Beyond the Last Layer: Deep Feature Loss Functions in Side-channel Analysis

Trevor Yap
trevor.yap@ntu.edu.sg
Nanyang Technological University
Singapore

Stjepan Picek
picek.stjepan@gmail.com
Radboud University
Nijmegen, The Netherlands
Delft University of Technology
Delft, The Netherlands

Shivam Bhasin
sbhasin@ntu.edu.sg
Nanyang Technological University
Singapore

ABSTRACT

This paper provides a novel perspective on improving the efficiency of side-channel analysis by applying two deep feature loss functions: Soft Nearest Neighbor (SoftNN) and Center loss. By leveraging these loss functions during the deep neural networks (DNNs) training phase, our study illuminates how profiling attacks can be more powerful. Deep feature loss functions incorporate the outputs from the DNN's intermediate layers into their computations, which reduces the distance between similar data points. As such, these techniques enhance the DNN's ability to generate more precise and meaningful representations, thereby improving its discriminative power. This paper presents empirical evidence illustrating the effectiveness of SoftNN and Center loss in strengthening DNN-based side-channel attacks. For instance, when using Center loss together with the focal loss ratio (FLR), it requires the least number of traces to break the ASCADf dataset. On the other hand, applying SoftNN with FLR successfully recovers the key for the ASCADr dataset with the least traces. The insights presented in this study can act as a baseline for more advanced investigations into the utility of such loss functions in deep learning-based side-channel analysis.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; • **Security and privacy** → **Side-channel analysis and countermeasures**.

KEYWORDS

Side-channel analysis, Neural Network, Deep Learning, Profiling attack, Loss function, Deep Features

ACM Reference Format:
Trevor Yap, Stjepan Picek, and Shivam Bhasin. 2023. Beyond the Last Layer: Deep Feature Loss Functions in Side-channel Analysis. In *Proceedings of the 2023 Workshop on Attacks and Solutions in Hardware Security (ASHES '23), November 30, 2023, Copenhagen, Denmark.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3605769.3623996

ASHES '23, November 30, 2023, Copenhagen, Denmark

ACM ISBN 979-8-4007-0262-4/23/11.
https://doi.org/10.1145/3605769.3623996

1 INTRODUCTION

Side-channel analysis (SCA) exploits leakages stemming from physical implementations instead of the weaknesses of underlying cryptographic algorithms. Power consumption and electromagnetic emanation are common physical leakages that can be easily obtained and analyzed. A commonly used side-channel analysis approach is known as the profiling attack. Profiling attacks are usually considered the most powerful SCAs that can even break implementations with side-channel countermeasures like masking or desynchronization. In recent years, the usage of deep neural networks (DNNs) has gained much attention due to their ability to recover the secret key without preprocessing [2, 12], which pave the way for deep learning-based side-channel analysis (DLSCA).

Hyperparameter tuning is crucial in obtaining well-performing neural network models to retrieve the secret key. One such hyperparameter to consider is the type of loss function, as it plays a central role in the training of DNN. The loss function calculates the loss between the actual and desired labels. From this loss, the weights of the DNN are updated accordingly. Therefore, the type of loss function determines the success of breaking the implementation. While some works have proposed custom loss functions for SCA [6, 22, 24], the metrics from the loss functions do not necessarily give an accurate representation of whether the trained model can break the implementation [14, 23]. Furthermore, executing a practical attack usually considers guessing entropy evaluation, which is computationally costly. Therefore, more attention is needed to develop potent loss functions to train the DNN for SCA.

Deep features are the features/embeddings acquired from the output of each intermediate layer within the DNN. In [13], Perin et al. explored how the perceived information of the sensitive variables and irrelevant sample points is processed in each intermediate layer[1]. For a well-performing DNN, as deeper the DNN's layers go, the irrelevant information is compressed, and simultaneously, the information about the sensitive variable increases. This shows that deep features contain crucial information in a well-trained DNN that manages to recover the secret key. Furthermore, recently [20] proposed a novel distance metric known as Hybrid Distance that is used within the Triplet loss to train a Triplet network. The authors used this Triple network and loss function to process the traces into the embedding space before using these embeddings for a template attack. They showed that by doing this, the performance of template attacks improves drastically. Inspired by the above

[1]The authors explored how first-order masking, where the secret variable is split into two shares, is processed within a neural network.

works, we ask the following question: *How efficient would the side-channel attack be if we incorporate deep features into the loss function of the DNN in a typical profiling attack?* Consequently, we explore the effectiveness of deep feature loss functions proposed by the deep learning community, but now, for DLSCA.

Our Contributions. Our main contributions can be summarized as follows:

(1) We present the visualization of how two deep feature loss functions, namely Soft Nearest Neighbor (SoftNN) and Center loss, decrease the distance between similar data points for the Chipwhisperer dataset.
(2) We provide empirical results to depict the effectiveness of SoftNN and Center loss in DLSCA. We benchmarked with commonly used loss functions found in SCA on various datasets. In the best case, SoftNN with focal loss ratio (FLR) required the least traces to break ASCADr. Center loss with FLR similarly could reach the best results. Indeed, the DNN trained with this loss function required the least traces to retrieve the key for ASCADf. We further conducted experiments on desynchronized traces. In three out of four datasets protected by desynchronization, Center loss required either the least traces or reached the best median number of traces to recover the key. As for SoftNN with FLR, it obtained the best median number of traces to break ASCADf with a desynchronization level of 50 (ASCADf_desync50).

The source code for our models and results can be obtained from https://github.com/yap231995/Deep_Features_Loss_function_SCA.

Paper Organization. The structure of the paper is as follows. Section 2 provides the necessary background on profiling attacks and presents the mathematical formulation of all the loss functions used. The prior works on loss functions within the realm of DLSCA are discussed in Section 3. Next, we provide visualizations of how SoftNN and Center loss affect the deep features of a DNN in Section 4. This helps to illustrate how each loss function impacts the intermediate layers of a DNN. Section 5 presents the datasets and experimental framework used. We experimentally validate the effectiveness of SoftNN and Center loss in Section 6. Lastly, we conclude our work and provide future research directions in Section 7.

