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]

Delft University of Technology, 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 evaluation method for the evaluation labs/certification schemes. To reach this performance level, recent works significantly improved the deep learning-based attacks from various perspectives, like hyperparameter tuning, design guidelines, or custom neural network architecture elements. Still, limited 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 commonly used in the deep learning-based side-channel analysis (both "traditional" one and those designed for side-channel analysis). Our results show that FLR loss outperforms other loss functions in various conditions while not having computation overheads compared to common loss functions like categorical cross-entropy.

Keywords: Side-channel analysis, Deep Learning, Loss function, Focal loss

1 Introduction

Side-channel analysis (SCA) is one of the most popular tools to exploit the implementation weakness of an algorithm [MOP06]. Commonly, SCA attacks can be divided into two categories: direct attacks and profiling attacks. The former attack method analyzes the leakages from the target device directly, while the latter requires a copy of the target device. An attacker would first learn the characteristic of the copied device (profiling phase), then launch an attack on the target device (attack phase).

With stronger attack assumptions, profiling attacks are considered more powerful than their counterpart. The rise of deep learning in recent years further increased the profiling attacks' capability. Specifically, such attacks can break targets protected with countermeasures [KPH+19, WP20, ZBHV19]. Moreover, compared with the conventional profiling attack (i.e., template attack [SJP03]) that relies upon points of interest selection, deep learning-based SCA has softer restrictions on data pre-processing/feature engineering.

Unfortunately, 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 [ZBHV19, WAGP20, RWPP21], evaluation metric design [ZZN+20], as well as model's interpretability and explainability [WWJ+21]. 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 [PHJ+18, ZZN+20]. 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., [RZC$^+$21, PBP20]). Consequently, the SCA community put a significant attention on developing SCA-specific metrics and loss functions.

The cross-entropy ratio, or CER loss, is one of the recently developed methods to improve the performance of deep learning models in the SCA domain [ZZN$^+$20]. 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 [ZYJY18]. However, CER loss has two limitations. First, the CER's denominator calculation can be a challenging task. Indeed, even for the traces that belong to the same cluster, the classification difficulties of each other 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. As presented in this work, using ten traces for the denominator calculation doubles the training times compared with the CCE loss. Therefore, to overcome the limitations mentioned before, and inspired by the focal loss, 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 the noisy or imbalanced data efficiently.
- We discuss the hyperparameter tuning strategy of the proposed loss function.
- We perform systematic evaluation and benchmark on commonly used and recently proposed loss functions.

The remainder of this paper is organized as follows: Section 2 describes the background and defines the loss functions and datasets used in this paper. The related work is presented in Section 3, followed by a new loss function - Focal Loss Ratio being proposed in Section 4. Section 5 validates the proposed loss function experimentally and provides benchmarks with other loss functions. Finally, Section 7 concludes this paper.

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 profiled side-channel attacks, those two phases are commonly denoted as the profiling and attack phase. In our work, 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 depend on the leakage models and the considered cipher) that has zero on all the places, except one, denoting the membership of that class. This paper considers two commonly used deep learning models, multilayer perceptron (MLP) and convolutional neural networks (CNNs).

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: input layer, 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 [GBC16].

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) $\mathbf{T}$, 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 minimize the empirical risk represented by a loss function L on a dataset T of size M (i.e., on the profiling (training) set).

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 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 probability $S(k)$ for any key candidate k is then used as a maximum log-likelihood distinguisher in the following way:
$$S(k) = \sum_{i=1}^{Q} \log(\mathbf{p}_{i,v}). \tag{1}$$
The value $\mathbf{p}_{i,v}$ is the probability for a key k and input d_i, we obtain the class v (with $\sum_v^c \mathbf{p}_{i,v} = 1, \forall i$). A specific class v 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 amount of 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 the guessing entropy (GE) [SMY09]. Guessing entropy is the average position of k^* in $\mathbf{g}$. 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 the machine learning domain, the loss indicates the difference between the predicted outputs of the model and the ground truth labels that belong to the input. The result of the loss function L is used to update the weights in the network with gradient descent, finally reducing the deviation between the predicted and real labels.

For classification, the common loss function is the categorical cross-entropy (CCE) and it has been used in various classification tasks [KLSS17, YWL+20, HZRS15]. Since side-channel attack can be considered as a classification task as well, CCE is also usually adopted in SCA [BPS+20, MPP16, KPH+19]. 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:

$$cce(y, \hat{y}) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{c} y_{i,j}, \log(\hat{y_{i,j}}), \tag{2}$$

where c denotes the number of classes.

Categorical cross-entropy loss has several variants depends on usage cases. Focal loss is one of the popular ones in dealing with class imbalance problems as well as improving learning efficiency. The definition of the focal loss is given in Equation 3.

$$L_{focal}(y, \hat{y}) = -\alpha(1-\hat{y})^{\gamma} CE(y, \hat{y}), \tag{3}$$

where CE 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. [ZBD+20]. 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, 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 $\mathbf{k}$. 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 [ZBD+20] on Github [1].

Zhang et al. proposed the cross-entropy ratio (CER) [ZZN+20]. 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. [ZZN+20] do not provide a value for n but state that increasing n should increase the accuracy of the metric. In our experiments, we use $n = 10$ to balance computational complexity and attack performance.

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.

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

2.3.1 ASCAD Datasets

The ASCAD dataset is generated by taking measurements from an ATMega8515 running masked AES-128 and is proposed as a benchmark dataset for SCA [BPS+20]. There are two versions of the dataset. The first 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 dataset is provided on the ASCAD GitHub repository [2].

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. The ASCAD_variable dataset is available on the ASCAD GitHub repository [3].

2.3.2 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. 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. Each trace consists of 2 200 features. This dataset is available at https://chesctf.riscure.com/2018/news.

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., [ZBHV19, PSK+18, MPP16, CDP17, KPH+19, MDP19, Tim19, RWPP21]. Among them, MLP and CNN become the most popular candidates to launch such attacks. The results turn out 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, [ZBHV19] proposed a methodology to find well-performing architectures for SCA, while [KPH+19] also researched different architectural choices and the influence of noise. More recently, [PCP20] 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. [MPP16] first explored the usage of deep learning techniques for SCA and mentioned the usability of the categorical cross-entropy or the mean squared error loss functions, which is followed by later works on deep learning-based SCA [ZBHV19, PCP20, BPS+20, Tim19, MWM21].

