Exploring Feature Selection Scenarios for Deep Learning-based Side-Channel Analysis

...and how much the model size matters

Guilherme Perin, Lichao Wu and Stjepan Picek

Delft University of Technology, The Netherlands

Abstract. One of the main promoted advantages of deep learning in profiling side-channel analysis is the possibility of skipping the feature engineering process. Despite that, most recent publications consider feature selection as the attacked interval from the side-channel measurements is pre-selected. This is similar to the worst-case security assumptions in security evaluations when the random secret shares (e.g., mask shares) are known during the profiling phase: an evaluator can identify points of interest locations and efficiently trim the trace interval. To broadly understand how feature selection impacts the performance of deep learning-based profiling attacks, this paper investigates four different feature selection scenarios that could be realistically used in practical security evaluations. The scenarios range from the minimum possible number of features to the whole available trace samples.
Our results emphasize that deep neural networks as profiling models show successful key recovery independently of explored feature selection scenarios against first-order masked software implementations of AES 128. Concerning the number of features, we found three main observations: 1) scenarios with less carefully selected point-of-interest and larger attacked trace intervals are the ones with better attack performance in terms of the required number of traces during the attack phase; 2) optimizing and reducing the number of features does not necessarily improve the chances to find good models from the hyperparameter search; and 3) in all explored feature selection scenarios, the random hyperparameter search always indicate a successful model with a single hidden layer for MLPs and two hidden layers for CNNs, which questions the reason for using complex models for the considered datasets. Our results demonstrate the key recovery with a single attack trace for all datasets for at least one of the feature selection scenarios.

Keywords: Side-channel Analysis · Deep learning · Feature Selection

1 Introduction

Side-channel analysis (SCA) explores unintentional leakage of information from electronic devices [MOP06]. Common targets are cryptographic algorithms executed on software (e.g., low-end IoT devices) or hardware (e.g., FPGAs or System-on-Chip) platforms. Among several proposed methods for side-channel attacks, differential power analysis (DPA) [KJJ99], correlation power analysis (CPA) [BCO04], and Mutual Information Analysis (MIA) [GBTP08] represent direct or non-profiling attacks. They require no knowledge about target implementation to deploy an attack. Nowadays, properly implemented masking [CJRR99, RP10] (at least second-order) and hiding [HOM06, CK10] (noise, shuffling, and misalignment) countermeasures represent strong protection combinations to defeat non-profiling attacks and deliver successful security evaluation results. On the other hand, profiling attacks assume a scenario where an adversary has a clone (open) target to learn approximated statistical distributions from the side-channel measurements. With it, the adversary can target the secret key of a second device, and the strength of this adversary is related to how much information the adversary has about the target implementation.

This information includes algorithm implementation details (e.g., source code) and access to random secret shares during profiling. Template attacks [CRR02], stochastic attacks [SLP05], and machine learning [LMBM13] are profiling attacks widely considered due to their practical aspects. However, they require feature extraction to locate points of interest (POI) representing the most leaking samples from measurements, which, in most cases, is only efficiently possible by knowing the secret random shares.

Deep learning has been widely explored as a competitive profiling attack [MPP16]. One of the main advantages of deep learning in profiling SCA is its possible deployment without pre-processing/feature engineering [MBC+20, LZC+21]. This means that raw measurements containing (typically) hundreds or thousands of sample points (features) are directly fed into a deep neural network, and the learning algorithm automatically detects the most leaking points. This is equivalent to concluding that POI selection would have a small impact on a security evaluation for the worst-case security assumptions. Whether this is true or not, we first need to understand how much POI selection impacts deep learning-based profiling attack results. Based on recently published results, this scenario would be practical against first-order masked implementations, including, for some cases, hiding countermeasures [CDP17]. To the best of our knowledge, no published work demonstrated the practical possibility of a successful deep learning-based profiling attack against second-order (or higher) masking schemes without some POI selection. In [MDP19] and [Tim19], the authors demonstrated that deep neural networks could fit three secret shares from simulated traces, but the same analysis was not demonstrated to be feasible with real side-channel measurements. *In the end, this still leaves unanswered the question of how much the POI selection helps (or interferes) in the attack performance of a deep neural network.*

To better understand how POI selection affects the outcome of a profiling attack, we define and explore the following feature selection scenarios:

- refined points of interest (RPOI) from Signal-to-Noise Ratio (SNR) peaks of the random secret shares;
- optimized points of interest (OPOI), where the attacked interval is the minimum possible interval that includes the main SNR peaks obtained from the known secret random shares;
- semi-optimized points of interest (SOPOI), where we assume that an adversary has no access to the random secret shares but has a rough idea of their timing location and selects a large window around this area;
- non-optimized points of interest (NOPOI), where the adversary has no access to the secret random shares and decides to apply profiling attacks over the full available trace interval.

RPOI and OPOI scenarios are aligned to the worst-case security evaluations. SOPOI and NOPOI relax the assumptions about the adversary and are closer to real-world attacks. Nevertheless, the complexity of deep neural networks is considered to be the same in all scenarios, i.e., up to eight hidden layers for multilayer perceptron (MLP) and convolutional neural network (CNNs). Additionally, when considering larger trace intervals, the model accuracy can be very high for the best-found deep learning models, and attacks can be successful with a single attack trace.

The main contributions of this paper are:

1. We explore four feature selection scenarios for profiling side-channel analysis that would be realistically adopted during security evaluations. The scenarios range from the worst-case security (adversary with knowledge about implementation and access to the random secret shares) to assumptions of a more relaxed adversary with limited knowledge about target implementation and no access to the secret random shares.

Results, therefore, indicate how much a specific feature selection scenario impacts the evaluation results;

2. We deploy extensive random hyperparameters search for each feature selection scenario on different datasets. For each specific combination of neural network type, leakage model, dataset, and feature selection, we conduct a random hyperparameter search of 500 models. We observed that each model training on a single GPU takes approximately 5 minutes to 1 hour. Therefore, we limited a random search to 500 models as this process takes up to 48 hours to complete for the worst case when considering multiple GPUs. We demonstrate that simple models containing up to eight hidden layers can successfully recover the key regardless of the feature selection scenario. We limit our analysis to eight hidden layers for two main reasons. First, this number of layers is based on attack performance provided by related works for ASCAD datasets [CDP17, KPH+19, ZBHV19, WAGP20, PCP20]. Second, adding more layers shows poor performance for the evaluated datasets and selected training settings (i.e., number of epochs, number of training traces). Nevertheless, our results indicate that neural networks with a single hidden layer in the case of multilayer perceptron and two hidden layers in the case of convolutional neural networks can successfully recover the key in all scenarios when feature selection is relaxed. This questions the need for adopting complex models for the same datasets or similar targets [LZC+21, WHJ+21]. We also apply the same hyperparameter search process to desynchronized traces with the NOPOI scenario and recover the key without defining new CNN hyperparameters ranges. This indicates that bypassing desynchronization countermeasures can be performed without the improvement of CNN architecture complexity.
3. For the NOPOI scenario, in which no feature selection is considered, we apply the same best found model for one AES key byte on the remaining key bytes. In this case, the model is initialized with the same weights and trained on each separate key byte. Our results show that we can recover the key with less than three attack traces for most key bytes.
4. We show that inconsistencies between accuracy and guessing entropy are also affected by feature selection when the target is a protected software implementation. In most cases, we show that attacking the minimum possible trace intervals (commonly considered in related works) is a situation where validation accuracy does not reflect attack performance. For longer attacked trace intervals, the neural networks may also be trained with more leaky points of interest, and validation accuracy shows to be a relevant metric for profiling SCA. This also sheds new perspective to insights from [PHJ+18] since, up to now, feature selection was not connected with the ability of machine learning metrics to provide a proper estimation of the SCA performance.