2 BACKGROUND

2.1 Profiling Attacks

Profiling attacks are considered the worst-case scenario in SCA as they allow the most powerful attacker. Profiling attacks consist of profiling and an attack phase. The adversary has two devices in this setup: a prototype (clone) device and a target device. The adversary either has control over the key or knows the key of the prototype device, but the key of the target device is unknown to the adversary. In the profiling phase, a distinguisher $\mathcal{F}$ is built by taking profiling traces from the prototype. Using the distinguisher $\mathcal{F}$, a score $y_i = \mathcal{F}(\boldsymbol{t}_i)$ is obtained for each attack trace $\boldsymbol{t}_i$ attained from the target device. The log-likelihood score is then calculated for all the key candidates k for a fixed number of attack traces N_a: $s_{N_a}(k) = \sum_{i=1}^{N_a} log(y_i[z_{i,k}])$ where $z_{i,k} = C(p_i, k)$ is the intermediate sensitive variable of the cryptographic primitive C based on the key k and public variable p_i of the traces $\boldsymbol{t}_i$. We sort the log-likelihood score of each key in a guessing vector $G = [G_0, G_1, \ldots, G_{|\mathcal{K}|-1}]$ with G_0 corresponding to the score of the most likely key candidate while $G_{|\mathcal{K}|-1}$ is the score of the least likely key candidate. The rank of the key is indicated as the index of the guessing vector $\boldsymbol{G}$. The guessing entropy GE is defined as the average rank of the correct key k^* for a fixed number of experiments. The attack is successful if $GE = 0$ (or some sufficiently small value) and the smallest number of traces required to obtain $GE = 0$ is denoted as $NTGE$. In DLSCA, DNN is trained as a distinguisher $\mathcal{F}$. The typical DNN architectures used in SCA are the Multilayer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs).

2.2 Baseline Loss Functions

This section presents two baseline loss functions $\mathcal{L}_{base}$ that have been used in SCA. The first baseline loss function we considered is the commonly used categorical cross-entropy (CCE), and the second baseline loss function is called the focal loss ratio (FLR), which is customized for the SCA domain. Here, we define B as the batch size specified for training the DNN.

2.2.1 Categorical Cross Entropy. CCE is the most common loss function for different classification tasks [5, 8, 21]. It measures the distance between two distributions and is used to train DNN by minimizing

$$CCE(y, \hat{y}) = \frac{1}{B} \sum_{i=1}^{B} \sum_{j=1}^{c} y_{i,j} log(\hat{y}_{i,j}).$$

In DLSCA, CCE is the most used loss function [15]. According to [7], it is generally preferred over the custom loss functions used in SCA.

2.2.2 Focal Loss Ratio. FLR is proposed in [6] as an improvement over the custom loss function known as the cross-entropy ratio (CER) [24] to tackle imbalanced data. The authors added weights to CER based on focal loss [9]. FLR can be calculated as

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

where y is the true label and y_s is the shuffled label. Here, we set $\alpha = 0.25$ and $\gamma = 2.0$ as recommended in [6]. Furthermore, our experiments use $n = 10$ as explored in [6] and [24].

2.3 Deep Features Loss Functions

We present two loss functions that use the deep features of the DNN: Soft Nearest Neighbor (SoftNN) and Center loss. These can be used with the above baseline loss function to train the DNN.

2.3.1 Soft Nearest Neighbour. SoftNN is first presented in [17]. Subsequently, a new hyperparameter called temperature T is introduced into the formula from [4] to control the relative importance of the distance between a pair of points. The soft nearest neighbor

loss at temperature T is defined as

$$\mathcal{L}_{SoftNN}(x,y) = -\frac{1}{B}\sum_{i=1}^{B} log\left(\frac{\sum\limits_{\substack{j=1\\ i\neq j\\ y_i=y_j}}^{B} e^{-\frac{\|x_i-x_j\|_2^2}{T}}}{\sum\limits_{\substack{j=1\\ i\neq j}}^{B} e^{-\frac{\|x_i-x_j\|_2^2}{T}}}\right). \quad (1)$$

SoftNN characterizes how close a pair of data points of the same class/label is relative to other data points. In other words, SoftNN compares both inter-class and intra-class distances between the data points. The temperature can be interpreted as the variance of the Gaussian distribution when calculating the probability of the sampling neighboring points. At lower temperatures, the distances between closely related representations dominate the SoftNN loss, while widely separated representations provide little impact on SoftNN.

In [17], Frosst et al. considered the SoftNN as a regularizer with the baseline loss function CCE to train the DNN. The SoftNN is computed for every deep feature $f^{(l)}$ of the DNN. Then, the following loss is minimized:

$$\mathcal{L}(f,x,y) = \mathcal{L}_{base} + \lambda \sum_{l=1}^{L-1} \mathcal{L}_{SoftNN}(x, f^l(x)), \quad (2)$$

where f^1 is the first layer, f^L is the logit layer (i.e., the layer that feeds into the softmax), and λ controls the influences from SoftNN during training.

2.3.2 Center Loss. In [18], the authors proposed the Center loss function to enhance the discriminative power within the deep features of the DNN. The Center loss function could simultaneously learn the center of a deep feature for each class while penalizing the distance between the deep features and its class center. The Center loss is computed as

$$\mathcal{L}_{Center}(x) = \frac{1}{2}\sum_{i=1}^{B} \left\|x_i - c_{y_i}\right\|_2^2, \quad (3)$$

where c_{y_i} is the center of the class y_i. The center can be learned for each epoch by updating the centers in epoch $t+1$ through

$$c_y^{t+1} = c_y^t - \kappa \Delta c_y^t$$

for each class y where κ controls the rate at which it updates the centers and

$$\Delta c_y^t = \frac{\sum_{i=1}^{B} \mathbb{1}_{y_i=y}(c_y - x_i)}{1 + \sum_{i=1}^{B} \mathbb{1}_{y_i=y}}.$$

The authors used the Center loss together with the baseline loss function CCE to train the DNN. They minimized the following loss function:

$$\mathcal{L}(f,x,y) = \mathcal{L}_{base} + \lambda \mathcal{L}_{Center}(f^{L-1}(x)), \quad (4)$$

with λ controlling the influences of Center loss. We note that Center loss considers only intra-class distance, unlike SoftNN. Therefore, it leaves the job of comparing inter-class to $\mathcal{L}_{base}$.

3 RELATED WORKS

To reach optimal performance, hyperparameter tuning to find well-performing architecture is essential. For example, [23] first proposed a methodology to construct such architectures. On the other hand, [19] and [16] explored the usage of automated neural architecture search in SCA domains. While such works consider various hyperparameters, loss functions are commonly not investigated (i.e., there is a single fixed loss function to be used). CCE is the most common loss function used in SCA [15]. Masure et al. further confirmed that CCE is related to Perceived Information, which is the lower bound of the Mutual Information between the sensitive intermediate and the leakage [10].