More recently, two novel loss functions optimized for SCA have been proposed [ZBD+20, ZZN+20]. As already mentioned, Zaid et al. proposed ranking loss, a loss function that maximizes the models' success rate by performing a pairwise comparison between the possible different key hypotheses. Zhang et al. proposed the cross-entropy ratio, the ratio between the categorical cross-entropy of the original profiling traces and a set of profiling traces with incorrect labels. The CER loss function should, according to the authors, be better suited for imbalanced profiling data [ZZN+20].

For both papers, although the newly proposed loss functions are compared to the categorical cross-entropy, the generality of such loss functions is unclear due to the limited testing on different test scenarios (i.e., deep learning models and leakage models). *Our work proposes a novel loss function optimized for SCA, and we perform a broad benchmark with the loss functions mentioned above.* Finally, Kerkhof et al. recently conducted a

[2] https://github.com/ANSSI-FR/ASCAD/tree/master/ATMEGA_AES_v1/ATM_AES_v1_fixed_key
[3] https://github.com/ANSSI-FR/ASCAD/tree/master/ATMEGA_AES_v1/ATM_AES_v1_variable_key

systematic evaluation of a number of loss functions ("traditional" machine learning ones like categorical cross-entropy and mean squared error) but also the SCA-related ones (CER and ranking loss) [KWPP21]. Their analysis showed that CER performs the best in SCA, followed by the 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. 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). We can categorize the positive samples p into two categories based on their similarity S to the anchor sample:

- Easy samples: $S(a,p) < S(a,n)$.
- Hard samples: $S(a,n) < S(a,n)$.

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, 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., 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 (thus reducing the profiling traces size), 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.

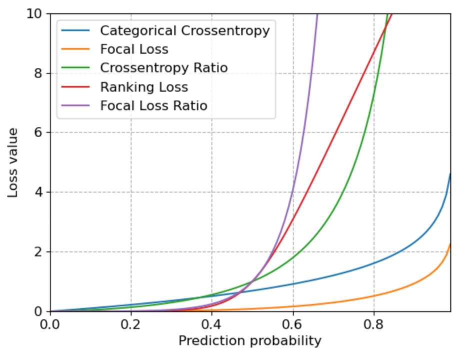
4.2 Focal Loss Ratio

Two actions are essential to be taken to address the problems identified in the previous section. First, the hard samples should be prioritized in the training process than the easier ones. Second, the weight of each class should be parameterized. Following this, we propose the Focal Loss Ratio (FLR). The definition is shown in Equation 6.

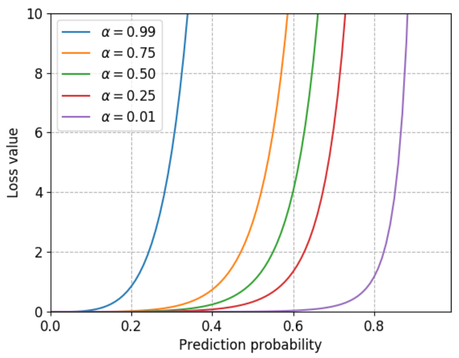
$$FLR(y, \hat{y}) = \frac{-\alpha(1-\hat{y})^{\gamma} CE(y, \hat{y})}{\frac{1}{n}\sum_{i=1}^{n} -\alpha(1-\hat{y})^{\gamma} CE(y_s, \hat{y})}, \tag{6}$$

where y are the true labels, y_s are the shuffled labels, CE is the categorical cross-entropy, and n is the number of sample-incorrect label pairs to use. In Equation 6, α and γ are introduced to weigh 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.

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 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 Section 6.



(a) Comparison between loss functions.



(b) The influence of α towards the FLR.

Figure 1: Demonstration of different loss functions.

4.3 Hyperparameter Tuning

Compared with other loss functions, FLR loss introduces additional hyperparameters, thus it requires careful tuning. For α and γ selection, to investigate their influence as well as to reach the top performance in the considered testing scenarios, three strategies are introduced in this paper. For the first strategy, we use the values given by [LGG+17], 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.

Finally, we introduce class re-balancing into our loss function [CJL+19]. 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 Equation 4.3.

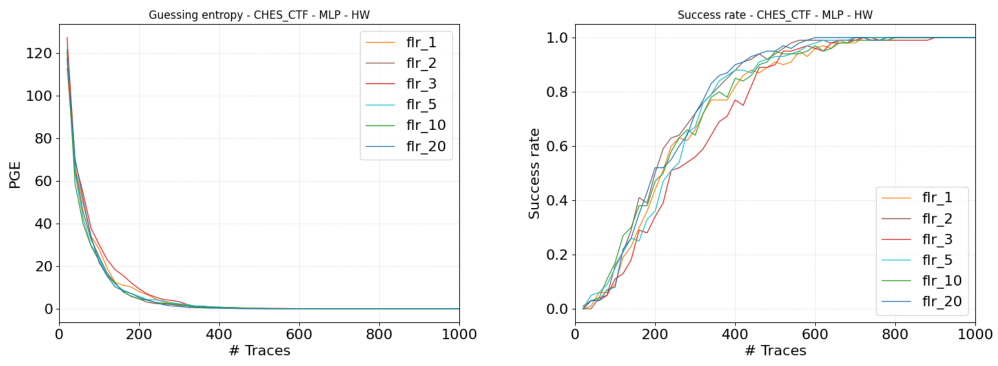
Table 1: Hyperparameter space for multi-layer perceptrons.

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

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

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 [CJL+19], we set $\beta = 0.999$. In our results, models trained with these settings are referred to as focal_balanced.

Exhaustive search is used to determine the optimal value of n. In Figure 2, we show the performance of the same model on the CHES_CTF dataset using FLR and different values of n. While the difference in $\overline{N}_{T_{GE}}$ is small, the median $\overline{N}_{T_{GE}}$ reaches the minimum 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 in comparison to $n = 1$. Therefore, we set n to three for our experiments with FLR.



(a) Guessing entropy. **(b)** Success rate.

Figure 2: Guessing entropy and success rate for FLR with different values of n.

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.

5.1 Experiment Setup

We consider the combinations of datasets, leakage models, and architecture types. Regarding model architecture tuning, we notice that 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 functions.

