Hacking Lab Reproducibility project: A Novel Evaluation Metric for Deep Learning-Based Side Channel Analysis and Its Extended Application to Imbalanced Data

Delft University of Technology
Wolfgang Bubberman 4704673 w.l.bubberman@student.tudelft.nl
Sengim Karayalcin 4598539 s.karayalcin@student.tudelft.nl
Jari Bervoets 4594363 j.m.bervoets@student.tudelft.nl

April 9, 2021

Abstract

The field of side channel analysis (SCA) has increasingly been using Deep Learning to enhance the attacks. However, attack evaluation metrics, like Guessing Entropy (GE) and Success Rate (SR), are computationally inefficient. Furthermore, traditional Deep Learning metrics, like accuracy and precision, are not suitable for evaluating Deep Learning models in the context of SCA, as classes are often imbalanced. However, recently Zhang et al. have proposed a new evaluation metric for SCA, Cross Entropy Ratio (CER), that provides a good indication of the success of the attack and is viable to embed in Deep Learning Algorithms. Additionally this metric can be used as a loss function to train better models when training data is imbalanced. Throughout this report, a reproduction of the results of the paper introducing CER will be showcased, and a self-developed metric, the Log-Likelihood Ratio (LLR), will be introduced as well. These two metrics were compared to Cross Entropy (CROSS) and each other as loss functions, using several neural network architectures and data-sets. The final results of this report will showcase that CER, as a loss function, in the context of SCA, when classes are imbalanced, is better than using regular Cross Entropy. LLR performs slightly worse than CER in almost every scenario, but is generally better than Cross Entropy. Therefore, this report shows that the results from Zhang et al. are reproducible and the CER metric can be used as a evaluation metric to more accurately evaluate Deep Learning models.

1 Introduction

Side channel analysis (SCA) is a type of attack that exploits the weakness of physical implementations instead of software vulnerabilities, as was proposed by Kocher [10]. Recently, Deep Learning techniques have seen adaptation in the field of SCA and show comparable, and sometimes even better, performance than the more conventional methods. Deep Learning SCA might seem promising but it still remains the question why exactly Deep Learning SCA works so well in certain applications and situations [1]. Deep learning SCA

is effectively done by mapping it to a supervised classification problem, a problem that is well studied and known by now. A recent discovery by Masure et al. [14] indicates that the Negative Log Likelihood is related to a lower bound of Mutual Information between the sensitive intermediate value and the leakage. Therefore, the deep learning paradigm might be suitable for SCA from a worst-case point of view.

However, deep learning models are evaluated on metrics like accuracy and precision, whilst in the SCA world evaluation metrics like Guessing Entropy and Success Rate [24] are more common. These metrics differ quite a lot and on top of this, deep learning metrics might be misleading for SCA according to Cagli et al. [4]. Suppose that the attacker can acquire several traces for varying plaintexts, the accuracy metric is not sufficient alone to evaluate the attack performance. The accuracy metric only takes into consideration the label corresponding to the maximal score and does not consider the other labels, a metric like Success Rate would.

Deep learning is also prone to be influenced by class imbalance, as this could cause a bias towards a majority class of the training set, and therefore have sub-par performance. Whilst there are some proposed methodologies for imbalanced data, like using re-sampling techniques to re-balance the data [15, 18] or compensating for the minority class in the training algorithm [26], these techniques are not always optimal as they cannot use the full sets of training data.

Therefore, doing research into the right metrics for Deep Learning SCA is crucial. The issues traditional SCA metrics have with evaluation of attack performance and the issues that arise when class imbalance comes into play are the main driving forces behind the research of this report.

In this paper, the metric proposed by Zhang et al. [33] will be discussed and a new metric will be proposed. The metric of Zhang et al. is the Cross Entropy Ratio (CER), which evaluates the performance of deep learning models for SCA. As CER does not require the large amount of attacks that Guessing Entropy and Success Rate do, it can be calculated faster and can thus be used as a Deep Learning SCA metric. The CER metric can also be used as a loss function and shows improvements with imbalanced data compared to more traditional approaches. In this report another metric will be introduced, the Log-Likelihood Ratio (LLR), which approximates the ratio of the key likelihood compared to the likelihood of random other keys. LLR can also be used as a loss function, in the similar fashion as CER.

2 Background

Deep learning has become a fast growing research area in machine learning and pattern recognition society. It has gained success in the field of speech recognition, computer vision and language processing, and it is making strides in other fields now as well. A Deep Neural Network (DNN) is defined as an artificial neural network with multiple hidden layers of neurons between the input and output layers [21]. Deep networks are also more complex and computationally demanding than traditional networks, as the neurons they consist of can have indirect effects on each other.

2.1 Side channel analysis (SCA)

One type of SCA is profiled side channel attacks. These profiled side channel attacks consist of two steps: first the adversary creates a leakage model by analyzing the leakages of recre-

ation of the victim's device, and afterwards they exploit this leakage model to extract the key from the victim's device. These leakages, such as power consumption and electromagnetic emanation, are often obtainable with high quality and low cost. Some variants of these profiled side channel attacks are the stochastic attack [22] and the (pooled) template attack [5], there also exist non-profiled SCA attacks that only utilise an attacking phase, such as the Mutual Information Analysis [6], Correlation Power Analysis [3], and Differential Power Analysis [9]. However, what is quite interesting about these profiled side channel attacks is that they can be seen as a classification problem, where the adversary needs to classify intermediate values based on the leakage traces. Therefore we can perform these profiled side channel attacks with supervised learning.