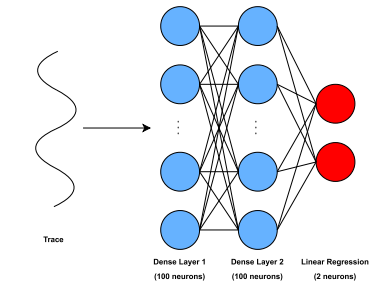
Recently, novel loss functions customized for the SCA domain have been proposed. Zaid et al. developed a custom loss function known as Ranking Loss that uses a sigmoid function to compare pairwise the correct key and the different key hypotheses to maximize the success rate [22]. Another custom loss function called the CER loss function is presented by Zhang et al. and it represents the ratio between the CCE of profiling traces with its original labels and a set of profiling traces with shuffled labels [24]. In [7], Kerkhof et al. systematically compared the above loss functions with other "traditional" machine learning loss functions, e.g., mean square error. They concluded that CER is the best loss function, followed by CCE. Interestingly, they reported that Ranking Loss did not perform as well as expected but only worked well in certain architectures. [6] introduced FLR, which enhances CER by incorporating weights to address the class imbalance in the dataset further, leading to improved results compared to CER. Nonetheless, none of the loss functions above consider deep computation features.

4 VISUALIZATION OF DEEP FEATURE LOSS FUNCTION

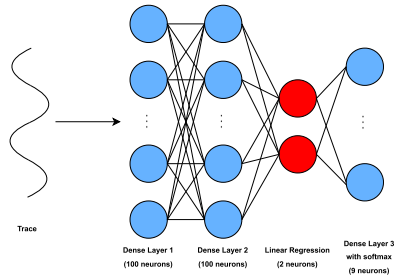
In this section, we visualize how two deep feature loss functions affect the deep features using a simple MLP. The architectures of the MLP for SoftNN and Center loss are illustrated in Figure 1. We use the ReLU activation function in dense layers 1 and 2 and run the experiments with a batch size of 512 on Adam with a learning rate of $5e-5$. The experiments are conducted on the Chipwhisperer dataset [11]. This dataset runs the unprotected AES-128 implementation on the Chipwhisperer CW308 Target and targets the first byte in the first round of the AES substitution box with a fixed key. We use 10000 traces for training and 2000 traces for visualization.

For SoftNN, the last layer is a linear regression that outputs a two-dimensional embedding, allowing us to directly plot the features on a two-dimensional surface for visualization (see Figure 1a). We train this MLP solely with SoftNN without any baseline loss function. In other words, we train the MLP with Eq. (1). The visualization is depicted in Figure 2. Since SoftNN compares inter-class and intra-class distances of the traces, we observe from Figure 1 that SoftNN helps to increase the distances between classes and decrease the distances for samples in the same class simultaneously.

On the other hand, since Center loss only minimizes the intra-class distance, we modify the MLP by adding another dense layer with softmax activation function (see Figure 1b). Therefore, we train this MLP with CCE as the baseline loss function together with Center loss as the deep feature loss function presented in Eq. (4).

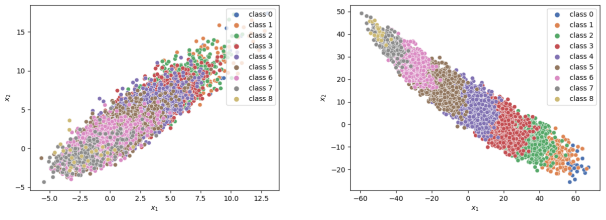


(a) SoftNN



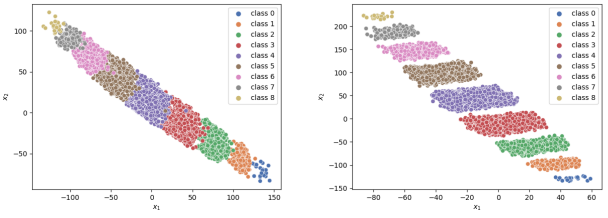
(b) CCE+Center Loss

Figure 1: DNN Architecture used for visualization.



(a) Epoch 1
Loss: 1.8064

(b) Epoch 5
Loss: 0.7379

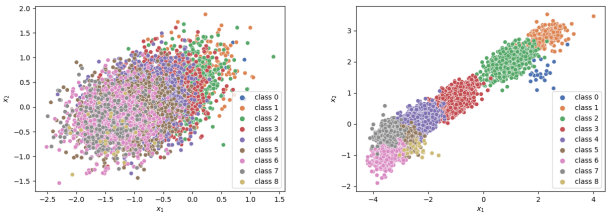


(c) Epoch 25
Loss: 0.2029

(d) Epoch 50
Loss: 0.0848

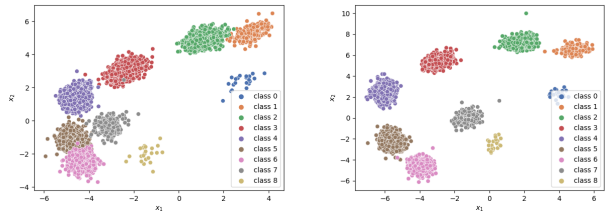
Figure 2: Visualization of the class separation when using SoftNN.

This means that the deep features that the Center loss is applied to are the two neurons in red shown in Figure 1. The visualization of these two red neurons when using Center loss for training is illustrated in Figure 3. If we compare the visualization of the deep features when the training is without Center loss (see Figure 12 in Appendix A), we see that the Center loss further minimizes the intra-class distances as the values of using CCE only are higher.



(a) Epoch 1
Loss: 2.5606

(b) Epoch 10
Loss: 1.5819



(c) Epoch 30
Loss: 1.0319

(d) Epoch 50
Loss: 0.6738

Figure 3: Visualization of the separation when using Center loss together with CCE.

5 EXPERIMENTAL SETTING

5.1 Datasets

We consider three publicly available datasets running the Advanced Encryption Scheme (AES) [3], representing common scenarios faced in DLSCA. We focus on attacking a single byte of the secret key. Furthermore, we investigate two common hypothetical leakage models: the Identity (ID) and the Hamming Weight (HW) leakage models.

ASCAD. The ASCAD dataset is a first-order masked AES implementation on an 8-bit AVR microcontroller (ATMega8515) [1]. We target the third byte of the first round AES substitution box since that is the first masked key byte. The dataset contains two versions known as ASCADf and ASCADr. ASCADf consists of traces built with a fixed key for profiling and attack. On the other hand, ASCADr contains traces for profiling that are attained from a random key setting, while the traces for the attack phase are from the target device with a fixed key. We use 45000 traces for profiling for both datasets. For the attack phase, 2000 attack traces for ASCADf and 10000 attack traces for ASCADr are used. ASCADf contains traces with 700 sample points, while the traces in ASCADr are composed of 1400 sample points.

CHES_CTF. At the Conference on Cryptographic Hardware and Embedded Systems (CHES) in 2018, a dataset called CHES_CTF was released. The CHES_CTF dataset consists of traces from running a first-order masked AES on a 32-bit STM microcontroller. It can be found at https://chesctf.riscure.com/2018/news. We use 45000

traces for profiling and 3000 traces for attack. Both profiling and attack traces consist of different fixed keys, unlike the ASCADf dataset. We attack the first byte of the key. Each trace contains 2200 sample points.