Algorithm 1 Model tuning and the evaluation strategy.

```
1: Generate, train, and test $M$ 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. For each loss function, we set M to be 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). While for the evaluation (step 5), we look at the guessing entropy, success rate, and training time.

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	RMSProb, Adam

Table 3: Hyperparameter space for convolutional neural networks.

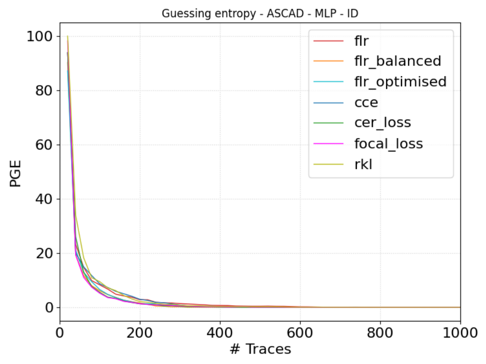
Hyperparameter	Options
Convolution layers	1 to 2 in a step of 1
Convolution filters	8 to 32 in a step of 4
Kernel size	10 to 20 in a step of 2
Pooling type	Max pooling, Average pooling
Pooling size	2 to 5 in a step of 1
Pooling stride	2 to 10 in a step of 1
Dense layers	2 to 3 in a step of 1
Neurons 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	RMSProb, Adam

Besides that, we compare the performance of models trained with FLR on an unprotected implementation of AES, then introducing masking and random desynchronization countermeasures. For this experiment, we use the ASCAD_fixed, ASCAD_plain, and ASCAD_desync50 datasets.

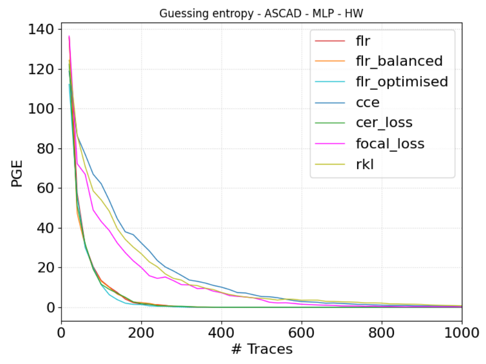
For our experiments on the ASCAD datasets, 50 000 profiling traces are used. For the CHES_CTF, we use 45 000 profiling traces. In the attacking phase, we use up to 2 000 traces for the ASCAD_fixed dataset and up to 3 000 traces for the ASCAD_variable and CHES_CTF datasets. In some of the plots, the x-axis is reduced to increase visibility.

5.2 ASCAD_fixed

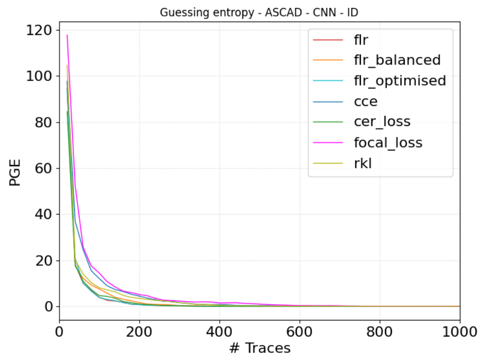
We present the results of the models with different functions on the ASCAD_fixed dataset. Figure 3 and Figure 4 show the guessing entropy and success rate with different attack models and leakage models.



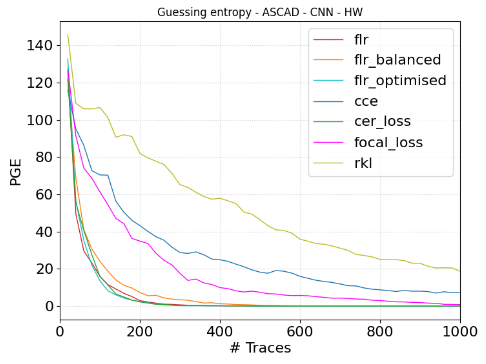
(a) MLP models, ID leakage.



(b) MLP models, HW leakage.



(c) CNN models, ID leakage.

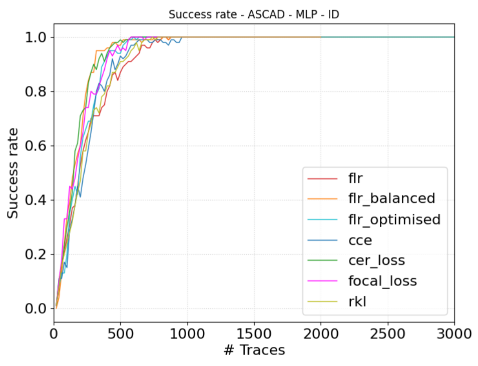


(d) CNN models, HW leakage.

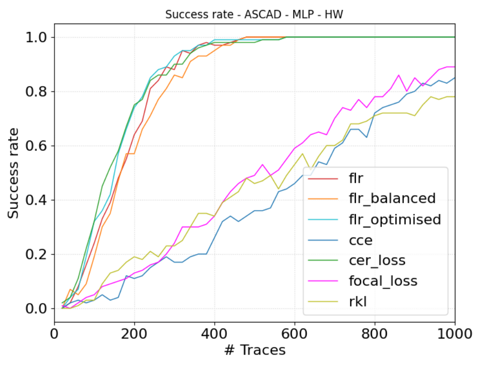
Figure 3: Guessing entropy of the optimized models on the ASCAD_fixed dataset.

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. On the other side, FLR loss and CER loss perform comparably. Still, as shown in Table 4, where the median $\overline{N}_{T_{GE}}$ are evaluated, 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 few traces. Although optimal strategy differs per scenario, their variation is limited.

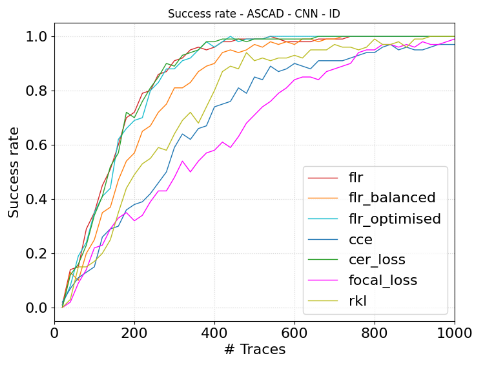
Figure 5 shows the training times for the 100 random models with each of the loss functions. Except for the CER loss, there is no significant difference between the compared loss functions. The slowdown for the CER loss is caused by choice for n (equals 10).



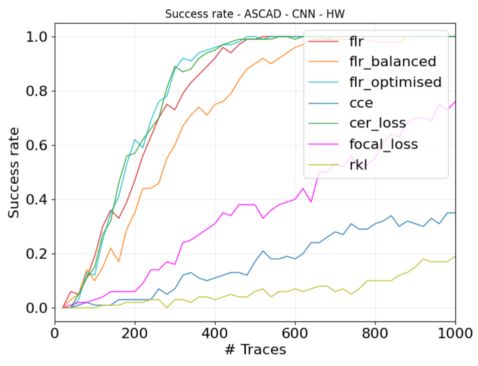
(a) MLP models, ID leakage.



(b) MLP models, HW leakage.



(c) CNN models, ID leakage



(d) CNN models, HW leakage.

Figure 4: Success rate of the optimized models on the ASCAD_fixed dataset.

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

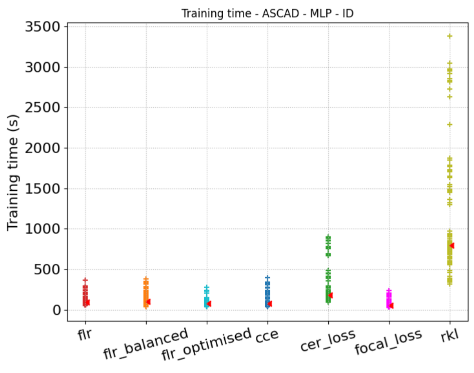
	L_{focal}	CCE	CER loss	RKL	FLR	FLR_balanced	FLR_optimized
MLP ID	580	860	570	900	810	540	680
MLP HW	1480	1560	560	1620	460	570	510
CNN ID	1250	1360	600	1760	610	850	550
CNN HW	1840	>2000	540	>2000	570	790	560

Although choosing a lower n would reduce the training time, it reduces the probability of selecting hard samples, influencing the attack performance.