We provide the best-found MLP and CNN models for all feature selection scenarios in this link: https://github.com/AISyLab/feature_selection_ascad.

2 Background

2.1 Deep Learning-based Profiling SCA

Profiling techniques use the fact that side-channel measurements follow an unknown distribution that can only be approximated by an assumed statistical distribution for the leakage. The first approach for profiling attacks is the template attack, where an adversary assumes that the leakage follows a multi-variate Gaussian distribution [CRR02]. The profiling phase consists of computing statistical parameters for a Gaussian mixture model. Thus, the model is built for each possible hypothetical leakage class (e.g., all possible Hamming weight values of a byte). In the attack phase, the adversary computes

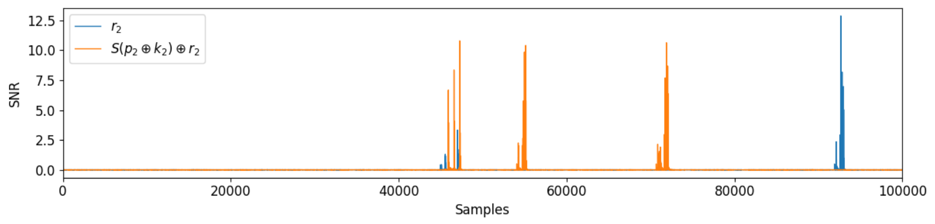


Figure 1: SNR of r and s_r secret shares for the **ASCADf** dataset.

the probability that a new side-channel measurement (under attack) belongs to a certain class by using the computed probability density function from the approximate statistics. While the profiling attack assumes a more powerful attacker than a non-profiling one, it requires significantly fewer traces than direct attacks to break the target: sometimes, only one trace is sufficient.

Machine learning methods learn the statistical parameters from data according to (usually) a limited number of tunable hyperparameters. This way, profiling attacks based on machine learning methods have the advantage of skipping the assumption about the statistical distribution of side-channel leakages. Additionally, deep neural network models represent functions that map input data $\mathcal{X}$ to output class probabilities $\hat{\mathcal{Y}}$. The mapping is performed by a function $f(\mathcal{X}, \theta) \rightarrow \hat{\mathcal{Y}}$, where θ is a set of parameters learned during the training phase. In the profiling SCA domain, $\mathcal{X}$ is a set of N side-channel traces, $\mathcal{X} = \mathrm{X_N}$, which is also a 2D-array with N rows and J columns. Each point in the array $\mathcal{X}$ is an element $t_{i,j}$, where i indicates the side-channel trace index and j indicates point index inside a side-channel trace i. Here, a *point* $t_{i,j}$ is also referred as a *sample* or *feature*.

The learned mapping between input side-channel traces $\mathrm{X_N}$ and outputs probabilities $\hat{\mathcal{Y}}$ depends on the estimated number of classes presented in $\mathrm{X_N}$. This number of classes, C, is derived from a leakage function implementing an operation processed inside the interval of J samples. The leakage function can provide an estimated leakage of an n-bit intermediate variable S, e.g., the Hamming weight or the Identity values.

2.2 Leakage Assessment of the Considered Software AES Datasets

In this section, we perform leakage assessment on the considered datasets. All datasets refer to side-channel measurements collected from AES 128 implementations. For the experiments conducted in this paper, we selected two publicly available SCA datasets containing side-channel measurements from 8-bit software devices.

ASCAD with a Fixed Key - ASCADf. **ASCADf** [1] dataset contains 60 000 side-channel measurements collected from an 8-bit ATMega device. All measurements are encryption operations with a fixed key. Each trace contains 100 000 sample points representing the electromagnetic emission from the first AES 128 encryption round. The implementation is protected with the first-order Boolean masking, and an S-box output byte is represented as:

$$s_{r,i} = Sbox(p_i \oplus k_i) \oplus r_i, \tag{1}$$

where r_i is a secret value randomly generated for each key byte and each encryption operation, p_i is the i-th plaintext, and k_i is the i-th key byte. Figure 1 shows the signal-to-noise ratio (SNR) of secret shares $s_{r,2} = Sbox(p_2 \oplus k_2) \oplus r_2$ and r_2.

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

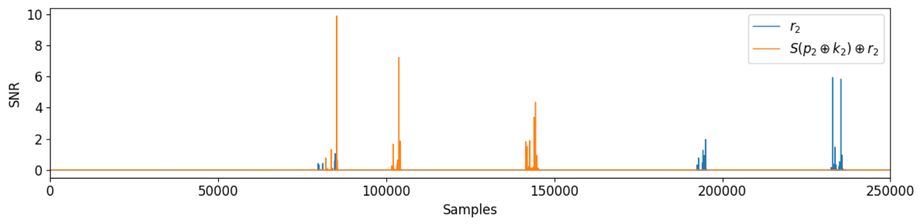


Figure 2: SNR of r and s_r secret shares for the **ASCADr** dataset.

In this work, from 60 000 traces, 50 000 are considered for profiling, 5 000 for validation, and 5 000 for the attack phase. For all the results with **ASCADf** dataset, guessing entropy is computed with 3 000 traces, which are randomly selected from the 5 000 traces in each separate key rank execution.

ASCAD with Random Keys - ASCADr. **ASCADr** [2] contains 300 000 traces collected from a software implementation of AES 128, where the first 200 000 measurements have random keys and 100 000 contain a fixed key. Each measurement contains 250 000 samples. The countermeasure on S-box output follows the same implementation indicated in Eq. (1). Figure 2 shows the Signal-To-Noise ratio (SNR) of secret shares $s_{r,2} = Sbox(p_2 \oplus k_2) \oplus r_2$ and r_2.

For this dataset, 200 000 traces are considered for profiling, 10 000 for validation, and 10 000 for the attack phase. For all the results with **ASCADr** dataset, guessing entropy is computed with 5 000 traces (in both validation and attack phases), which are randomly selected from the 10 000 traces in each separate key rank execution.

3 Related Works

Commonly used datasets in deep learning-based SCA publications represent side-channel acquisitions from software platforms. ASCAD datasets often appear as studied cases. The authors of these datasets released trace sets with raw measurements containing thousands of features. However, most recent studies consider trimmed versions of the traces strategically selected from where leakage (main SNR peaks of secret shares) is located [ZBHV19, WAGP20, BPS+20, PCP20]. This feature selection process is done based on the implementation details and the knowledge of secret mask shares. As such, we cannot assume that previous publications deployed deep learning attacks without points of interest selection.