2.2 Supervised learning

Supervised learning is the learning of a function through labelled training data. As the training data is labelled, there is a desired output for every input. Supervised learning first analyses the training data, and then tries to generate a function to predict labels for unlabelled data.

2.3 Multilayer Perceptron (MLP)

A Multilayer Perceptron (MLP) is a feed-forward artificial neural network that has at least three layers of neurons [17]. These three layers are the input layer, the hidden layer and the output layer. Each neural unit, except for the input layer's neurons, uses a nonlinear activation function such as *softmax* [23]. MLP uses a form of supervised learning called back-propagation for training [19]. The multiple layers and non-linear activation functions make it so that MLPs can recognize and classify data that is not linearly separable, unlike a linear perceptron. The way neural networks classify how close they are to the desired output is through loss functions. These loss functions significantly increase the ability of the neural network to predict things like paths and ranking [29], of which the latter is key to SCA.

2.4 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN) are a type of feed-forward DNN that exists out of multiple layers. Given this, there exist many different variants of CNN architectures [7]. However, the basic structure of each variant is the same. They each consist out of four types of layers: the convolutional, pooling, fully-connected and output layers [32]. The convolutional layer is made up out of several convolutional kernels, and aims to learn feature representations of the inputs, this is done by computing different feature maps in each of its kernels. The usage of activation functions also occurs in the convolutional layers, a common activation function here is the *tanh* [11] or *ReLU* [16] function. The pooling layer computes shift-invariances by reducing the resolution of the feature maps, it can be found most often between two convolutional layers, as all feature maps of the pooling layer are connected to their corresponding feature map of the connected convolutional layers. Two kind of pooling operations are used most often, the average pooling operation [27] and the max pooling operation [2]. By placing multiple convolutional and pooling layers behind each other, there is a gradual extraction of higher-level feature representations [12]. The fully-connected layers come after these multiple stacked convolutional and pooling layers [8]. The fully-connected layers aim to perform high-level reasoning by connecting all previous

neurons, and then generating global semantic information from that. Intuitively, last layer is always the output layer, for which commonly the *softmax* function gets used in SCA.

If the loss function of the CNN gets minimized, and therefore the difference between the current output and the desired output of the CNN, the best possible set of parameters for the CNN is found. Therefore training a CNN is an optimization problem.

2.5 Guessing Entropy (GE)

One of the ways to measure the vulnerability of encryption against a side channel analysis (SCA) is the Guessing Entropy (GE), originally proposed by Massey et al. [13]. GE measures the average number of key candidates to test after the side channel attack [24]. The higher the GE, the more wrong key guesses have to be checked before the correct key value is considered. Therefore, GE measures the average computation cost required for a successful side channel attack and is a good leakage evaluation metric [34].

Guessing **E**ntropy can be defined as:

$$GE(N_a) = \mathbf{E}(g_{S_a}(k^*))$$

where $\mathbf{E}$ is the mean function, N_a is the set of the attack traces, g_{S_a} is a vector of key guesses ordered by their predicted log-likelihood, for a random subset $S_a \subset N_a$, and $g_{S_a}(k^*)$ is the index of the correct key hypothesis k^*. Definition based on Massey et al. [13].

2.6 Success rate

Success rate (SR) is also a commonly used metric in SCA [24]. Success rate measures the probability that an attacker guesses the correct key within a certain number of leakage measurements [20]. An intuitive way of assessing the Success Rate is to perform the attack several times and estimate the Success Rate based on this. However, this might be too expensive, both time and computation wise. Therefore, suggestions have been made for approximations of the Success Rate, for example the one by Standeart et al. [25].

Success **R**ate can be defined as:

$$SR(N_a) = Pr[g_{S_a}(k^*) = 1]$$

where $Pr[x]$ is the probability of x, N_a is the set of the attack traces, g_{S_a} is a vector of key guesses ordered by their predicted log-likelihood, for a random subset $S_a \subset N_a$, and $g_{S_a}(k^*)$ is the index of the correct key hypothesis k^* [14]. Note that if $g_{S_a}(k^*) = 1$ the attack is successful because the correct key is the highest ranked key.

3 Metrics

The main problem with metrics like Success Rate and Guessing Entropy is that they are computationally hard to estimate as a large number of attacks need to be mounted to get an accurate estimation. On the other hand, the Deep Learning metrics, like accuracy and precision, do not give an accurate view of how successful an attack will be, especially when there is class imbalance [18]. Because of this, there is a need for metrics that can be efficiently computed and give an accurate indication of how successful an attack will be. Additionally, a metric that can be computed efficiently might have extended applications as a loss function for the Deep Learning models.

Several different metrics have been developed over the past few years. Negative Log-Likelihood (NLL) is a proposed loss function for which it has been shown that its minimization is asymptotically equivalent to minimizing Cross Entropy [14]. Another metric that is more focused on evaluation of a model is the Leakage Distribution Difference (LDD) [28].

In this report Cross Entropy ratio (CER) proposed by Zhang et al. [33] will be further studied. The results Zhang et al. obtained will be reproduced, and the effectiveness of CER as a loss function for additional state-of-the-art Deep Learning architectures will be evaluated. Additionally, we propose a new metric that combines the ideas behind NLL and CER. In this section both of these metrics will be described in more detail.