5.2 DNN Architecture and Training Setting

Due to the random weight initialization, the model's performance may fluctuate with each training. Therefore, we follow the framework from [6] to evaluate the effectiveness of each loss function tested. The framework is presented in Algorithm 1.

Algorithm 1 Framework to evaluate loss function

1: Generate randomly 100 different models.
2: Train these models on all the different loss functions.
3: Use these 100 trained models for the attack phase.
4: Select the best models M_{best} with the lowest $NTGE$.
5: Repeatedly train M_{best} for 10 times.
6: Record the best $NTGE$ as $NTGE_{best}$ and the median $NTGE$ of the 10 M_{best} as $NTGE_{median}$.

The hyperparameter search space to generate the 100 DNN architectures is presented in Table 1. In our case, we set the pooling stride equal to the pooling size. We train the DNN with 50 epochs.

Hyperparameter	Options
MLP	
Number of Dense Layers	1 to 8 in a step of 1
Neurons per layer	10, 20, 50, 100, 200, 300, 400, 500
CNN	
Convolution layers	1 to 4 in step of 1
Convolution filters	4 to 16 in step of 4
Kernel size	26 to 52 in step of 2
Pooling type	Average or Max
Pooling size	2 to 10 in step of 2
Number of Dense Layers	1 to 4 in a step of 1
Neurons per layer	10, 20, 50, 100, 200, 300, 400, 500
Others	
Batch size	300 to 1100 in a step of 100
Activation function	*ReLU* or *SeLU*
Optimizer	Adam or RMSprop
Learning Rate	$0.0005, 0.0001, 1e-4, 5e-4$
Weight Initializer	Random Uniform or Glorot Uniform or He Uniform

Table 1: Hyperparameter search space.

When introducing deep feature loss function into the training, it introduces additional hyperparameters to tune. First, the hyperparameter λ defines how much influence the deep features loss function has over training. As for SoftNN and Center loss, each has one additional hyperparameter to tune. SoftNN requires the hyperparameter called temperature T. With initial testing, temperature T with values less than 100 resulted in unstable training. Furthermore, the lower the temperature, the greater the influence of the intra-distance between data points over the SoftNN loss. Therefore, we consider values from 100 to 495 to be randomly tested. On the other hand, the hyperparameter κ needed tuning when Center loss is used. *kappa* controls the learning rate at which the centers c_y are updated, which is essential to the performance of the model. Similarly, the training becomes unstable for κ greater than 0.005. Hence, we decided to randomly select from the values 0.005, 0.001, 0.0005, and 0.0001. Table 2 summarises the values considered.

Hyperparameter	Options
λ	1, 0.5, 0.1, 0.05, 0.01, 0.005, 0.001
Temperature, T	100 to 495 in step of 5
κ	0.005, 0.001, 0.0005, 0.0001

Table 2: Hyperparameter search space for λ, T and κ.

According to [7], CER is the best-performing loss function, followed by CCE. Since FLR is proposed as a better alternative to CER, we will compare SoftNN and Center loss with the custom loss function FLR and the commonly used loss function CCE.

6 EXPERIMENTAL RESULTS

In this section, we present the results based on the framework shown in Algorithm 1 for ASCADf, ASCADr, and CHES_CTF. To evaluate the effectiveness of the deep feature loss function in DLSCA, we employ CCE and FLR as the baseline loss functions on top of deep feature loss functions. We train the DNN with Eq. (2) for SoftNN and Eq. (4) for Center loss. Subsequently, we showcase the effectiveness of the deep feature loss function on desynchronized traces in Section 6.1.

ASCADf. The results on the ASCADf dataset for both $NTGE_{best}$ and $NTGE_{median}$ can be found in Table 3. For the MLP (HW) setting, although training with FLR only attains the most favorable results, when using SoftNN and Center loss, we can still obtain $NTGE$ that is very close to their baseline counterpart. In fact, CCE+Center loss obtains better $NTGE_{best}$ than just CCE. As for CNN (HW) and CNN (ID), we reach similar observations.

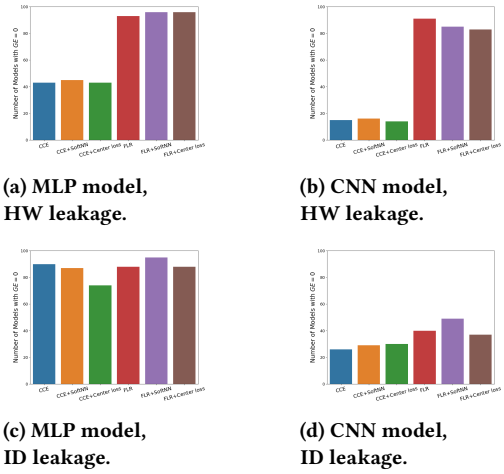
In fact, FLR+Center loss obtains a better $NTGE_{best}$ than FLR itself in CNN (HW). Moreover, CCE+SoftNN has a smaller $NTGE_{median}$ than CCE. CCE+SoftNN and CCE+Center loss have a better $NTGE_{best}$. For CNN (ID), CCE+Center loss has a smaller $NTGE_{best}$ than CCE. Furthermore, FLR+SoftNN and FLR+Center loss have a smaller $NTGE_{best}$ and $NTGE_{median}$ than only FLR. Among all other scenarios, the best result is obtained for the MLP (ID) setting when using FLR+Center loss. It obtains NT_{best} of 236 traces and NT_{median} of 344 traces. Moreover, we highlight that FLR+SoftNN attains the second best $NTGE_{median}$ of 386.5 traces and the third best $NTGE_{best}$ of 248 traces. These results illustrate that both SoftNN and Center loss are effective for the SCA domain.

	CCE	CCE+SoftNN	CCE+Center loss	FLR	FLR+SoftNN	FLR+Center loss
MLP (HW)	1461/1783.5	1531/1963	1136/1802	411/603	505/635	488/789.5
CNN (HW)	1787/> 2000	1614/1970.5	1610/> 2000	556/772	570/831.5	531/790
MLP (ID)	245/474	294/> 2000	349/544.5	328/612	248/386.5	236/344
CNN (ID)	457/962.5	633/994	403/> 2000	470/> 2000	377/716	297/727.5

Table 3: $NTGE$ on ASCADf (best/median).

Out of 100 different model architectures, we are also interested in the number of models that obtain $GE = 0$ for the different scenarios. Figure 4 shows the number of models that obtain $GE = 0$ for all the scenarios. Notice that for the setting with MLP (HW) and CNN

(HW), loss functions that use FLR as a baseline function obtain more models with $GE = 0$ compared to those using CCE. In the CNN (HW) setting, using FLR to train the DNNs allows us to acquire the most models with $GE = 0$. Despite that, training with FLR+SoftNN acquires the most number of models with $GE = 0$ for MLP (HW), MLP (ID), and CNN (ID).