In [MBC+20], authors demonstrated that CNNs could be efficient against side-channel measurements from software AES containing 160 000 sample points, indicating that attacking large-scale traces is not a limiting factor for deep learning-based SCA. Recently, the authors of [LZC+21] attacked the full trace intervals of software-based AES implementation, including ASCAD. The authors of [BCS21] attacked raw traces from **ASCADr** dataset by firstly selecting points of interest from the main SNR peaks of secret shares and later applying template attacks based on Linear Discriminant Analysis (LDA) [SA08]. These publications demonstrated the possibility of recovering a correct key byte with less than twenty attack traces ([BCS21] showed that the full key could be obtained with less than 32 attack traces), where different adversaries perspectives are assumed. [LZC+21] is the first appearance of an end-to-end deep learning-based profiling attack without any feature selection, where the authors considered highly complex deep neural networks, which include LTSM, attention, and convolution blocks. This may convey a possibly alarming

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

message that attacking raw measurements requires many neural network layers. As we show in this paper, we can recover a target key byte with a single attack trace without any feature selection by using very small MLP and CNN models.

4 Explored Feature Selection Scenarios

This section describes four specific scenarios to select points of interest for a deep learning-based profiling SCA. We start from the worst-case security assumption, where an adversary can identify points of interest based on implementation details and knowledge about mask shares. This assumption allows us to define two scenarios where the number of features is highly reduced. Next, we assume a slightly weaker adversary that does not know mask shares. However, the attacker can guess the location of the best attack interval from the full measured interval. The last scenario assumes that the adversary has no information about implementation and the complete measured intervals (raw traces) are attacked. Finally, we summarize the four scenarios in Table 1.

4.1 Refined Points Of Interest (RPOI)

In this scenario, we assume an adversary has access to the mask shares and has sufficient knowledge of the implementation details to select points of interest. The knowledge of implementation details is important to use the known random masks to compute all secret intermediates. This way, the adversary can compute the SNR of these intermediates and select Refined Points Of Interest (RPOI) from datasets indicating the most leaking samples. As we target the first-order protected AES implementations, the two sensitive variables representing the points of interest are assumed to be $s_r = S_{box}(p_i \oplus k_i) \oplus r$ and r.

The maximum number of RPOI is set to 100, in which 50 RPOI are selected from the highest SNR peaks obtained from $s_r = S_{box}(p_i \oplus k_i) \oplus r$ and the other 50 RPOI are selected from highest SNR peaks obtained from mask share r. We restricted the number of RPOIs to 100 to understand how the evaluated deep learning models perform against a highly reduced number of points of interest. Note that RPOIs are obtained from the SNR peaks observed from the full available trace interval. In Section 6, each profiling attack is applied to a minimum of 10 and a maximum of 100 RPOI. In each attack, we ensure that half of RPOI are selected from $s_r = S_{box}(p_i \oplus k_i) \oplus r$ leakages and the other half from r leakages. Intuitively, feature selection following the RPOI principle ends up attacking the minimum number of features. Our goal is to verify whether deep learning models benefit from the minimum number of input features that contain the maximum side-channel leakage information.

Objective 1. *Verify if the refined and minimized number of points of interest is beneficial for deep learning-based profiling attack performance.*

4.2 Optimized Points Of Interest (OPOI)

To select Optimized Points Of Interest (OPOI), we again assume the adversary has access to random secret shares and implementation knowledge. The main difference from RPOI scenario is that OPOI considers the minimum trace interval where the main SNR peaks from $s_r = S_{box}(p_i \oplus k_i) \oplus r$ and r are located.

This scenario follows most of the reported results in related works, such as [BPS+20, PCP20, ZBHV19, WAGP20]. Indeed, the ASCAD database provided datasets with the optimized points of interest selection. For **ASCADf**, the authors also released a separate optimized dataset by selecting 700 samples per trace, representing the most leaking interval. To do so, the authors evaluated the SNR of the intermediate values $s_r = S_{box}(p_2 \oplus k_2) \oplus r_2$ and r_2, as reported in [BPS+20]. Similarly, an optimized interval for **ASCADr** dataset is also provided, containing 1 400 samples per trace. Although the attacked interval is considered optimized (it contains the main SNR peaks from secret shares), the selected interval contains several noise samples, i.e., have low SNR values. A profiling model built from this interval should be, therefore, insensitive to the noisy samples. Therefore, the main reason for defining RPOI besides the OPOI is to understand how much deep learning models benefit from noisy samples' exclusion (or inclusion).

Objective 2. *Verify if selecting refined and minimized continuous trace interval including the main SNR peaks from the random secret shares is beneficial for deep learning-based profiling attack performance.*

4.3 Semi-optimized Points Of Interest (SOPOI)

We assume that an adversary selects a larger attack interval and guesses the points of interest location inside the complete measured interval. This assumption is realistic for several reasons. For instance, an adversary could split the most promising area of raw traces and attack several trace intervals window by window. Another example would be to conduct the plaintext correlation with side-channel measurements to identify the start interval of interest. Finally, an adversary could identify patterns inside the measurements indicating the AES activities, such as round executions (S-box, ShiftRows, MixColumns). In practice, the patterns can be obtained with low-pass filtering and/or resampling processes. Although not completely realistic in the black-box attack, this scenario is more practical than the OPOI case. One of the main goals of this work is to verify whether more samples are beneficial for a deep learning profiling model, so we create datasets with different trace interval sizes around the points of interest location from the OPOI case. By doing so, we may end up with two main situations: either a larger trace interval that includes more leaky point-of-interest related to $s_r = S_{box}(p_i \oplus k_i) \oplus r_i$ and r_i or the larger interval that includes only noisy samples. To solve this doubt, the evaluated datasets in the results section provide both situations.

Objective 3. *Verify if selecting an arbitrary trace interval that includes main SNR peaks from random secret shares and a significant number of noisy (non-leaky) samples is beneficial or irrelevant for deep learning-based profiling SCA.*

4.4 Non-optimized Points Of Interest (NOPOI)

To skip the feature selection process and profile over lengthy trace intervals, we also define a Non-Optimized Points Of Interest (NOPOI) scenario. This scenario was already considered in [LZC+21], where the authors proposed deep learning models that could break protected software AES implementations. Although results are competitive with state-of-the-art (they were able to recover the correct key byte from synchronized **ASCADf** and **ASCADr** with 6 and 8 attack traces, respectively), the models are very complex and contain more than 50 hidden layers (in particular, for attacking desynchronized **ASCADf** dataset, authors implemented a neural network with 56 hidden layers). From the adversary's perspective, attacking the complete measured trace interval (and leaving to the model the difficult feature selection task) is very advantageous. However, training deep neural networks with tenths of hidden layers may be very challenging and, depending on the attack circumstances, impractical due to time and memory constraints.

Attacking raw traces without following any pre-processing could sound counter-intuitive from the practical perspective. The datasets evaluated in this paper (and also in [LZC+21]) have trace lengths of 100 000 (**ASCADf**) and 250 000 (**ASCADr**) samples. The work of [WAGP20] already demonstrated the efficiency of deep learning architectures when the first layer is composed of *average-pooling* layer that, in the end, implements a window resampling of the input traces. Essentially, delivering this resampling task to the neural

Table 1: Possible feature selection scenarios for deep learning-based SCA with the synchronized measurements.