3.1 Cross-entropy ratio

Cross Entropy for a key $k \in \mathcal{K}$, where $\mathcal{K}$ is the space of possible keys, is defined as:

$$CE(k) = H(Pr[X, L^k], M_\theta[L^k]) \tag{1}$$

Here $Pr[X, L^k]$ is the joint distribution for $k \in \mathcal{K}$, $M_\theta[L^k]$ is the predicted distribution for $k \in \mathcal{K}$ and H is the Shannon-Entropy function. L^k denotes the labels generated with a certain key hypothesis $k \in \mathcal{K}$.

Now let $k^* \in \mathcal{K}$ be the correct key. Then CER is defined as

$$CER = \frac{CE(k^*)}{\mathbf{E}_{k \neq k^*}(CE(k))} \tag{2}$$

3.2 Log-Likelihood ratio

Log-Likelihood for a key candidate $k \in \mathcal{K}$ and a model M_θ is defined as:

$$LL(k) = \sum_{i=1}^{N} log(M_\theta(l_i^k)) \tag{3}$$

Where $L^k = \begin{pmatrix} l_1^k \\ l_2^k \\ \vdots \\ l_N^k \end{pmatrix}$ is the vector of labels generated for a key candidate $k \in \mathcal{K}$ with N samples and $M_\theta(l_i^k)$ is the predicted probability for label l_i^k.

Then the Log-Likelihood ratio is defined similarly to CER for a correct key k^*:

$$LLR = \frac{LL(k^*)}{\mathbf{E}_{k \neq k^*}(LL(k))} \tag{4}$$

3.3 Estimating LLR and CER

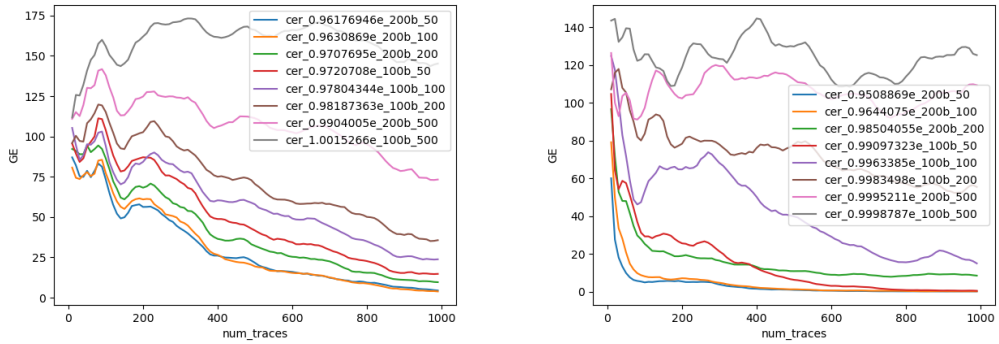
To use these functions as loss functions in a Deep Learning model, fast methods to estimate them are required. As described by Zhang et al. [33] the nominator for CER can be computed efficiently by calling the machine learning libraries' function to compute Cross Entropy. Additionally they show that the denominator can be computed by shuffling the correct labels and then computing the Cross Entropy as above.

To estimate LLR something similar can be done. The Log-Likelihood (LL) can be computed for the correct labels to compute the nominator. Subsequently, these labels can be shuffled, and then the LL of this new set of labels can be computed to estimate $\mathbf{E}_{k \neq k^*}(LL(k))$.

4 Experimental Evaluation of CER as metric

In this section some of the experiments to evaluate CER as a metric for Deep Learning SCA are reproduced. These experiments were conducted in the same way as they were by Zhang et al. [33]. Using the MLP from Benadjila et al. [1], several models were trained with varying batch size and epochs. Then on a separate attacking set, the GE and CER were computed. These experiments were conducted for both the Identity (ID) leakage model, and the Hamming Weight (HW) leakage model. It should be noted that for these models categorical Cross Entropy was used as the loss function.

As can be seen in Figure 1, the models with better Guessing Entropy generally also have lower CER scores. This leads to the conclusion that CER can be a good metric to evaluate the effectiveness of a model.



(a) Results using HW leakage model

(b) Results using ID leakage model

Figure 1: Results on the ASCAD data-set with Cross Entropy Ratio (CER) and varying hyperparameters

5 Experimental Evaluation of CER and LLR as loss functions

In this section several experiments to evaluate the effectiveness of CER and LLR as loss functions were conducted. First, the speed of the execution of all functions was tested. Secondly, to test the effectiveness of the loss functions, different types of models with several different hyperparameter setups were trained, using categorical Cross Entropy as a loss function as a control. Experiments have been run on several public data sets. For the

synchronized cases the AES_HD[1], DPA_v4_2[2] and the synchronized cases of ASCAD with a fixed key[3] are used. For these synchronized cases the models that were used are the best MLP as found by Benadjila et al. [1], and the CNN models that were tuned for the specific data-sets by Zaid et al. [31]. Additionally some models were ran with an Early stopping callback[4], monitoring valuation loss, with a patience of 20 epochs and restoring the best model. For the desynchronized cases the AES_RD[5] data-set and the desynchronized cases of the fixed key ASCAD[6] set were used. The models used were the specifically tuned models of Zaid et al. [31]. To run the experiments scripts based on the work by Zaid et al. were used.