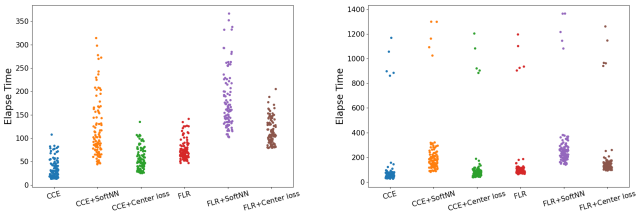


(a) MLP model, HW leakage. **(b) CNN model, HW leakage.**

(c) MLP model, ID leakage. **(d) CNN model, ID leakage.**

Figure 4: Number of trained models with $GE = 0$ out of 100 different model architectures for ASCADf.

Next, we are interested in the training time for the different lost functions. The training time for the 100 models can be seen in Figure 5. As expected, the training time when applying SoftNN and Center loss will increase since it requires additional computation. We note that SoftNN increases the training time more than the Center loss regardless of the base loss function. This is because SoftNN requires calculating per Eq. (1) for all layers, while Center loss only calculates Eq. (3) for the last layer.



(a) MLP model, ID leakage. **(b) CNN model, ID leakage.**

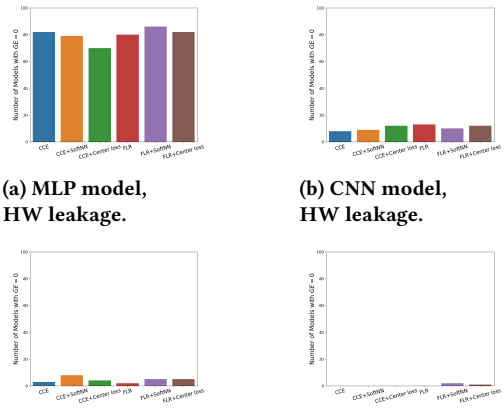
Figure 5: Training time of 100 random models per loss function on the ASCADf dataset.

ASCADr. Here, we present the experimental outcomes for ASCADr. The $NTGE_{best}$ and $NTGE_{median}$ results for ASCADr can be found in Table 4. Notice we reach similar observations as for ASCADf. Sometimes, the baseline loss functions perform better than training with the SoftNN or Center loss, but in those instances, the results of using deep features loss functions are very close. For example, in MLP (HW), the $NTGE_{best}$ and $NTGE_{median}$ for CCE+SoftNN and CCE+Center loss are less than 2000 traces. We also see instances where both SoftNN and Center loss obtain better results. For example, in MLP (HW), FLR+SoftNN and FLR+Center loss reach superior $NTGE_{median}$ than just training with FLR. Another instance is MLP (ID), where FLR+SoftNN and FLR+Center loss reach better $NTGE_{best}$ and $NTGE_{median}$ than just training with FLR. Among all the different settings, FLR+SoftNN reaches both the best $NTGE_{best}$ using 1317 traces and the best $NTGE_{median}$ with 2592 traces in CNN (HW) setting.

	CCE	CCE+SoftNN	CCE+Center loss	FLR	FLR+SoftNN	FLR+Center loss
MLP (HW)	3516/6377.5	3878/6877.5	4155/7523	2033/9154.5	3559/5463.5	3125/4456
CNN (HW)	3054/4642.5	5912/8016	3381/5408.5	2908/6812.5	1317/2592	1568/3867
MLP (ID)	8384/> 10000	7217/> 10000	8661/> 10000	8242/> 10000	4490/7574.5	9446/> 10000
CNN (ID)	$GE = 4$	$GE = 1$	$GE = 1$	$GE = 1$	9964/> 10000	8119/9681

Table 4: $NTGE$ on ASCADr (best/median)

Figure 6 showcases the number of trained models with $GE = 0$ for all scenarios for ASCADr. For MLP (HW), FLR+SoftNN obtains the most models with $GE = 0$. This is followed by FLR+Center loss and CCE, both having the same number of models with $GE = 0$. As for CNN (HW), FLR obtains the most models with $GE = 0$. This is followed by CCE+Center loss and FLR+Center loss with having the same number of models with $GE = 0$. For MLP (ID), CCE+SoftNN obtains the most trained models with $GE = 0$ compared to other loss functions. This is followed by FLR+SofNN, and then CCE+Center loss. In the case of CNN (ID), only FLR+SoftNN and FLR+Center loss obtain $GE = 0$ with less than 10000 traces.



(a) MLP model, HW leakage. **(b) CNN model, HW leakage.**

(c) MLP model, ID leakage. **(d) CNN model, ID leakage.**

Figure 6: Number of trained models with $GE = 0$ out of 100 different model architectures for ASCADr.

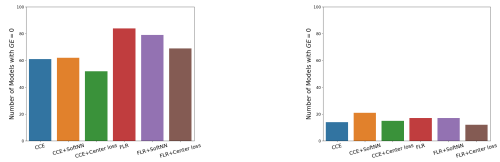
CHES_CTF. Table 5 illustrates the $NTGE_{best}$ and $NTGE_{median}$ for CHES_CTF. We observe that FLR obtains the best $NTGE_{best}$ and $NTGE_{median}$ among all the other scenarios. Despite that, using SoftNN and Center loss still obtain well-performing models that are close to their baseline loss function. For MLP (HW) and MLP (ID),

CCE+SoftNN and CCE+Center loss obtained a better $NTGE_{median}$ compared to CCE. On the other hand, CCE+SoftNN achieves a better $NTGE_{best}$ compared to CCE in the MLP (HW) setting while CCE+Center loss achieves a better $NTGE_{best}$ compared to CCE in CNN (HW). Furthermore, for CNN (HW), FLR+Center loss obtains the best $NTGE_{best}$ and $NTGE_{median}$. For the scenario concerning the ID leakage model, none of the loss functions managed to acquire a model with $GE = 0$. However, for the ID leakage model using CCE+Center loss, we acquire a model with $GE = 1$ when CCE and FLR only found models with $GE \geq 2$. Moreover, FLR+SoftNN also obtains a model with $GE = 1$ for the MLP (ID) scenario.