Scenario	Knowledge of r mask share	POI selection and pre-processing	Noisy/non-leaking samples
RPOI	Yes	Main SNR peaks of r and s_r. No pre-processing required.	No
OPOI	Yes	Minimum trace interval including SNR peaks of r and s_r. No pre-processing required.	Reduced
SOPOI	No	Large trace interval including main SNR peaks of r and s_r. No pre-processing required.	Significant
NOPOI	No	No POI selection and pre-processing is required.	All available

network is similar to performing trace resampling beforehand. Therefore, the proposed NOPOI scenario applies resampling to the input traces with a window of ten samples[3], resulting in trace lengths of 10 000 and 25 000 for **ASCADf** and **ASCADr**, respectively. Obviously, in this case, a first average-pooling (as considered in [WAGP20]) is not necessary anymore. Furthermore, as shown in the results section, a single hidden layer is sufficient to achieve results that are better than [LZC+21], questioning the reason for using very deep models for these datasets.

Objective 4. *Verify if attacking complete available trace interval with sub-sampling is beneficial for deep learning-based profiling SCA.*

5 Methodology for Model Selection

5.1 Model Selection in Profiling SCA

Model or algorithm selection is an important and analytically difficult part of a deep learning analysis for any domain. In the context of profiling SCA, recent publications usually follow one of three approaches:

1. Find the smallest possible model for a specific dataset [ZBHV19, WAGP20].
2. Small models selected from a short-term hyperparameter search [PCP20, BPS+20, RWPP21, WPP20].
3. Large models for more difficult problems (trace desynchronization bypassing and denoising) [LZC+21, WHJ+21].

The first approach requires more knowledge on the effect of hyperparameters on the learning process, and usually, several different configurations (which in a few cases include a grid search process as is the case of [ZBHV19]) are tested until the best hyperparameters are found. The second approach is more automated. Additionally, search algorithms are used to relax the expertise assumption required to understand hyperparameters effects, even if optimized ranges are required for a more efficient hyperparameters search. For the third approach, more complex models are defined to bypass hiding countermeasures, and more expertise is required for defining the hyperparameters.

From a literature review on different model selections, it is still difficult to conclude whether the number of features impacts the performance of a model. Of course, this conclusion cannot be done by only looking at the model's performance for a few datasets. None of the publications mentioned in this section evaluated the performance of a model for different feature selection scenarios on the same dataset. Therefore, in this paper, the hyperparameters related to the model size (number of layers, filters, kernels, strides, and

[3]Trace resampling is performed with **resample** module from **scipy.signal** python package v1.7.0.

Table 2: Random search configuration for MLPs.

Searched Hyperparameters	
Hyper-parameter	**Options**
Optimizer	`Adam`, `RMSprop`
Dense (or Fully-Connected) Layers	1, 2, 3, 4, 5, 6, 7, 8
Neurons	10, 20, 50, 100, 200, 300, 400, 500
Activation Function	`SeLU`, `ReLU`
Learning Rate	1e-3, 5e-3, 1e-4, 5e-4
Batch Size	100 to 1 000 (step: 100)
Weight Initialization	`random`, `glorot` or `he uniform`

neurons) are the same across all feature selection scenarios. By doing this, we aim to empirically provide evidence of whether the model size is highly affected by the different number of features.

5.2 Random Hyperparameter Search

We follow the second approach listed in section 5.1 for the model selection since we aim 1) at defining an algorithm selection process independent of the evaluated dataset and 2) at verifying how a unique hyperparameter search process performs across multiple feature selection scenarios. We perform a random hyperparameter search for MLP and CNN models. Our search space allows the selection of a deep neural network with up to eight hidden layers. For each specific number of points of interest, we search for 500 different models. This process is separately applied to MLPs and CNNs architectures for the Hamming weight and Identity leakage models.

Tables 2 and 3 list the covered search space for each hyperparameter. MLP models are randomly selected to contain at most eight hidden (dense) layer and all layers are defined with the same number of neurons. Based on the best MLPs reported in literature [PCP20, BPS+20], we only allow two possible activation functions, namely `SeLU` or `ReLU`. To define the optimizer, we consider only `Adam` and `RMSprop`. Also, we allow the search for different weight initialization options, which are `random`, `glorot` or `he uniform`. The options for CNN require the definition of convolution and pooling layer hyperparameters. The maximum number of hidden layers for CNNs is eight (excluding pooling and batch normalization layers from the counting), which we define as a maximum of four convolution layers and four dense layers. Again, for CNNs, dense layers will all contain the same number of neurons. The number of filters in a convolution layer is always twice the number of filters from the previous convolution layer. For both MLP and CNN cases, learning rate and batch size are not fixed but also included in the random search process. For all cases, the only fixed hyperparameter is the number of epochs, which is always set to 100. This number of epochs is aligned with related works, and, in our experiments, we observed that training for more than 100 epochs either leads to overfitting or shows no performance improvements for the considered model sizes. The only regularization method always present is the batch normalization layer after each pooling layer in CNNs.

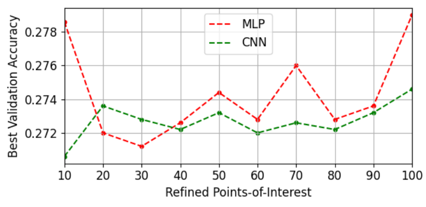
6 Results

6.1 ASCADf

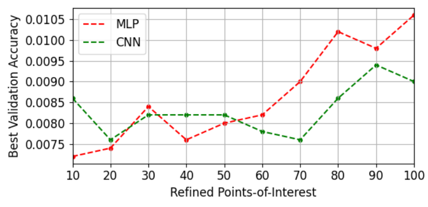
RPOI. Figure 3 shows the maximum validation accuracy obtained after deploying a random search of 500 models for the dataset that results from feature selection based on refined points of interest. Note that the random search is applied to different numbers of RPOI, starting from 10 and increasing it until 100 points. Results from Figures 3a and 3b indicate that the Hamming weight leakage model does not provide good results for reduced number of points of interest. We can only observe that the maximum number

Table 3: Random search configuration for CNNs.

Searched Hyperparameters	
Hyper-parameter	**Options**
Optimizer	`Adam`, `RMSprop`
Convolution Layers	1, 2, 3, 4
Convolution Filters	$4*2^{i-1}$, $8*2^{i-1}$, $12*2^{i-1}$, $16*2^{i-1}$ (i = conv. layer index)
Convolution Kernel	1 to nf (nf=10 if RPOI, nf=50 otherwise)
Convolution Stride	1 to nf (nf=10 if RPOI, nf=50 otherwise)
Pooling Type	`maxpooling`, `avgpooling`
Pooling Size	2, 4, 6, 8, 10
Pooling Stride	Pooling Size
Dense (or Fully-Connected) Layers	1, 2, 3, 4
Neurons	10, 20, 50, 100, 200, 300, 400, 500
Activation Function	`SeLU`, `ReLU`
Learning Rate	1e-3, 5e-3, 1e-4, 5e-4
Batch-Size	100 to 1 000 (step: 100)
Weight Initialization	`random`, `glorot` or `he uniform`

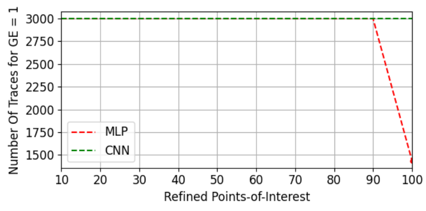


(a) Hamming weight.