5.1 Speed of the loss functions

The speed of the varying loss functions used throughout the paper has been measured and compared. This gave the results that can be seen in Table 1.

Every loss function was ran a hundred times to get a value for the time, and this was then done in turn a thousand times to get a good estimation on the average run time. What we can see from the results is that Cross Entropy Ratio (CER) and Log-Likelihood Ratio (LLR) are roughly 3 times as slow as the Cross Entropy (CROSS) function, which makes sense as the computation of LLR and CER is more complex.

Loss function	time (s)	std (s)
CROSS	0.87	0.038
CER	2.28	0.072
LLR	2.33	0.090

Table 1: Run times for loss functions

5.2 Experiments on the ASCAD data-set

The ASCAD database provides traces of a first order masked implementation of AES on a ATMEGA8515. Here, the masks are assumed to be known to simplify the attacks. The tests were ran on the fixed key portion of the database. Within this data-set several different options are available. As desynchronization can be added to the data-set this section is split into two subsections which discuss the results on the synchronized data-set, and the results on the desynchronized data-set.

For the training phase 45 000 traces were used as the training set, and 5 000 for the validation set. The remaining 10 000 traces were used as the attacking set to compute the Guessing Entropy.

5.2.1 Experiments on the synchronized ASCAD data-set

For the synchronized database two different models were used. The best MLP model from Benadjila et al. [1], and the CNN model that was tuned for this data-set by Zaid et al. [31].

[1] https://github.com/AESHD/AES_HD_data-set
[2] http://www.dpacontest.org/v4/42_traces.php
[3] https://github.com/ANSSI-FR/ASCAD/tree/master/ATMEGA_AES_v1
[4] https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping
[5] https://github.com/ikizhvatov/randomdelays-traces
[6] https://github.com/ANSSI-FR/ASCAD/tree/master/ATMEGA_AES_v1

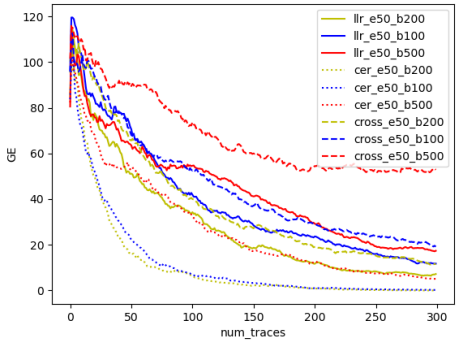
The hyperparameter configurations that were tested are all possible combinations of batch sizes 100, 200 and 500. For the CNNs a learning rate of 1e-3 and 50 and 100 epochs ere used. For the MLPs a learning rate of 1e-5 and 100 and 200 epochs were used.

In Figure 2 it can be seen that for both architectures the models trained with CER as the loss function are generally best, with the models using Cross Entropy being worst. The CNNs trained with LLR as a loss function are about as good as the ones trained with Cross Entropy and the MLPs using LLR are only slightly worse than the MLPs using CER. In Figure 3 the results using early stopping are displayed. For the MLPs there is not that significant a difference with the models that were not stopped early, except that some of the models were stopped very early and display performance that is a lot worse because of it. For the CNNs it can be seen that the performance of the LLR models is slightly better than when no early stopping is employed. Additionally, it can be seen that for the CNNs the models using CER and LLR as loss functions stop a lot earlier than the models using Cross Entropy.

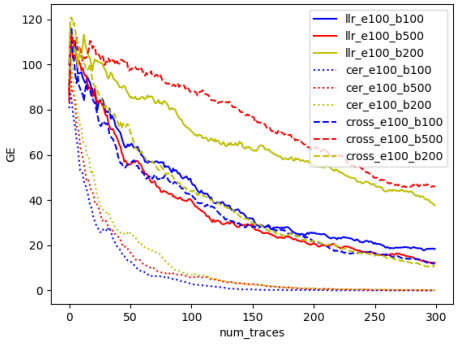
5.2.2 Experiments on the desynchronized ASCAD data-set

For the desynchronized cases, two different amounts of desynchronization (50 and 100) tested. In this paper, only the CNN model for each specific desynchronization as described by Zaid et al. [31] was used. This was because the CNN best models from Benadjila et al. [1] were very computationally intensive to train. Additionally, for these models Zhang et al. [33] already tested the CER and categorical cross-entropy loss functions. For the desync 50 case a learning rate of 1e-3 and a batch size of 256 was used, and the models were trained for 50 epochs. For the desync 100 case a learning rate of 1e-2 and a batch size of 256 was used, and the models were trained for 50 epochs. These hyperparameters were chosen as these are the ones used by Zaid et al. [31]. Only one set of hyperparameters was used for these models as they are computationally a lot more expensive to train than the models for the synced cases.

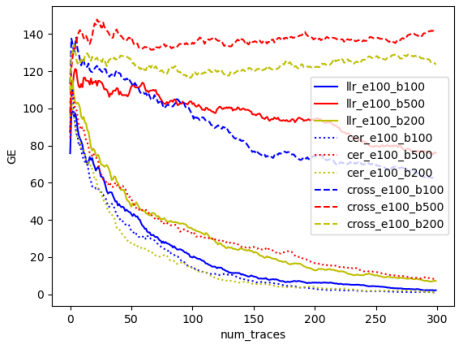
In Figure 4 it can be seen that for both amounts of desynchronization the model using CER performs the best. Additionally, it can be seen that for when the desynchronization is set to 100, the model using LLR as the loss function does not perform better than random guessing.