	CCE	CCE+SoftNN	CCE+Center loss	FLR	FLR+SoftNN	FLR+Center loss
MLP (HW)	1071/2459.5	1019/1902	1351/1824.5	549/956.5	713/1676	1129/1542.5
CNN (HW)	1383/2689	1502/2325	1192/1680	818/1596.5	1118/1724.5	656/1013
MLP (ID)	$GE = 2$	$GE = 2$	$GE = 1$	$GE = 3$	$GE = 1$	$GE = 1$
CNN (ID)	$GE = 7$	$GE = 2$	$GE = 1$	$GE = 7$	$GE = 19$	$GE = 3$

Table 5: $NTGE$ on CHES_CTF (best/median)

The number of models with $GE = 0$ for the HW leakage model is illustrated in Figure 7. We observe that FLR acquires the most models with $GE = 0$ for the MLP (HW) setting. On the other hand, CCE+SoftNN obtains the most models for the CNN (HW) setting instead.



(a) MLP model, HW leakage. **(b) CNN model, HW leakage.**

Figure 7: Number of trained models with $GE = 0$ out of 100 different model architectures for CHES_CTF.

6.1 Results for Desynchronized Traces

In this section, we explore how applicable are deep feature loss functions on datasets protected additionally with desynchronization. We consider two desynchronization levels, 50 and 100, on both ASCADf and ASCADr. We denote the ASCADf_desync50 and ASCADf_desync100 for ASCADf with desynchronization 50 and 100 respectively. Similarly, ASCADr_desync50 and ASCADr_desync100 are datasets with desynchronization of 50 and 100 on ASCADr, respectively. For ASCADf_desync50 and ASCADf_desync100, we use 10000 attack traces. On the other hand, for ASCADr_desync50 and ASCADr_desync100, we use 20000 attack traces.

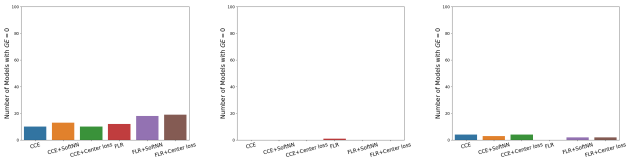
ASCADf_desync50. The $NTGE_{best}$ and $NTGE_{median}$ are depicted in Table 6 for ASCADf_desync50. Notice that introducing desynchronization to the traces increases the difficulty in finding a model with $GE = 0$.

With CNN (ID), using SoftNN and Center loss results in better $NTGE_{best}$ compared to their baseline counterparts. Furthermore, both CCE+SoftNN and FLR+SoftNN have $NTGE_{median}$ that is less than 10000 traces. Next, CNN trained with FLR obtained the best $NTGE_{best}$ for CNN (HW) among the other scenarios. In terms of $NTGE_{median}$, FLR+SoftNN is the best with 1684.5 traces, followed by FLR+Center loss with 2020.5.

	CCE	CCE+SoftNN	CCE+Center loss	FLR	FLR+SoftNN	FLR+Center loss
MLP (HW)	$GE = 210$	$GE = 223$	$GE = 202$	$GE = 15$	$GE = 2$	$GE = 5$
CNN (HW)	3208/4106.5	2987/4325	3306/4316.5	710/5061.5	1450/1684.5	1440/2020.5
MLP (ID)	$GE = 40$	$GE = 23$	$GE = 27$	9997/> 10000	$GE = 14$	$GE = 13$
CNN (ID)	7508/> 10000	7102/9137.5	7114/> 10000	$GE = 1$	6103/6798	9844/> 10000

Table 6: $NTGE$ on ASCADf_desync50 (best/median)

With respect to the number of models with $GE = 0$, FLR+Center loss obtains the most models in CNN (HW). Within the same setting of CNN (HW), FLR+SoftNN followed next by achieving the second highest number of models with $GE = 0$. The only MLP that reaches $GE = 0$ under the HW leakage model is trained using FLR. Surprisingly, none of the 100 CNN trained using FLR under the ID leakage model obtained $GE = 0$ with less than 10000 traces. Instead, CCE and CCE+Center loss obtain the most models in this setting. Nevertheless, from Table 6, FLR+SoftNN obtained the best $NTGE_{best}$ and best median $NTGE_{median}$ in the CNN (ID) setting.



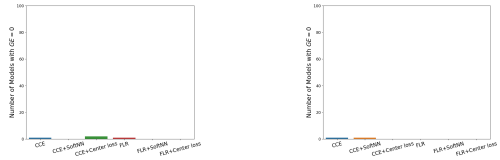
(a) CNN model, HW leakage. **(b) MLP model, ID leakage.** **(c) CNN model, ID leakage.**

Figure 8: Number of trained models with $GE = 0$ out of 100 different model architectures for ASCADf_desync50.

ASCADr_desync50. The $NTGE_{best}$ and $NTGE_{median}$ results are illustrated in Table 7. Although FLR attains better performing models compared to FLR+SoftNN and FLR+Center loss in MLP (HW), CNN (HW), and CNN (ID), both FLR+SoftNN and FLR+Center loss could still obtain a smaller GE compared to FLR in MLP (ID). Besides, the best $NTGE_{best}$ is obtained by training CNN with the HW leakage model using CCE+Center loss. We remark that CCE+SoftNN obtains a better $NTGE_{best}$ than CCE in the MLP (ID) setting. We only obtain a total of 8 models (see Figure 9), and CCE+SoftNN can find two models with $GE = 0$ whereas the rest of the loss functions only found one model.

	CCE	CCE+SoftNN	CCE+Center loss	FLR	FLR+SoftNN	FLR+Center loss
MLP (HW)	$GE = 36$	$GE = 35$	$GE = 35$	$GE = 21$	$GE = 60$	$GE = 59$
CNN (HW)	19000/> 20000	$GE = 2$	17458/> 20000	19669/> 20000	$GE = 66$	$GE = 16$
MLP (ID)	19906/> 20000	19497/> 20000	$GE = 3$	$Ge = 9$	$GE = 6$	$GE = 2$
CNN (ID)	$GE = 4$	$GE = 3$	$GE = 1$	$GE = 1$	$GE = 2$	$GE = 10$

Table 7: $NTGE$ on ASCADr_desync50 (best/median)



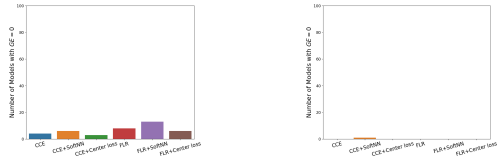
(a) CNN model, HW leakage. **(b) MLP model, ID leakage.**

Figure 9: Number of trained models with $GE = 0$ out of 100 different model architectures for ASCADr_desync50.

ASCADf_desync100. Next, we showcase the results in Table 8 for ASCADf_desync100. Among all the settings, the best $NTGE_{best}$ is obtained by training with FLR in CNN (HW), while the best $NTGE_{median}$ is obtained by using FLR+Center loss in CNN (HW). Among the MLP with the HW leakage model, all the tested loss functions obtain results above $GE > 10$, except the FLR+SoftNN and FLR+Center loss. Furthermore, when training a CNN in the ID leakage model, we attain a $GE = 0$ with 9895 traces when using CCE+SoftNN, while other loss functions obtain $GE \geq 7$. In terms of the total number of models with $GE = 0$ attained, FLR+SoftNN obtains the most models for CNN (HW) (see Figure 10).