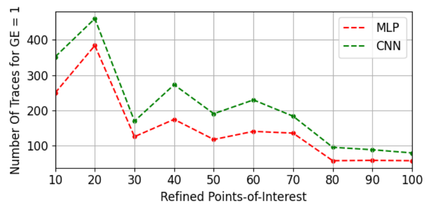


(b) Identity.

Figure 3: Refined Points Of Interest (RPOI): best validation accuracy results obtained from random hyperparameter search with the **ASCADf** dataset for different leakage models. Results are provided for MLP and CNN models.



(a) Hamming weight.

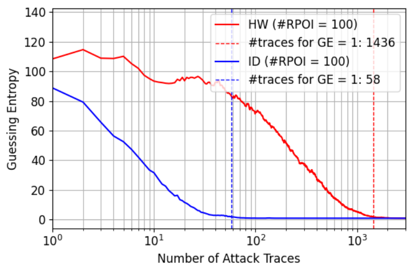


(b) Identity.

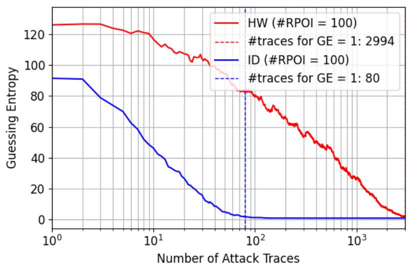
Figure 4: Refined Points Of Interest (RPOI): best number of attack traces to reach GE=1 obtained from random hyperparameter search with the **ASCADf** dataset for different leakage models. Results are provided for MLP and CNN models.

of RPOI (i.e., 100) leads to maximum validation accuracy for both MLP and CNN cases. When the Identity leakage model is considered, we can observe a more clear pattern, where increasing the number of RPOI increases the model performance.

The behavior for maximum validation accuracy from Figure 3 is reflected on the minimum number of attack traces to achieve guessing entropy of 1, as shown in Figure 4. Indeed, we could only successfully recover the key with the Hamming weight leakage model after 500 searched models if 100 RPOI are considered, and this result is illustrated in Figure 4a. Figure 4b shows the minimum required number of attack traces obtained with the Identity leakage model, clearly showing that increasing the number of RPOI provides a better attack performance. Also, we can observe similar behavior for CNN and MLP models.

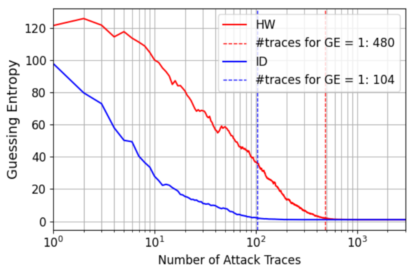


(a) MLP.

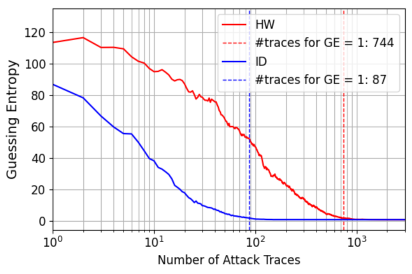


(b) CNN.

Figure 5: Refined Points Of Interest (RPOI):Best guessing entropy results for the **ASCADf** dataset with different leakage models.



(a) MLP.



(b) CNN.

Figure 6: Optimized points of interest (OPOI): best guessing entropy results for the the **ASCADf** dataset with different leakage models.

Figure 5 provides the guessing entropy for the best-found models from the executed random search. The Hamming weight leakage model in both MLP and CNN cases provides inferior results than the Identity leakage model. For the latter, successful key recovery is achieved after processing only 58 and 80 attack traces for best found MLP and CNN architectures, respectively.

OPOI. Results for the OPOI scenario are commonly reported in the literature with different deep neural network configurations [BPS$^+$20, ZBHV19, WAGP20, PCP20]. Here, we again apply the hyperparameter search process where the number of searches is limited to 500. The guessing entropy results for best-found models with OPOI scenario are shown in Figure 6. Results are aligned with state-of-the-art [ZBHV19, WAGP20], especially for the Identity leakage model. A first observation indicates that attacking a continuous and minimum interval that includes main SNR peaks from random secret shares provides superior results for the Hamming weight leakage model than the RPOI case. On the other hand, OPOI results for the Identity leakage model obtained with the best MLP and CNN models are inferior (but not too much) in comparison to results obtained with RPOI.

SOPOI. Semi-Optimized Points Of Interest (SOPOI) results in larger trace intervals that at least include the samples considered for the OPOI case. As defined in Objective 3, the main goal of this analysis is to understand how the significant addition of noisy samples interferes with the profiling attack performance. For that, we defined several SOPOI interval lengths ranging from 1 500 to 5 000 traces samples, with a step of 500 sample points. Figure 7 illustrates the SOPOI interval for the **ASCADf** dataset. As we can see, the trace intervals are constructed by leaving the previous OPOI interval as a reference.

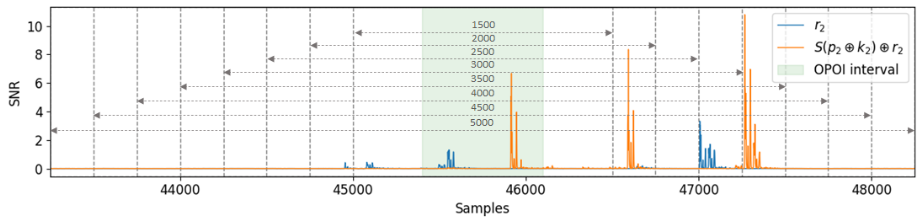
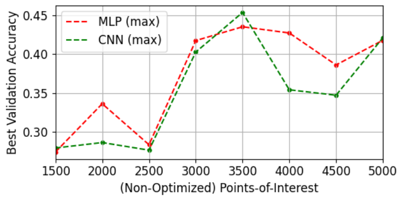
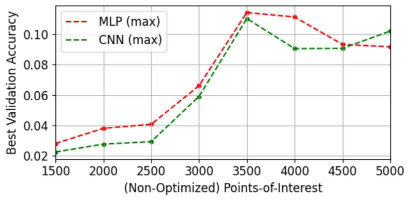


Figure 7: ASCADf SNR SOPOI. This figure shows how larger trace intervals are selected by having the OPOI trace set interval (highlighted in green) as a reference.



(a) Hamming weight.



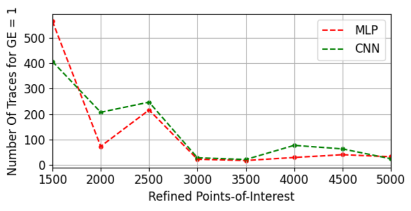
(b) Identity.