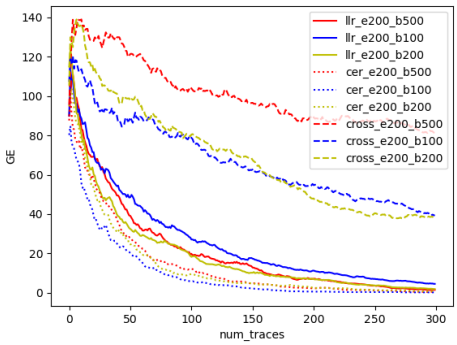
(a) 50 epochs CNN architecture



(b) 100 epochs CNN architecture

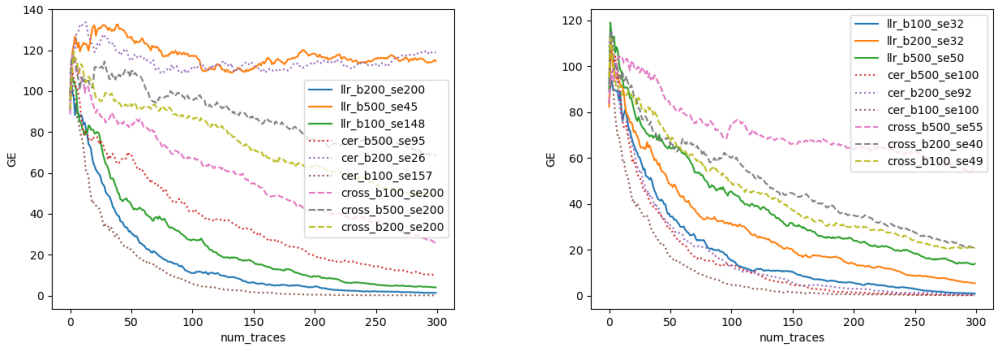


(c) 100 epochs MLP architecture



(d) 200 epochs MLP architecture

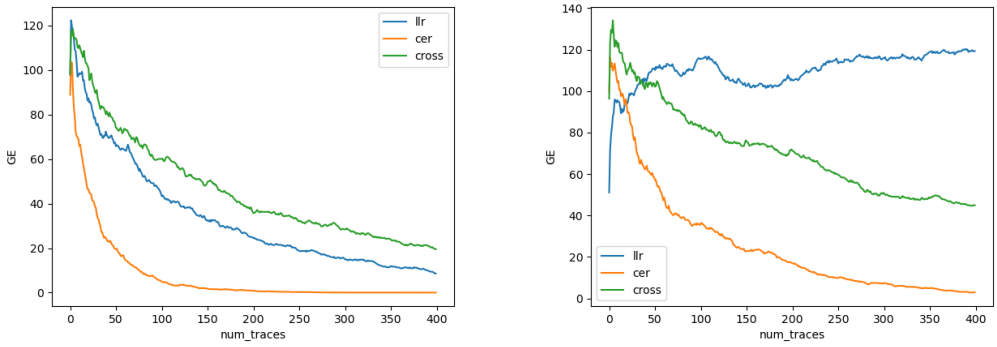
Figure 2: Results on the ASCAD data-set with Cross Entropy (CROSS), Cross Entropy Ratio (CER) and Log-Likelihood Ratio (LLR) as loss function



(a) Stopped early MLP architecture

(b) Stopped early CNN architecture

Figure 3: Results on the ASCAD data-set with Cross Entropy(CROSS), Cross Entropy Ratio (CER) and Log-Likelihood Ratio (LLR) as loss function, stopped early



(a) 50 desynced

(b) 100 desynced

Figure 4: Results on the desynced ASCAD data-set with Cross Entropy (CROSS), Cross Entropy Ratio (CER) and Log-Likelihood Ratio (LLR) as loss function

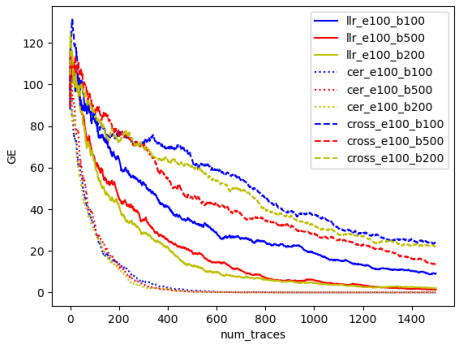
5.3 Experiments on the AES_HD data-set

The AES_HD data-set provides measurements of an unprotected implementation of AES on an FPGA.

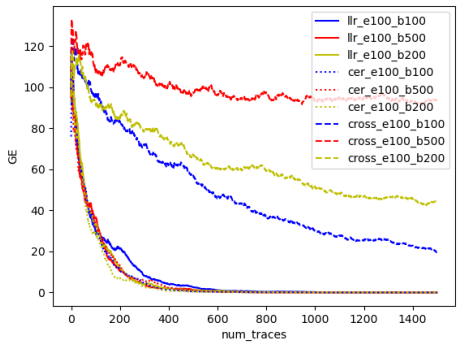
To test we again use the MLP model from Benadjila et al. [1], and the CNN model specific to this data-set of Zaid et al. [31]. The same configurations of hyperparameters as for the synchronized ASCAD set were used for the MLP, and for the CNN, 100 and 200 epochs were used.