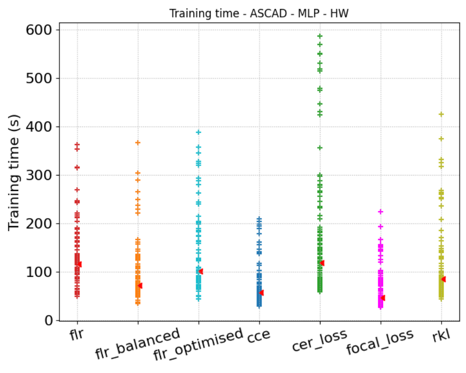
From our results on the ASCAD_fixed dataset, we can conclude that the FLR is a suitable loss function for deep learning-based SCA. It outperforms other loss functions in most cases without negative impacts on training efficiency.

5.3 ASCAD_variable

Next, loss functions are tested on the ASCAD_variable dataset. The guessing entropy of each loss function is presented in Figure 6. For the ID leakage model, neither the MLPs or CNNs reach a GE of 1 with less than 3 000 traces. Indeed, the ASCAD_variable dataset is a more difficult dataset to attack than its fixed-key counterpart. Still, the CER loss and



(a) ID leakage



(b) HW leakage

Figure 5: Training time of 100 random models per loss function on the ASCAD_fixed dataset.

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 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 $\overline{N}_{T_{GE}} = 1\,340$). While for CNN, FLR outperforms CER ($\overline{N}_{T_{GE}} = 800$ versus $\overline{N}_{T_{GE}} = 950$). In terms of ranking loss, unfortunately, it performs the worst in most of the test scenarios.

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

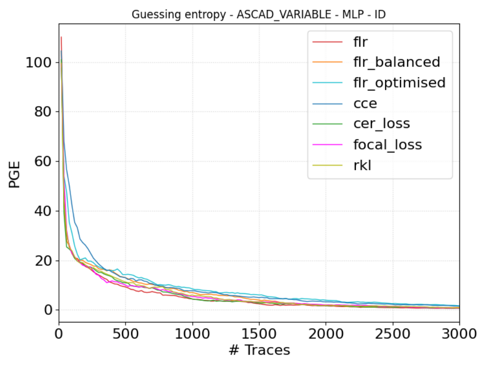
	L_{focal}	CCE	CER loss	RKL	FLR	FLR_balanced	FLR_optimized
MLP ID	>3 000	>3 000	>3 000	>3 000	>3 000	>3 000	>3 000
MLP HW	1 940	2 600	1 340	2 910	2 180	2 460	1 800
CNN ID	>3 000	>3 000	>3 000	>3 000	>3 000	>3 000	>3 000
CNN HW	>3 000	2 840	950	>3 000	880	1 670	1 020

Next, the success rates (SR) of each loss function are shown in Figure 7. Interestingly, the FLR (default version) equipped model reaches a higher SR slightly faster than the other loss functions with the ID leakage model. For the HW leakage scenarios, the FLR and CER loss perform equally well. 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 solid choice.

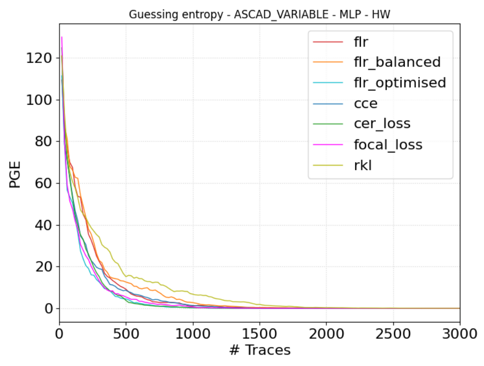
5.4 CHES_CTF

In this subsection, we discuss the results of the CHES_CTF dataset. Figure 8 shows the guessing entropy in the different scenarios and Figure 9 shows the success rates.

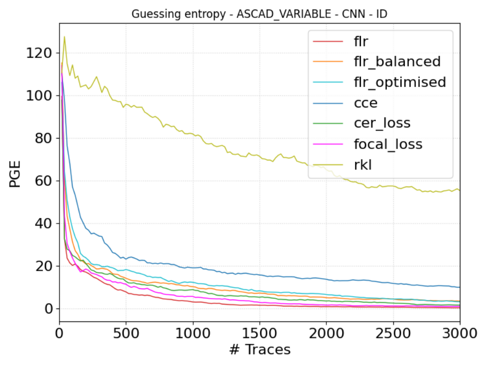
Together with the success rates presented in Figure 9, for all considered loss functions, 3 000 attack traces is not sufficient to obtain the correct key for the ID leakage model. Still, from the results, we see a significant performance improvement with the MLP models and the ID leakage when using FLR_balanced. 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 FLR_balanced performs the best with MLP, but it performs mediocre with CNN. Something similar is



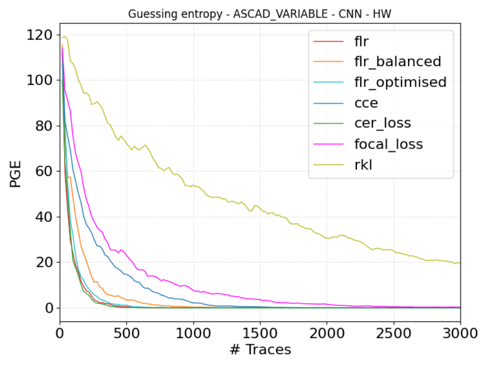
(a) MLP models, ID leakage.



(b) MLP models, HW leakage.



(c) CNN models, ID leakage.



(d) CNN models, HW leakage.

Figure 6: Guessing entropy of the optimized models on the ASCAD_variable dataset.

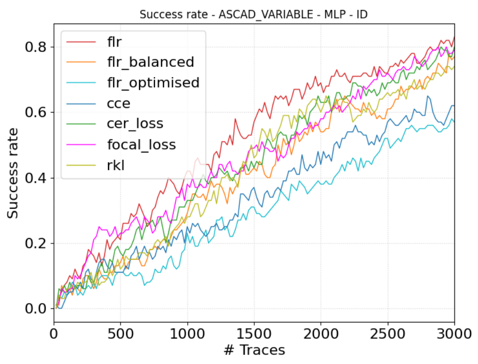
also visible for the FLR_optimized.

When the HW leakage 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 models that successfully retrieved the correct key with a CNN model. The median out 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, and the performance is approximately equal to the CER loss.

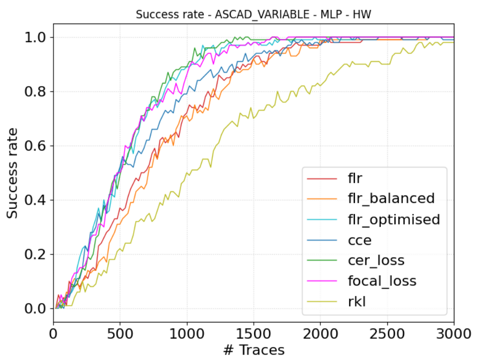
Table 6: Median $\overline{N}_{T_{GE}}$ on the CHES_CTF dataset. The lowest $\overline{N}_{T_{GE}}$ for each scenario is marked blue.