	CCE	CCE+SoftNN	CCE+Center loss	FLR	FLR+SoftNN	FLR+Center loss
MLP (HW)	$GE = 204$	$GE = 222$	$GE = 224$	$GE = 14$	$GE = 7$	$GE = 9$
CNN (HW)	5590/8652	7756/9081	5930/7775.5	2590/5277	3756/4861.5	3051/4405.5
MLP (ID)	$GE = 3$	$GE = 18$	$GE = 16$	$GE = 16$	$GE = 19$	$GE = 14$
CNN (ID)	$GE = 7$	9895/> 10000	$GE = 7$	$GE = 15$	$GE = 12$	$GE = 8$

Table 8: *NTGE* on ASCADf_desync100 (best/median)



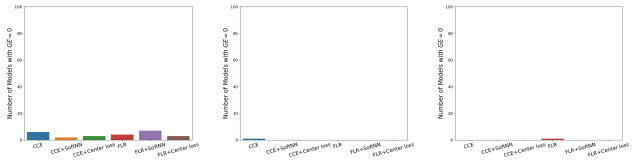
(a) CNN model, HW leakage. **(b) CNN model, ID leakage.**

Figure 10: Number of trained models with $GE = 0$ out of 100 different model architectures for ASCADf_desync100.

ASCADr_desync100. The results are shown in Table 9 for ASCADr_desync100. Although training CCE and FLR solely could obtain better results in scenarios like MLP (ID) and CNN (ID), and the best $NTGE_{best}$ is obtained by using FLR in CNN (HW), SoftNN and Center loss could improve the performance in some cases. For instance, the best $NTGE_{median}$ among all the scenarios is reached with FLR+Center loss for CNN (HW). Furthermore, in the same setting of CNN (HW), CCE+SoftNN and CCE+SoftNN reach a better $NTGE_{best}$ compared to CCE, their baseline counterpart. Moreover, FLR+SoftNN obtains the most models with $GE = 0$ for the CNN model with the HW leakage model (see Figure 11).

	CCE	CCE+SoftNN	CCE+Center loss	FLR	FLR+SoftNN	FLR+Center loss
MLP (HW)	$GE = 12$	$GE = 15$	$GE = 18$	$GE = 12$	$GE = 5$	$GE = 9$
CNN (HW)	19220/> 20000	18105/> 20000	18615/> 20000	9788/16344	11724/> 20000	10864/16209
MLP (ID)	19525/> 20000	$GE = 7$	$GE = 19$	$GE = 6$	$GE = 1$	$GE = 4$
CNN (ID)	$GE = 1$	$GE = 1$	$GE = 1$	19992/> 20000	$GE = 1$	$GE = 2$

Table 9: *NTGE* on ASCADr_desync100 (best/median)



(a) CNN model, HW leakage. **(b) MLP model, ID leakage.** **(c) CNN model, ID leakage.**

Figure 11: Number of trained models with $GE = 0$ out of 100 different model architectures for ASCADr_desync100.

Overall, one observation is that when desynchronization has been added, experiments with the HW leakage model reach better results compared to those using the ID leakage model. This suggests that when desynchronization is added, due to the larger number of classes, the DNN struggles to perform better. This means that choosing the right leakage model is crucial in performing the attack.

Which Loss Function is Better? First, between the two baseline loss functions, FLR obtains better-performing models compared to CCE. This confirms the observation made in [6]. Still, the addition of deep feature loss functions could potentially improve the attack. Regarding which deep feature loss function to use, SoftNN and Center loss perform well in different scenarios. Among the synchronized datasets, it can be seen that in 6 out of 12 scenarios, SoftNN acquires the most number of models with $GE = 0$. Furthermore, the performance of SoftNN is constantly in the top few for synchronized datasets. Therefore, between SoftNN and Center loss, we recommend using SoftNN if the dataset is synchronized. However, we do note that SoftNN requires the most time to run compared to the other loss functions. Therefore, if time is an issue, Center loss would be recommended. On the other hand, Center loss reaches better performance compared to SoftNN. In three out of four datasets protected by desynchronization, Center loss reaches either the least traces to break the target or the best $NTGE_{median}$ to recover the key. As a result, if the dataset is protected by desynchronization, Center loss would be preferred compared to SoftNN.

7 CONCLUSIONS AND FUTURE WORK

We have presented two well-studied deep feature loss functions: SoftNN and Center loss. We evaluated the effectiveness of these deep feature loss functions with the CCE and FLR as the baseline loss functions in the SCA domain. Our results show that incorporating either SoftNN or Center loss could obtain better performances than just using CCE or FLR as the baseline function. This suggests that deep feature loss functions could provide significant model performance improvements. Therefore, we recommend to consider

the deep feature loss function in hyperparameter tuning. We further note that SoftNN and Center loss are loss functions created for general purposes, while FLR is developed for the SCA domain exclusively. Yet, SoftNN and Center loss could attain better performance. Hence, one direction to look into is to develop a deep feature loss function suited for SCA through theoretical insights like mutual information. Furthermore, imbalanced data are common and realistic in SCA, especially when considering the Hamming weight or Hamming distance leakage models. Therefore, one could explore a deep feature loss function that tackles the imbalance problem within SCA. We hope this work serves as a starting point to consider the exploration of deep feature loss functions within the realm of SCA, as well as including various loss functions in the hyperparameter tuning for DLSCA.

8 ACKNOWLEDGEMENT

This work received funding in the framework of the NWA Cybersecurity Call with project name PROACT with project number NWA.1215.18.014, which is (partly) financed by the Netherlands Organisation for Scientific Research (NWO). Additionally, this work was supported in part by the Netherlands Organization for Scientific Research NWO project DISTANT (CS.019).

REFERENCES

[1] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. 2020. Deep learning for side-channel analysis and introduction to ASCAD database. *J. Cryptogr. Eng.* 10, 2 (2020), 163–188. https://doi.org/10.1007/s13389-019-00220-8

[2] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. 2017. Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures. In *Cryptographic Hardware and Embedded Systems – CHES 2017*, Wieland Fischer and Naofumi Homma (Eds.). Springer International Publishing, Cham, 45–68.