For the training phase 45 000 traces were used as the training set, and 5 000 for the validation set. The remaining 25 000 traces were used as the attacking set to compute the Guessing Entropy.

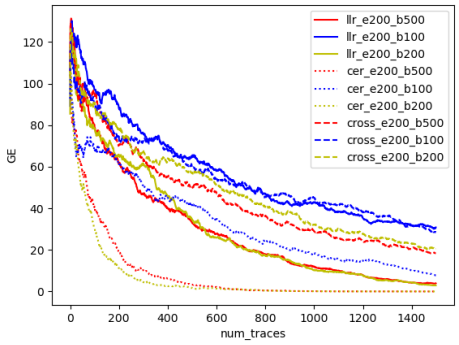
In Figure 5 it can again be seen that the models using CER as the loss function generally perform significantly better than the models using other loss functions. Additionally, in Figure 6 it can be seen that for the MLPs, the models using LLR perform about as well as the models using CER, but it can also be seen that the models using CER stop significantly earlier.



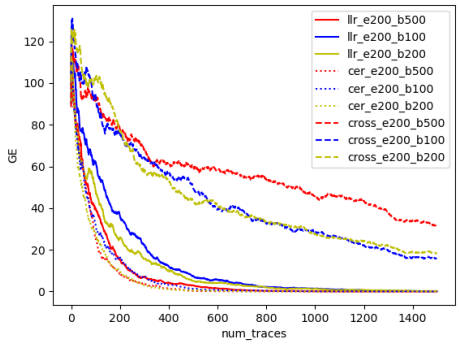
(a) 100 epochs CNN architecture



(b) 100 epochs MLP architecture

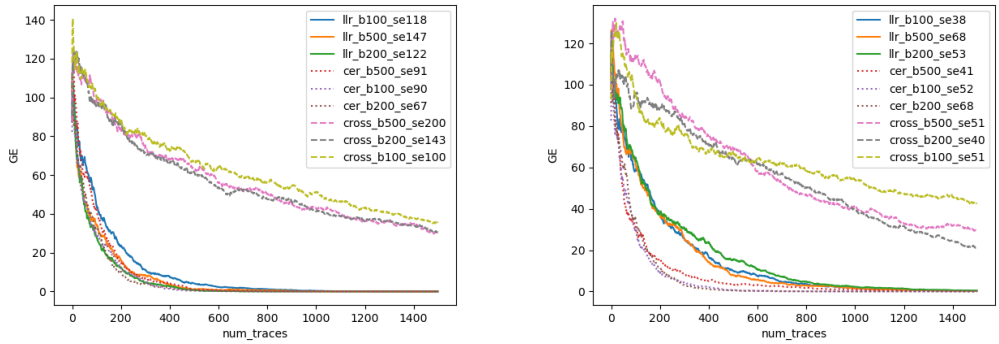


(c) 200 epochs CNN architecture



(d) 200 epochs MLP architecture

Figure 5: Results on the AES HD data-set with Cross Entropy (CROSS), Cross Entropy Ratio (CER) and Log-Likelihood Ratio (LLR) as loss function



(a) Stopped early MLP architecture

(b) Stopped early CNN architecture

Figure 6: Results on the AES_HD data-set with Cross Entropy (CROSS), Cross Entropy Ratio (CER) and Log-Likelihood Ratio (LLR) as loss function, stopped early

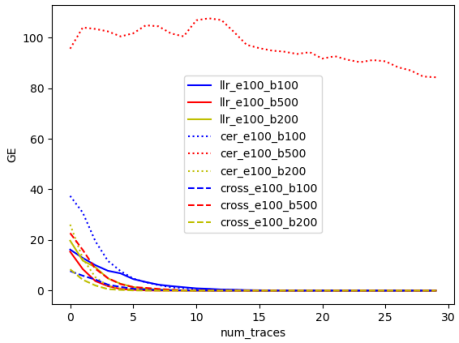
5.4 Experiments on the DPA_v4 data-set

The DPA_v4.2 data-set was used which provides a masked implementation of AES on an Atmel ATMega-163 smart card. The masks are assumed to be known here to simplify the attacks.

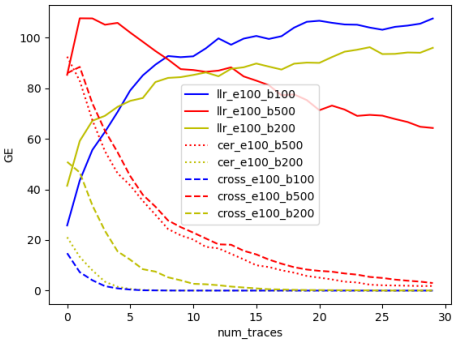
To test we again use the MLP model from Benadjila et al. [1], and the CNN model specific to this data-set of Zaid et al. [31]. The same configurations of hyper-parameters as for the synchronized AES_HD set were used for both of these models.

For the training phase 4 000 traces were used as the training set, and 500 for the validation set. The remaining 500 traces were used as the attacking set to compute the Guessing entropy.