	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	>3 000
CNN HW	>3 000	>3 000	>3 000	>3 000	2 740	>3 000	2 000

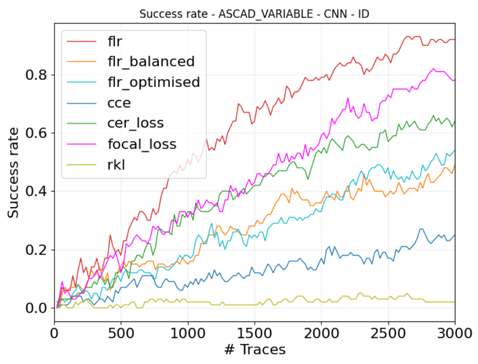
We can conclude that the FLR is a good candidate when attacking difficult datasets as the performance is better than the CER loss and ranking loss in most cases. Additionally, FLR outperforms conventional loss functions such as categorical cross-entropy. It does so without increasing the training efficiency.



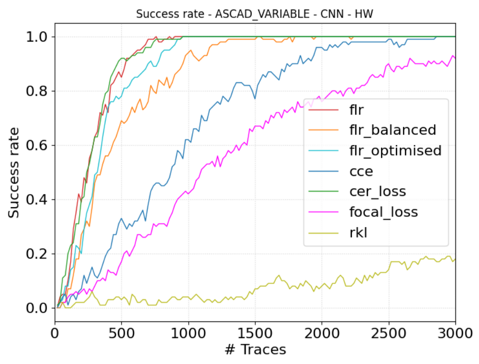
(a) MLP models, ID leakage.



(b) MLP models, HW leakage.



(c) CNN models, ID leakage.



(d) CNN models, HW leakage.

Figure 7: Success rate of the optimized models on the ASCAD_variable dataset.

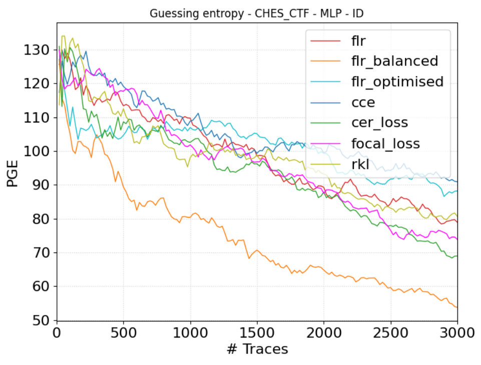
5.5 Countermeasures

Finally, we look at the difference in the performance of the different loss functions when considering unprotected implementations and countermeasures. Figure 10 shows the guessing entropy on the ASCAD_plain dataset (where we assume that the mask is known).

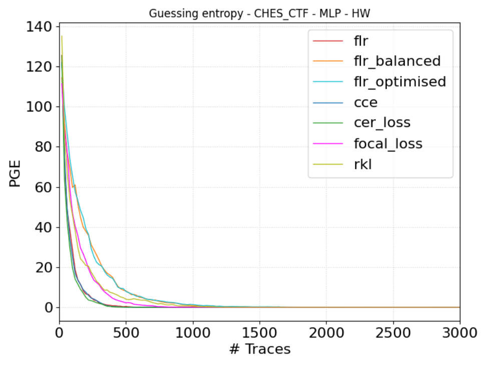
As visible from the GE plots, the unprotected dataset is easily broken. Every test scenario retrieves the correct key byte trivially with less than ten traces. The models trained with the FLR perform slightly better, within every scenario a median $\overline{N}_{T_{GE}}$ of 6 or 7, while the other loss functions often need 8 or more traces.

If we look at the impact that adding masks has by comparing these results with the models in Figure 3, we see the largest increase $\overline{N}_{T_{GE}}$ for the models with focal loss or categorical cross-entropy. This shows that the FLR and CER loss are slightly more resilient to a countermeasure such as masking.

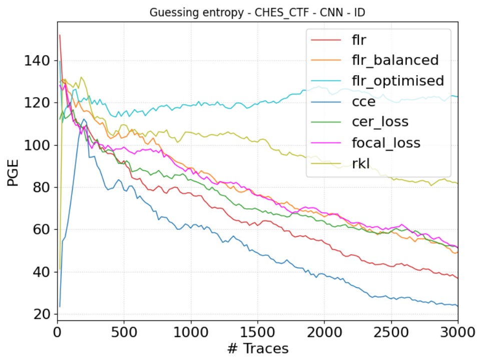
The trace desynchronization decreases the performance of every model drastically. Figure 11 shows the GE of the models on the ASCAD_desync50 dataset. Similar to before, the MLPs perform poorly, and none of the loss functions results in a successful attack. When we look at the performance of the CNNs, we see that when the ID leakage model is used, the FLR and CER loss functions are not very successful. None of the models equipped with these functions leads to a successful attack, while only the successful model is trained with the categorical cross-entropy. Indeed, trace desynchronization significantly increases the cross-entropy between the predicted and ground-truth labels. When using the ratio loss such as FLR and CER, a huge loss value would be used for gradient descent



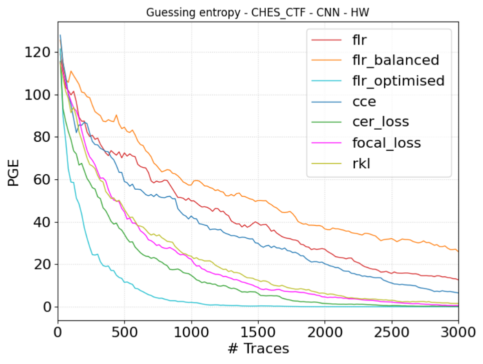
(a) MLP models, ID leakage.



(b) MLP models, HW leakage.



(c) CNN models, ID leakage.



(d) CNN models, HW leakage.

Figure 8: Guessing entropy of the optimized models on the CHES_CTF dataset.

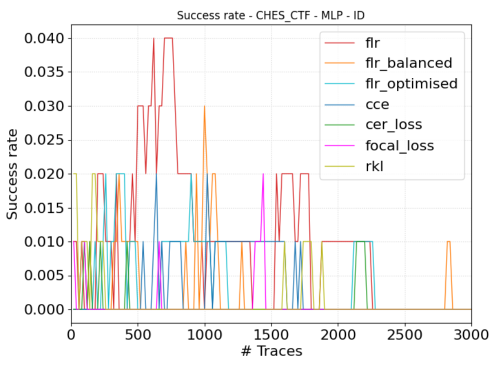
and may cause the model difficult to converge. Naturally, reducing the learning rate could be a good choice the deal with this problem. Interestingly, this effect is not visible when the HW leakage model is considered. In that case, the FLR functions outperform every other function, requiring less than half the number of traces compared to the categorical cross-entropy (1 250 versus 2 660). CER loss and focal loss do not reach a GE of 1 when using 3 000 traces or less to attack. It is important to note that the FLR loss function performs well while not incurring additional computational complexity in the calculation process.