Figure 8: Semi-Optimized Points Of Interest (SOPOI): the best validation accuracy results obtained from a random hyperparameter search for the the **ASCADf** dataset and different leakage models. Results are provided for MLP and CNN models.

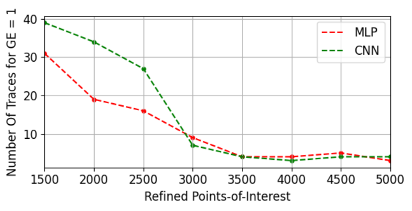
Notice also that the larger SOPOI interval also includes more leaky samples, and for this reason, we expect a higher attack performance (besides the fact that more noisy samples are also included).

Figure 8 shows the best-found validation accuracy for each different SOPOI interval. Figure 8a directly illustrates the effect of including more leaky samples in the attacked interval. As expected, when the length of attacked interval achieves 2 000 samples, leakage from $S(p_2 \oplus k_2) \oplus r_2$ is included, and the model search result in a model with higher validation accuracy (34%) in comparison to the best model from 1 500 sample interval (27.5%). When increasing the attacked interval to 2 500 samples, we see that the accuracy decreases again, as the extra 500 samples only contain noisy samples. However, when the attacked interval increases even more and more SNR peaks are included, the best model performance increases again until the attacked interval reaches 3 500 samples, which is the minimum interval to include all SNR peaks from Figure 7. Interestingly, increasing the interval to 5 000 by only including more noisy samples does not significantly deteriorate the best model performance. In conclusion, we may assume that deep learning models can deal quite well with a significant number of noisy samples in side-channel traces. The results for the Identity leakage model follow a similar pattern, as shown in Figure 8b, indicating that this behavior is not limited to a specific leakage model. An important remark from this analysis is that our best-found models achieved a validation accuracy that is significantly higher than random guess values for both Hamming weight (i.e., 27.5%, when all traces are classified as Hamming weight 4) and the Identity (i.e., 0.39%) leakage models. This already demonstrates the benefit of attacking datasets by considering a larger trace interval.

The evolution of the number of required attack traces to reach guessing entropy equal to 1 shows similar behavior to validation accuracy. As shown in Figure 9, the inclusion of more leaky samples until achieving 3 500 samples allow us to find better profiling models, and the further inclusion of noisy samples until we reach an interval of 5 000 samples does not reduce the chances of finding the best model that keeps similar performance.

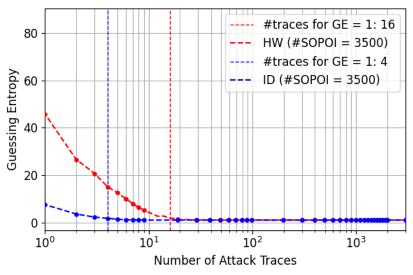


(a) Hamming weight.

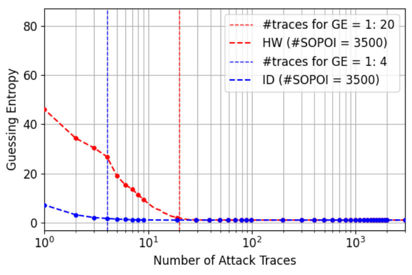


(b) Identity.

Figure 9: Semi-optimized Points Of Interest (SOPOI): best number of attack traces to reach GE=1 obtained from random hyperparameter search with the the **ASCADf** dataset for different leakage models. Results are provided for MLP and CNN models.



(a) MLP.



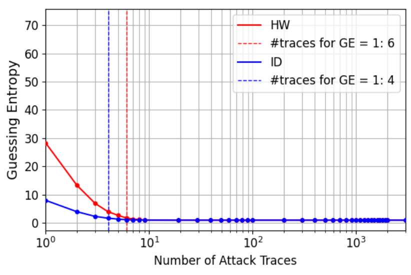
(b) CNN.

Figure 10: Semi-optimized points of interest (SOPOI): best guessing entropy results for the **ASCADf** dataset with different leakage models.

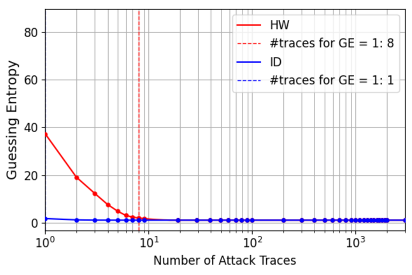
Finally, Figure 10 shows the guessing entropy for the best found model in the SOPOI scenario. We can recover the correct key byte after processing only 4 and 16 attack traces for the best-found MLP model in the Hamming weight and Identity leakage models, respectively. CNN case also shows similar performance for the Identity leakage model. Not surprisingly, the best-found model happens when we attack a trace interval of 3 500 samples.

NOPOI. Figure 11 shows the performance of the best-found models when the Non-optimized Points of Interest (NOPOI) scenario is considered. In this case, we apply a pre-processing step by resampling the raw traces into side-channel traces with 10 000 samples. Resampling provides benefits by reducing the number of trainable parameters in the input layer. The best found MLP with the Identity leakage model was able to successfully recover the correct key byte after processing only four attack traces, as illustrated in Figure 11a. At the same time, as shown in Figure 11b, *the best CNN model with the Identity leakage model requires a single attack trace to achieve GE equal to 1.* Exceptionally, for MLP cases, we also considered *l1* and *l2* regularization techniques in the random hyperparameter search. As **ASCADf** dataset contains a small number of profiling traces (i.e., 50 000), even small neural networks can overfit. As the NOPOI scenario considers all available features, the model could fit noise instead of leaky samples, as noisy samples are more numerous. This way, adding regularization prevents overfitting, improves generalization, and provides better attack results in comparison to results without *l1* and *l2* regularization.

To the best of our knowledge, this is the first time that a profiling attack was able to retrieve the correct key byte from the **ASCADf** *dataset with a single attack trace when no knowledge of mask shares are considered.*

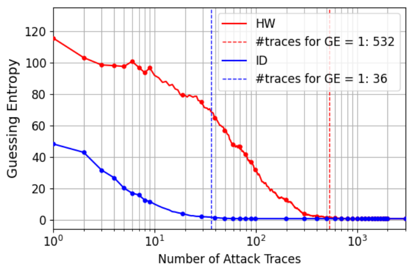


(a) MLP.

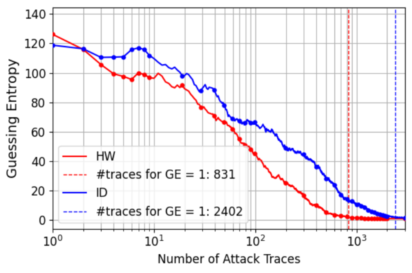


(b) CNN.

Figure 11: Non-optimized points of interest (NOPOI): best guessing entropy results for the **ASCADf** dataset with different leakage models.



(a) CNN with data augmentation.



(b) CNN without data augmentation.

Figure 12: Non-optimized points of interest (NOPOI) with desynchronization: best guessing entropy results for the **ASCADf** dataset with and without data augmentation.