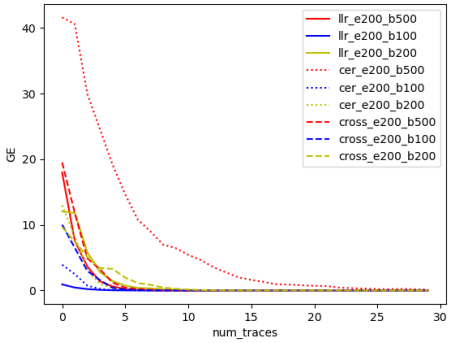
In Figure 7 the results significantly deviate from all of the other results. Firstly, for the CNNs, the model using CER and batch size 500 is significantly worse than all the other models for both 100 and 200 epochs. Additionally, for the MLPs, the models using LLR are not good enough to mount successful attacks. In Figure 8 it can also be seen that none of the models actually stopped early, and therefore the results are similar to the results in Figure 7, except that one of the MLP models using LLR is now able to mount a successful attack.



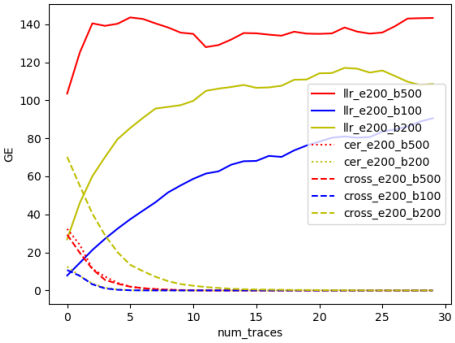
(a) 100 epochs CNN architecture



(b) 100 epochs MLP architecture

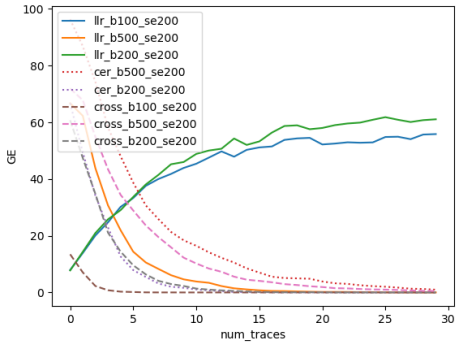


(c) 200 epochs CNN architecture

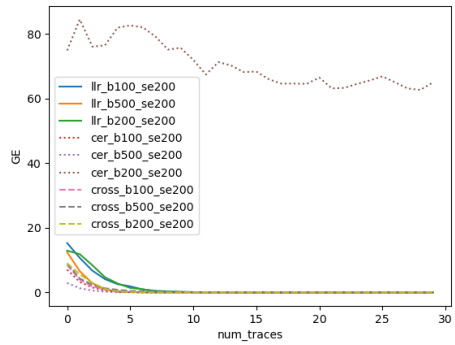


(d) 200 epochs MLP architecture

Figure 7: Results on the DPA_v4 data-set with Cross Entropy (CROSS), Cross Entropy Ratio (CER) and Log-Likelihood Ratio (LLR) as loss function



(a) Stopped early MLP architecture



(b) Stopped early CNN architecture

Figure 8: Results on the DPA_v4 data-set with Cross Entropy (CROSS), Cross Entropy Ratio (CER) and Log-Likelihood Ratio (LLR) as loss function, stopped early

5.5 Experiments on the AES_RD data-set

The AES_RD data-set is a set of measurements of an AES implementation with random delays introduced throughout the execution of the algorithm. Because of this, the MLP model that was used for the other data-sets is not powerful enough to conduct an attack on this data-set. Therefore the only model that was tested is the model specific to this data-set of Zaid et al. [31]. This model was tested for all three loss functions with a batch size of 50, a learning rate of 1e-3, and it was trained for 50 epochs. These hyperparameters were chosen as these are the ones used by Zaid et al. [31]. Only one set of hyperparameters was used for these models as they are computationally a lot more expensive to train than the models for the synced cases. For the training phase 20 000 traces were used as the training set, and 5 000 for the validation set. The remaining 25 000 traces were used as the attacking set to compute the Guessing Entropy.

In Figure 9 it can again be seen that the model using CER as the loss function performs better than both the model using LLR, and the model using regular Cross Entropy.

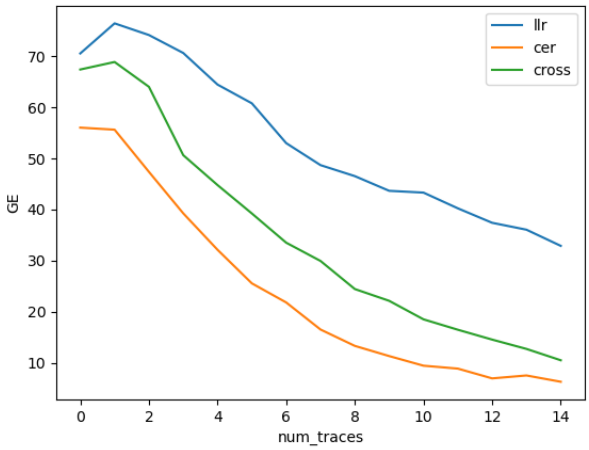


Figure 9: Results on the AES_RD data-set with Cross Entropy (CROSS), Cross Entropy Ratio (CER) and Log-Likelihood Ratio (LLR) as loss function

6 Discussion

The results showcased in the results section of the experiments on DPA_v4 are inconsistent with all the other results. The MLPs using LLR perform only slightly better than random guessing on this data-set and we have no real explanation for this. There are also some strange results regarding CER with a batch size of 500 and 200 epochs in this data-set for the CNN architecture. This could be explained by the fact that there are only 4 000 training samples, and that a larger batch size for CER resulted in bad results.