[3] Morris Dworkin, Elaine Barker, James Nechvatal, James Foti, Lawrence Bassham, E. Roback, and James Dray. 2001. Advanced Encryption Standard (AES). https://doi.org/10.6028/NIST.FIPS.197

[4] Nicholas Frosst, Nicolas Papernot, and Geoffrey Hinton. 2019. Analyzing and Improving Representations with the Soft Nearest Neighbor Loss. arXiv:1902.01889 [stat.ML]

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[6] Maikel Kerkhof, Lichao Wu, Guilherme Perin, and Stjepan Picek. 2022. Focus is Key to Success: A Focal Loss Function for Deep Learning-Based Side-Channel Analysis. In *Constructive Side-Channel Analysis and Secure Design*, Josep Balasch and Colin O'Flynn (Eds.). Springer International Publishing, Cham, 29–48.

[7] Maikel Kerkhof, Lichao Wu, Guilherme Perin, and Stjepan Picek. 2023. No (good) loss no gain: systematic evaluation of loss functions in deep learning-based side-channel analysis. *Journal of Cryptographic Engineering* 13, 3 (May 2023), 311–324. https://doi.org/10.1007/s13389-023-00320-6

[8] Kussul, Nataliia and Lavreniuk, Mykola and Skakun, Sergii and Shelestov, Andrii. 2017. Deep Learning Classification of Land Cover and Crop Types Using Remote Sensing Data. *IEEE Geoscience and Remote Sensing Letters* 14, 5 (2017), 778–782. https://doi.org/10.1109/LGRS.2017.2681128

[9] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*. 2980–2988.

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

[11] Colin O'Flynn and Zhizhang David Chen. 2014. ChipWhisperer: An Open-Source Platform for Hardware Embedded Security Research. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*.

[12] Guilherme Perin, Lichao Wu, and Stjepan Picek. 2022. Exploring Feature Selection Scenarios for Deep Learning-based Side-channel Analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2022, 4 (Aug. 2022), 828–861. https://doi.org/10.46586/tches.v2022.i4.828-861

[13] Guilherme Perin, Lichao Wu, and Stjepan Picek. 2022. I Know What Your Layers Did: Layer-wise Explainability of Deep Learning Side-channel Analysis. Cryptology ePrint Archive, Paper 2022/1087. https://eprint.iacr.org/2022/1087 https://eprint.iacr.org/2022/1087.

[14] Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni. 2018. The Curse of Class Imbalance and Conflicting Metrics with Machine Learning for Side-channel Evaluations, volume=2019. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 1 (Nov. 2018), 209–237. https://doi.org/10.13154/tches.v2019.i1.209-237

[15] Stjepan Picek, Guilherme Perin, Luca Mariot, Lichao Wu, and Lejla Batina. 2023. SoK: Deep Learning-Based Physical Side-Channel Analysis. *ACM Comput. Surv.* 55, 11, Article 227 (feb 2023), 35 pages. https://doi.org/10.1145/3569577

[16] Jorai Rijsdijk, Lichao Wu, Guilherme Perin, and Stjepan Picek. 2021. Reinforcement Learning for Hyperparameter Tuning in Deep Learning-based Side-channel Analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2021, 3 (Jul. 2021), 677–707. https://doi.org/10.46586/tches.v2021.i3.677-707

[17] Ruslan Salakhutdinov and Geoff Hinton. 2007. Learning a Nonlinear Embedding by Preserving Class Neighbourhood Structure. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 2)*, Marina Meila and Xiaotong Shen (Eds.). PMLR, San Juan, Puerto Rico, 412–419. https://proceedings.mlr.press/v2/salakhutdinov07a.html

[18] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. 2016. A Discriminative Feature Learning Approach for Deep Face Recognition. In *Computer Vision – ECCV 2016*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (Eds.). Springer International Publishing, Cham, 499–515.

[19] Lichao Wu, Guilherme Perin, and Stjepan Picek. 2022. I Choose You: Automated Hyperparameter Tuning for Deep Learning-based Side-channel Analysis. *IEEE Transactions on Emerging Topics in Computing* (2022), 1–12. https://doi.org/10.1109/TETC.2022.3218372

[20] Lichao Wu, Guilherme Perin, and Stjepan Picek. 2022. The Best of Two Worlds: Deep Learning-assisted Template Attack. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2022, 3 (Jun. 2022), 413–437. https://doi.org/10.46586/tches.v2022.i3.413-437

[21] Baoguo Yuan, Junfeng Wang, Dong Liu, Wen Guo, Peng Wu, and Xuhua Bao. 2020. Byte-level malware classification based on markov images and deep learning. *Computers & Security* 92 (2020), 101740. https://doi.org/10.1016/j.cose.2020.101740

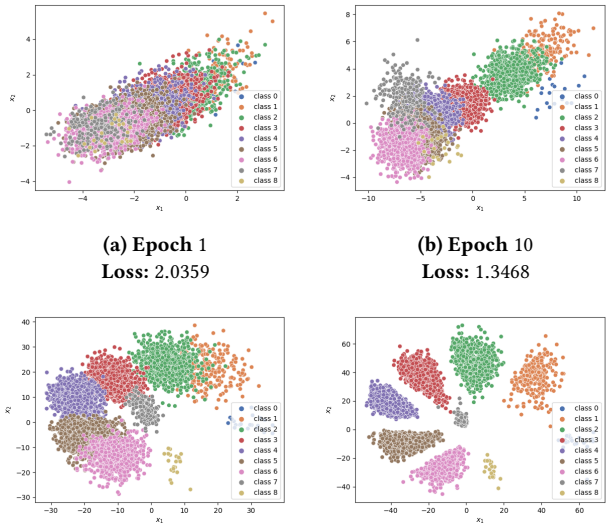
[22] Gabriel Zaid, Lilian Bossuet, François Dassance, Amaury Habrard, and Alexandre Venelli. 2020. Ranking Loss: Maximizing the Success Rate in Deep Learning Side-Channel Analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2021, 1 (Dec. 2020), 25–55. https://doi.org/10.46586/tches.v2021.i1.25-55

[23] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. 2019. Methodology for Efficient CNN Architectures in Profiling Attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2020, 1 (Nov. 2019), 1–36. https://doi.org/10.13154/tches.v2020.i1.1-36

[24] Jiajia Zhang, Mengce Zheng, Jiehui Nan, Honggang Hu, and Nenghai Yu. 2020. 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* 2020, 3 (Jun. 2020), 73–96. https://doi.org/10.13154/tches.v2020.i3.73-96

A VISUALIZATION WITH BASELINE FUNCTIONS ONLY

We use the same MLP architecture as used in Section 4 that was trained with Center loss. Here, we train MLP with only CCE without any deep feature loss function. Figure 12a shows the visualization of the deep features.



(a) Epoch 1
Loss: 2.0359

(b) Epoch 10
Loss: 1.3468

(c) Epoch 30
Loss: 0.3846

(d) Epoch 50
Loss: 0.0784

Figure 12: Visualization of the separation when using CCE only.