., Picek, S.: Remove some noise: on pre-processing of side-channel measurements with autoencoders. IACR Trans. Cryptogr. Hardware Embed. Syst. 389–415 (2020)
26. Wu, L., Won, Y., Jap, D., Perin, G., Bhasin, S., Picek, S.: Explain some noise: ablation analysis for deep learning-based physical side-channel analysis. IACR Cryptol. ePrint Arch. 717 (2021). https://eprint.iacr.org/2021/717
27. Yuan, B., Wang, J., Liu, D., Guo, W., Wu, P., Bao, X.: Byte-level malware classification based on Markov images and deep learning. Comput. Secur. **92**, 101740 (2020). https://doi.org/10.1016/j.cose.2020.101740. https://www.sciencedirect.com/science/article/pii/S0167404820300262
28. Zaid, G., Bossuet, L., Dassance, F., Habrard, A., Venelli, A.: Ranking loss: maximizing the success rate in deep learning side-channel analysis. IACR Trans. Cryptogr. Hardware Embed. Syst. **2021**(1), 25–55 (2020). https://doi.org/10.46586/tches.v2021.i1.25-55. https://tches.iacr.org/index.php/TCHES/article/view/8726
29. Zaid, G., Bossuet, L., Habrard, A., Venelli, A.: Methodology for efficient CNN architectures in profiling attacks. IACR Trans. Cryptogr. Hardware 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
30. Zhang, J., Zheng, M., Nan, J., Hu, H., Yu, N.: A novel evaluation metric for deep learning-based side channel analysis and its extended application to imbalanced data. IACR Trans. Cryptogr. Hardware Embed. Syst. 73–96 (2020)
31. Zhu, D., Yao, H., Jiang, B., Yu, P.: Negative Log Likelihood Ratio Loss for Deep Neural Network Classification (2018). http://arxiv.org/abs/1804.10690