7 Conclusion and future work

As can be seen in the results using CER as a loss function in the context of side channel analysis is consistently better than using Cross Entropy. In addition to reproducing the results from Zhang et al. [33] several different CNN architectures were tested, and in almost every case CER used as a loss function outperforms Cross Entropy. LLR performs worse than CER in almost every scenario, but it is generally better than Cross Entropy. Some other works by Zaid et al. [30] have recently proposed another loss function specifically for side channel analysis. This loss function was however not tested for the case of imbalanced classes, but for the Identity leakage model it is shown that it outperforms both Cross Entropy and CER.

Additionally, the results show that the results from Zhang et al. [33] concerning CER as a metric are reproducible, and this means that CER could be used as a metric to more accurately evaluate Deep Learning models, even when classes are severely imbalanced.

In future work, the impact of early stopping on the effectiveness of side channel attacks could be investigated. Additionally, the effectiveness of LLR and CER as loss functions when the leakage model does not result in imbalanced classes could be investigated. Furthermore, the ranking-loss function [30] could be compared to CER and LLR as loss functions when classes are imbalanced. Finally, developing new loss functions using genetic programming to evolve new loss functions specifically for SCA could be interesting.

References

[1] 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. *ANSSI, France & CEA, LETI, MINATEC Campus, France. Online verfügbar unter https://eprint. iacr. org/2018/053. pdf, zuletzt geprüft am*, 22:2018, 2018.

[2] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 111–118, 2010.

[3] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *International workshop on cryptographic hardware and embedded systems*, pages 16–29. Springer, 2004.

[4] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 45–68. Springer, 2017.

[5] Omar Choudary and Markus G. Kuhn. Efficient template attacks. In *Smart Card Research and Advanced Applications*, pages 253–270. Springer International Publishing, 2014.

[6] B Gierlichs, L Batina, P Tuyls, and B Preneel. Mutual information analysis. a generic side-channel distinguisher. chesâ08, lncs, 2008.

[7] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.

[8] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[9] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Annual international cryptology conference*, pages 388–397. Springer, 1999.

[10] Paul C Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Annual International Cryptology Conference*, pages 104–113. Springer, 1996.

[11] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop bt-neural networks: Tricks of the trade. *Neural Networks: Tricks of the Trade*, 2012.

[12] Lingqiao Liu, Chunhua Shen, and Anton van den Hengel. The treasure beneath convolutional layers: Cross-convolutional-layer pooling for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4749–4757, 2015.

[13] James L Massey. Guessing and entropy. In *Proceedings of 1994 IEEE International Symposium on Information Theory*, page 204. IEEE, 1994.

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

[15] Ajinkya More. Survey of resampling techniques for improving classification performance in unbalanced datasets. *arXiv preprint arXiv:1608.06048*, 2016.

[16] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[17] V. Neagoe, A. Ciotec, and G. Cucu. Deep convolutional neural networks versus multilayer perceptron for financial prediction. In *2018 International Conference on Communications (COMM)*, pages 201–206, 2018.

[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):1–29, 2019.

[19] Martin Riedmiller. Advanced supervised learning in multi-layer perceptronsâfrom backpropagation to adaptive learning algorithms. *Computer Standards & Interfaces*, 16(3):265–278, 1994.

[20] Matthieu Rivain. On the exact success rate of side channel analysis in the gaussian model. In *International Workshop on Selected Areas in Cryptography*, pages 165–183. Springer, 2008.

[21] Sarat Kumar Sarvepalli. Deep learning in neural networks: The science behind an artificial brain, 10 2015.

[22] Werner Schindler, Kerstin Lemke, and Christof Paar. A stochastic model for differential side channel cryptanalysis. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 30–46. Springer, 2005.

[23] Sagar Sharma. Activation functions in neural networks. *towards data science*, 6, 2017.

[24] François-Xavier Standaert, Tal G Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 443–461. Springer, 2009.

[25] O-X Standaert, Eric Peeters, Gaël Rouvroy, and J-J Quisquater. An overview of power analysis attacks against field programmable gate arrays. *Proceedings of the IEEE*, 94(2):383–394, 2006.

[26] Yanmin Sun, Mohamed S Kamel, Andrew KC Wong, and Yang Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378, 2007.

[27] Tao Wang, David J Wu, Adam Coates, and Andrew Y Ng. End-to-end text recognition with convolutional neural networks. In *Proceedings of the 21st international conference on pattern recognition (ICPR2012)*, pages 3304–3308. IEEE, 2012.

[28] Lichao Wu, Léo Weissbart, Marina Krcek, Huimin Li, Guilherme Perin, Lejla Batina, and Stjepan Picek. On the attack evaluation and the generalization ability in profiling side-channel analysis. Technical report, Cryptology ePrint Archive, Report 2020/899, 2020. https://eprint. iacr. org âŠ, 2020.

[29] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Broeck. A semantic loss function for deep learning with symbolic knowledge. In *International Conference on Machine Learning*, pages 5502–5511. PMLR, 2018.

[30] Gabriel Zaid, Lilian Bossuet, FranÃ§ois 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.

[31] 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.

[32] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[33] 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.

[34] Ziyue Zhang, A Adam Ding, and Yunsi Fei. A fast and accurate guessing entropy estimation algorithm for full-key recovery. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 26–48, 2020.