Finally, we can conclude that the FLR loss is a robust solution in presence of countermeasures. As expected, there is an increase in the required number of traces for a successful attack, but this increase is smaller when compared to other loss functions. When random desynchronization is introduced, the losses with a ratio perform poorly when the ID leakage model is used. When the HW leakage model is considered, models with FLR outperform the other loss functions by a large margin.

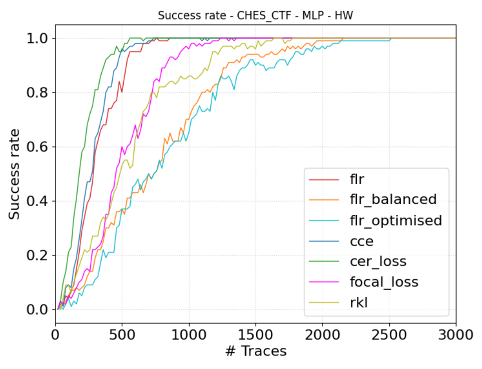
6 Discussion

FLR loss performs well in various test scenarios. The only downside to using FLR as a loss function is the introduction of the α and γ parameters. In our experiments, we used three different strategies: IN

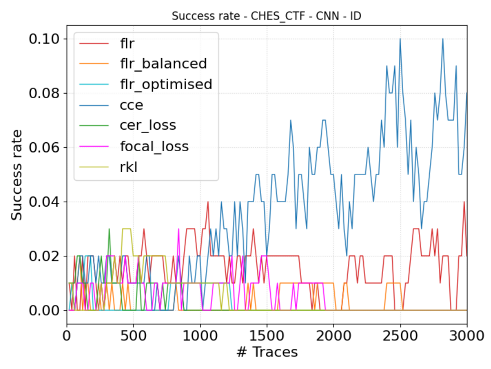
- Fixed value: $\alpha = 0.25$ and $\gamma = 2.0$.



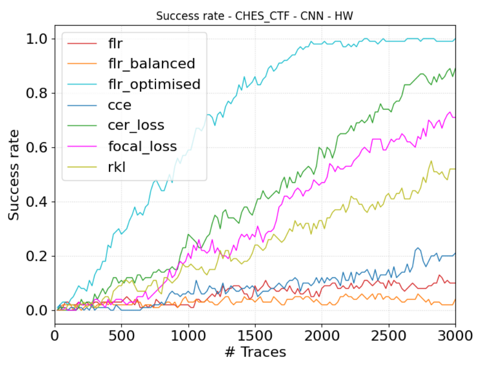
(a) MLP models, ID leakage.



(b) MLP models, HW leakage.



(c) CNN models, ID leakage.



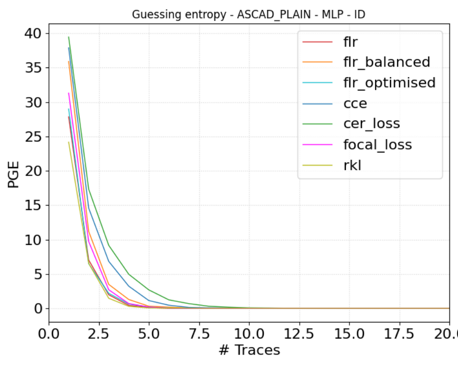
(d) CNN models, HW leakage.

Figure 9: Success rate of the optimized models on the CHES_CTF dataset.

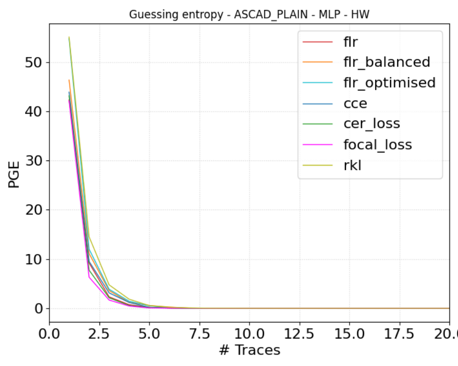
- Optimized via random search.
- Determined by the frequency of each class.

Throughout the experiments, there was not a single strategy that worked best in every scenario. However, in almost all cases, the best performing FLR variants have the fixed α values for every class. 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 where 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. Indeed, from Section 4.2, one should note that with lower α, the samples that trigger high loss value are the ones misclassified with high confidence (probability).

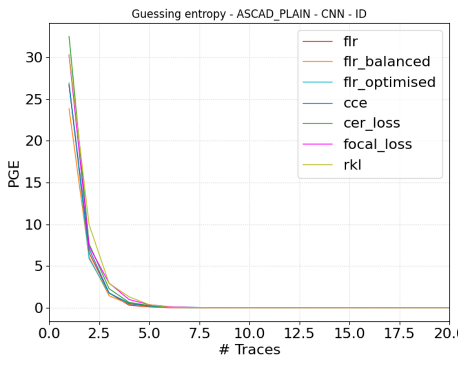
Compared with other loss functions that require models to be confident about predicting, this FLR configuration softens the restriction for the predictions: only (very) hard negative will be penalized, while the others that are correctly classified, or even misclassified but with low confidence would have limited loss contributions. From the learning perspective, loss functions that force 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, thus increasing the generality of the model and helping to learn from the imbalanced data.



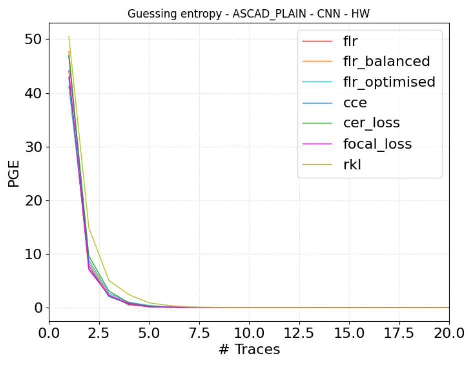
(a) MLP models, ID leakage.



(b) MLP models, HW leakage.



(c) CNN models, ID leakage.



(d) CNN models, HW leakage.

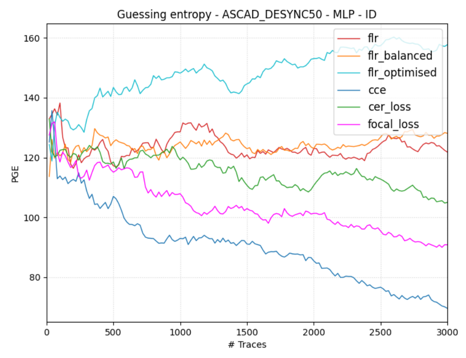
Figure 10: Guessing entropy of the optimized models on the ASCAD_plain dataset.

To test our hypothesis, we performed an additional set of experiments on the 12 test scenarios. The search space for α enlarges 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 Table 7 and Table 8.

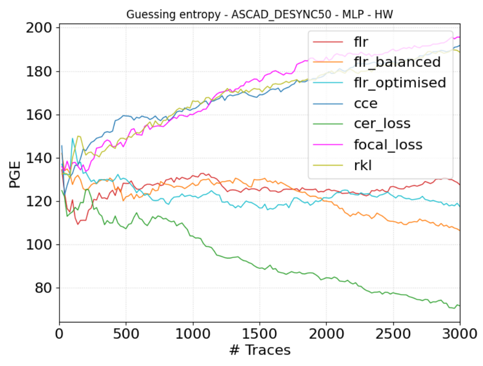
Table 7: Median $\overline{N}_{T_{GE}}$ on the ASCAD_fixed dataset. The lowest $\overline{N}_{T_{GE}}$ for each scenario is marked blue.

	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

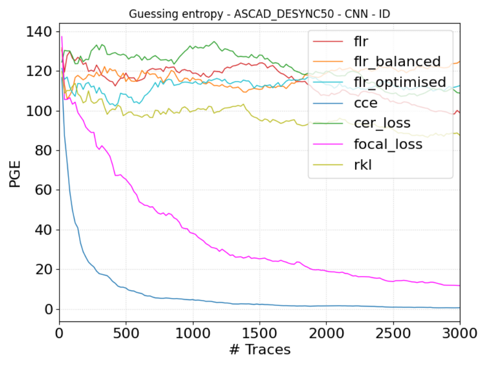
These results confirm our hypothesis. In the scenarios in which the class balanced FLR was previously best, such as the ASCAD_fixed scenarios, the FLR with our new strategy still performs very well. For instance, when attacking ASCAD_fixed with MLP and the ID leakage model, the best performing model 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