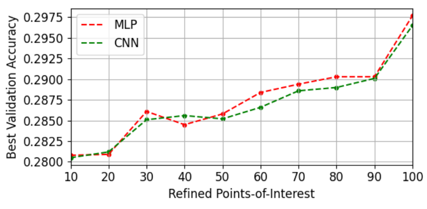
NOPOI with Trace Desynchronization. In this section, we also evaluate the possibility of attacking the full trace interval from **ASCADf** in the presence of trace desynchronization. For that, we take the original raw measurements from the **ASCADf** dataset that contains 100 000 samples per trace and perform artificial trace shifts. The number of shifted samples is randomly selected between -50 and 50. After, we again apply trace resampling, and this process results in side-channel measurements with 10 000 samples.

Figure 12 shows results for **ASCADf** with desynchronized side-channel traces. In Figure 12a, results were obtained when we apply data augmentation to the training process by randomly shifting the training traces to the left or to the right by up to 10 samples. As we can see, the attack is successful with 36 attack traces for the Identity leakage model. The Hamming leakage model requires 532 attack traces to reach guessing entropy equal to 1. On the other hand, when data augmentation is disregarded, we need a significantly higher number of attack traces to succeed.

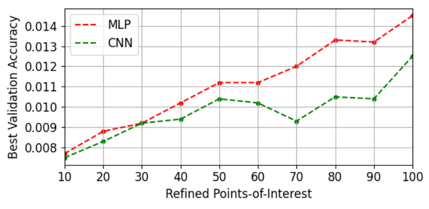
Attacking the Full AES Key with NOPOI Scenario. In this section, we retrain the best-found model with the NOPOI scenario (without desynchronization) for the full AES key. We apply this process for MLP and CNN, including the Hamming weight and Identity leakage models. The best model is always initialized with the same weights as the best model found in the random hyperparameter search process. Table 4 displays the number of required attack traces to reach guessing entropy equal to 1 for each separate key byte. As we can see, for the MLP case with the Hamming weight leakage model, the full AES key from **ASCADf** dataset is recovered with less than 20 attack traces.

Table 4: Minimum number of attack traces to obtain guessing entropy equal to 1 with the **ASCADf** dataset for all key bytes (results provided by MLP and CNN, for the Hamming weight and Identity leakage models).

Model Type	Leakage Model	Required Attack Traces (per key byte)
CNN	$Sbox(p_i \oplus k_i)$	1, 1, 1, >3000, 2, >3000, >3000, >3000, >3000.0, >3000, >3000, 1, >3000, >3000, 3, >3000
CNN	$HW(Sbox(p_i \oplus k_i))$	3, 3, 14, 6, 8, 7, >3000, 8, >3000, 7, >3000, 11, >3000, 40, 10, >3000
MLP	$Sbox(p_i \oplus k_i)$	1, 1, 4, 3, 3, 2, 3, 3, 7, 4, 4, 2, 14, 20, 2, 7
MLP	$HW(Sbox(p_i \oplus k_i))$	2, 3, 6, 6, 6, 5, 10, 6, 7, 6, 7, 6, >3000, 20, 7, 7



(a) Hamming weight.



(b) Identity.

Figure 13: Refined points of interest (RPOI): best validation accuracy results obtained from random hyperparameter search with the **ASCADr** dataset for different leakage models. Results are provided for MLP and CNN models.

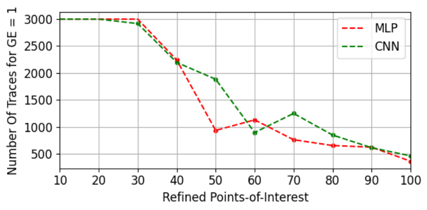
6.2 ASCADr

RPOI. Figure 13 shows validation accuracy results for RPOI scenario for the **ASCADr** dataset. Again, we evaluate ten different numbers of points of interest, ranging from 10 up to 100 points. For each specific number of RPOIs, we deploy a random hyperparameter search. The best found MLP and CNN models indicate a better performance as more points of interest are considered for the Hamming weight leakage model scenario. The **ASCADr** dataset contains 200 000 profiling traces, and this might be the reason for better results obtained with the Hamming weight leakage model in comparison to results obtained for **ASCADf**. For the Identity leakage model, results for the **ASCADr** dataset also indicate that more refined features improve the performance of the model.

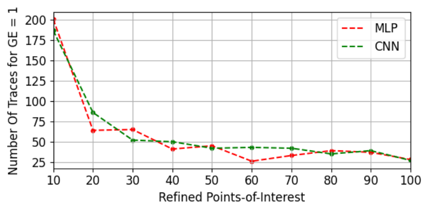
Figure 14 shows the relationship between the number of required attack traces to achieve guessing entropy equal to 1 and the number of RPOIs. For the Hamming weight leakage model, Figure 14a indicates that, for both best-found MLP and CNN models, at least 30 RPOIs are required for successful key recovery. This figure also indicates that more RPOI allows us to find better models. Figure 14b illustrates the number of required attack traces when the Identity leakage model is considered. In this case, we can recover the key for any considered number of RPOIs, and adding more than 20 RPOIs does not significantly improve the attack performance.

Figure 15 shows the guessing entropy evolution for the best found MLP and CNN models when the RPOI scenario is considered. Best models configured with the Identity leakage model show superior results compared to the Hamming weight leakage model. Note, however, that best models are always found for the maximum number of RPOI, i.e., 100, indicating that neural network models tend to learn better when more input features.

OPOI. Figure 16 shows the guessing entropy for the best models for the **ASCADr** dataset with OPOI feature selection scenario. Results obtained are aligned with state-of-the-art results [RWPP21] for **ASCADr** dataset for the same feature selection case. As we can see, using the Identity leakage model allows us to recover the correct key with only 78 attack traces when CNNs are considered. The best results are obtained for MLP with

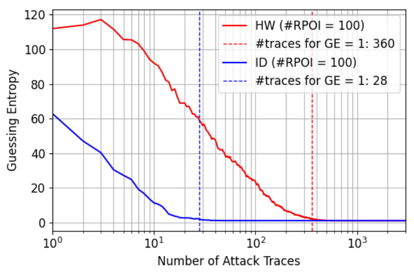


(a) Hamming weight.

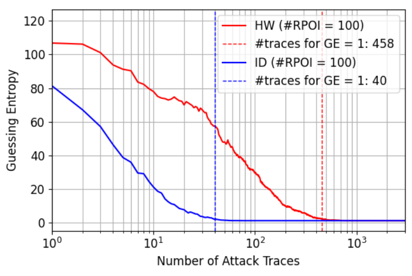


(b) Identity.

Figure 14: Refined Points Of Interest (RPOI): best number of attack traces to reach GE=1 obtained from random hyperparameter search with the **ASCADr** dataset for different leakage models. Results are provided for MLP and CNN models.

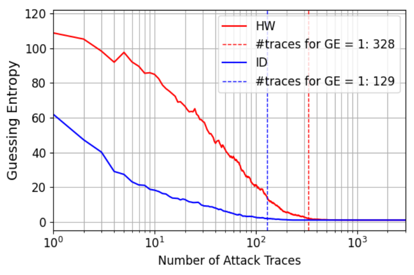


(a) MLP.