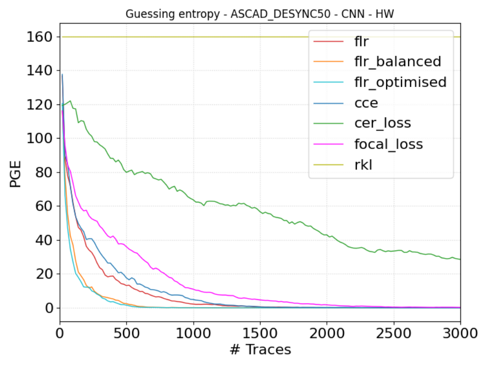
(a) MLP models, ID leakage.



(b) MLP models, HW leakage.



(c) CNN models, ID leakage.



(d) CNN models, HW leakage.

Figure 11: Guessing entropy of the optimized models on the ASCAD_desync50 dataset.

Table 8: Median $\overline{N}_{T_{GE}}$ on 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

Table 9: Median $\overline{N}_{T_{GE}}$ on the CHES_CTF 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	1 220	630	480	1 860	1 080
CNN ID	>3 000	>3 000	>3 000	>3 000	>3 000
CNN HW	>3 000	>3 000	>3 000	>3 000	2 070

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 that is optimized for deep learning-based side-channel analysis. More precisely, we started by identifying the pros and cons of several loss functions in the context of SCA. Using those characteristics, we constructed a new loss function for deep learning-based SCA called the focal loss ratio (FLR). By testing FLR on various combinations of datasets, leakage models, and neural network architectures, we confirmed the outstanding performance of FLR. 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.

References

[BPS+20] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. Study of Deep Learning Techniques for Side-Channel Analysis and Introduction to ASCAD Database-Long Paper. *Journal of Cryptographic Engineering*, 10(2):163–188, 2020.

[CDP17] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional Neural Networks with Data Augmentation against Jitter-Based Countermeasures-Profiling Attacks without Pre-Processing. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 45–68, 2017.

[CJL+19] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge J Belongie. Class-Balanced Loss Based on Effective Number of Samples. *CoRR*, abs/1901.05555, 2019.

[GBC16] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. http://www.deeplearningbook.org.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition, 2015.

[KLSS17] Nataliia Kussul, Mykola Lavreniuk, Sergii Skakun, and Andrey Shelestov. Deep Learning Classification of Land Cover and Crop Types Using Remote Sensing Data. *IEEE Geoscience and Remote Sensing Letters*, PP:1–5, 7 2017.

[KPH+19] Jaehun Kim, Stjepan Picek, Annelie Heuser, Shivam Bhasin, and Alan Hanjalic. Make Some Noise Unleashing the Power of Convolutional Neural Networks for Profiled Side-channel Analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems ISSN 2569-2925*, 2019(3):148–179, 2019.

[KWPP21] Maikel Kerkhof, Lichao Wu, Guilherme Perin, and Stjepan Picek. 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.

[LGG+17] Tsung-Yi Lin, Priya Goyal, Ross B Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. *CoRR*, abs/1708.02002, 2017.

[MDP19] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. A Comprehensive Study of Deep Learning for Side-Channel Analysis, 2019.

[MOP06] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, December 2006. ISBN 0-387-30857-1, http://www.dpabook.org/.

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

[MWM21] Thorben Moos, Felix Wegener, and Amir Moradi. Dl-la: Deep learning leakage assessment: A modern roadmap for sca evaluations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(3):552–598, Jul. 2021.

[PBP20] Guilherme Perin, Ileana Buhan, and Stjepan Picek. 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.

[PCP20] Guilherme Perin, Lukasz Chmielewski, and Stjepan Picek. Strength in numbers: Improving generalization with ensembles in machine learning-based profiled side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(4):337–364, Aug. 2020.

[PHJ$^+$18] Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni. The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(1):209–237, Nov. 2018.

[PSK$^+$18] Stjepan Picek, Ioannis Petros Samiotis, Jaehun Kim, Annelie Heuser, Shivam Bhasin, and Axel Legay. On the performance of convolutional neural networks for side-channel analysis. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 157–176. Springer, 2018.

[RWPP21] Jorai Rijsdijk, Lichao Wu, Guilherme Perin, and Stjepan Picek. Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(3):677–707, Jul. 2021.

[RZC$^+$21] Damien Robissout, Gabriel Zaid, Brice Colombier, Lilian Bossuet, and Amaury Habrard. Online performance evaluation of deep learning networks for profiled side-channel analysis. In Guido Marco Bertoni and Francesco Regazzoni, editors, *Constructive Side-Channel Analysis and Secure Design*, pages 200–218, Cham, 2021. Springer International Publishing.

[SJP03] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template Attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2002*, pages 13–28, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[SMY09] François-Xavier Standaert, Tal G. Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009*, pages 443–461, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[Tim19] Benjamin Timon. Non-Profiled Deep Learning-based Side-Channel attacks with Sensitivity Analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(2):107–131, 2 2019.

[WAGP20] Lennert Wouters, Victor Arribas, Benedikt Gierlichs, and Bart Preneel. Revisiting a Methodology for Efficient CNN Architectures in Profiling Attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(3):147–168, 2020.

[WP20] Lichao Wu and Stjepan Picek. Remove some noise: On pre-processing of side-channel measurements with autoencoders. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 389–415, 2020.

[WWJ$^+$21] Lichao Wu, Yoo-Seung Won, Dirmanto Jap, Guilherme Perin, Shivam Bhasin, and Stjepan Picek. Explain some noise: Ablation analysis for deep learning-based physical side-channel analysis. *IACR Cryptol. ePrint Arch.*, page 717, 2021.

[YWL$^+$20] Baoguo Yuan, Junfeng Wang, Dong Liu, Wen Guo, Peng Wu, and Xuhua Bao. Byte-level malware classification based on markov images and deep learning. *Computers & Security*, 92:101740, 2020.

[ZBD$^+$20] Gabriel Zaid, Lilian Bossuet, François Dassance, Amaury Habrard, and Alexandre Venelli. Ranking loss: Maximizing the success rate in deep learning side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(1):25–55, Dec. 2020.

[ZBHV19] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Methodology for efficient cnn architectures in profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(1):1–36, Nov. 2019.

[ZYJY18] Donglai Zhu, Hengshuai Yao, Bei Jiang, and Peng Yu. Negative Log Likelihood Ratio Loss for Deep Neural Network Classification. 4 2018.

[ZZN$^+$20] Jiajia Zhang, Mengce Zheng, Jiehui Nan, Honggang Hu, and Nenghai Yu. A novel evaluation metric for deep learning-based side channel analysis and its extended application to imbalanced data. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 73–96, 2020.