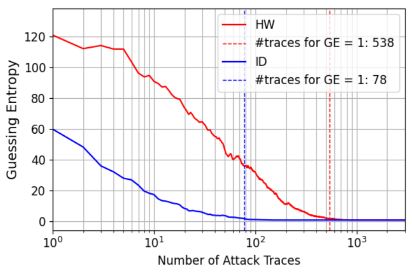


(b) CNN.

Figure 15: Refined points of interest (RPOI):Best guessing entropy results for the **ASCADr** dataset with different leakage models.



(a) MLP.



(b) CNN.

Figure 16: Optimized points of interest (OPOI): best guessing entropy results for the **ASCADr** dataset with different leakage models.

up to two hidden layers and CNN with up to 4 hidden layers.

SOPOI. To analyze the SOPOI feature selection scenario for the **ASCADr** dataset, we trimmed the interval from sample index 79 145 until 84 145 from the full available trace interval of 250 000 samples per trace. Figure 17 shows the SNR of secret shares computed over this SOPOI interval, which contains 5 000 sample points. This figure highlights the interval corresponding to the OPOI interval, commonly used in related works as the main attacked interval for deep learning-based profiling SCA. Following the same procedure executed for the **ASCADf** dataset, we run a random hyperparameter search for different trace intervals, ranging from 1 500 to 5 000 samples, with a step of 500 samples, as illustrated by the arrows in Figure 17. Again, the main idea is to verify the

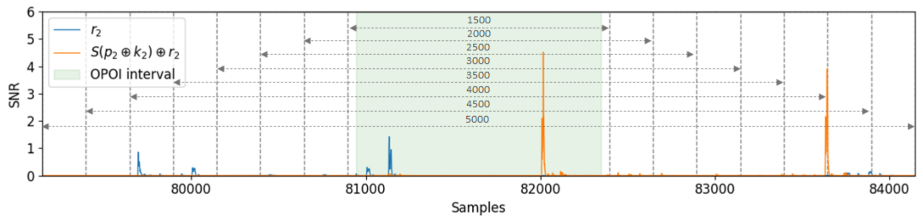
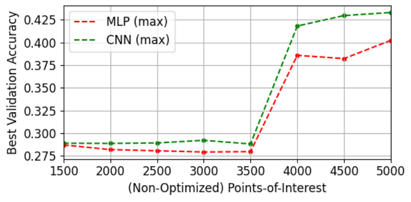
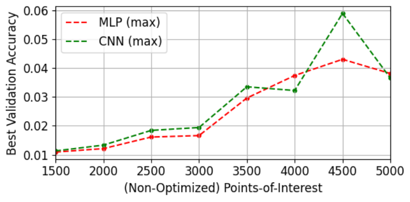


Figure 17: ASCADr SNR SOPOI. This figures shows how larger trace intervals are selected by having the OPOI trace set interval (highlighted in green) as reference.



(a) Hamming weight.



(b) Identity.

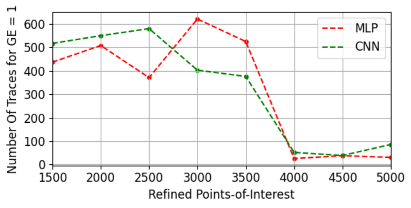
Figure 18: Semi-optimized Points Of Interest (SOPOI): best validation accuracy results obtained from random hyperparameter search with the **ASCADr** dataset for different leakage models. Results are provided for MLP and CNN models.

influence of additional noisy samples in deep neural network learning.

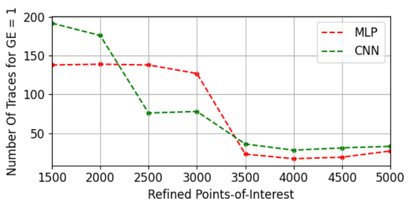
Figure 18 shows the best-found validation accuracy for each SOPOI interval. For the Hamming weight leakage model, as shown in Figure 18a, for both MLP and CNN cases, the best validation accuracy starts to show higher values after we consider SOPOI intervals higher than 3 500 samples. If we compare to Figure 17, we can observe that with 4 000 sample points, the attacked interval also includes other SNR peaks from secret shares. Until 3 500 samples, there is only the addition of noisy samples to the considered interval, and the best models show performance comparable to the OPOI case. For the CNN case, as illustrated in Figure 18b, the best validation accuracy gradually increases with the addition of more samples to the attacked SOPOI interval. Therefore, comparing MLP and CNN cases, we can immediately observe that MLP is more sensitive to the presence of non-leaky samples and can have a performance boost if additional leaky samples are added to the attacked interval.

Results in Figure 19 provide the minimum number of traces that the best model needs to reach a guessing entropy equal to 1. The minimum number of traces for each SOPOI interval follows a similar pattern observed for the best validation accuracy from 18. For the Hamming weight leakage model, the minimum number of traces reaches values lower than 100 after 4 000 samples are considered. This happens for both MLP and CNN scenarios. For the Identity leakage model, the addition of samples gradually improves the results for the CNN case. For the best-found MLPs, the addition of more samples provides successful attacks with less than 50 attack traces after 3 500 sample points are considered. Figure 20 shows the best guessing entropy results for SOPOI scenario. As we can see, the best found MLP and CNN models can recover the key with less than 40 attack traces.

NOPOI. In this scenario, we consider the full available samples from the **ASCADr** dataset traces (250 000). To improve the performance of deep neural network training and reduce memory overheads, we resample the full interval into 25 000 sample points. Figure 21 shows the guessing entropy for the best-found models in random hyperparameter

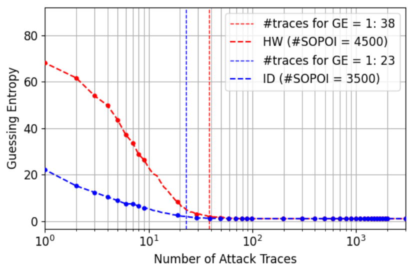


(a) Hamming weight.

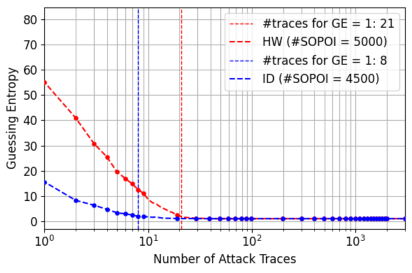


(b) Identity.

Figure 19: Semi-optimized points of interest (SOPOI): best number of attack traces to reach GE=1 obtained from random hyperparameter search with the **ASCADr** dataset for different leakage models. Results are provided for MLP and CNN models.

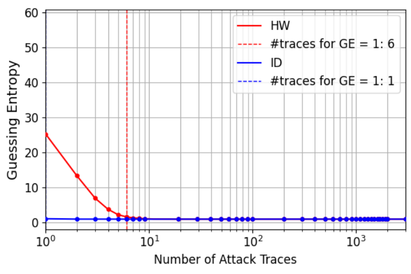


(a) MLP.

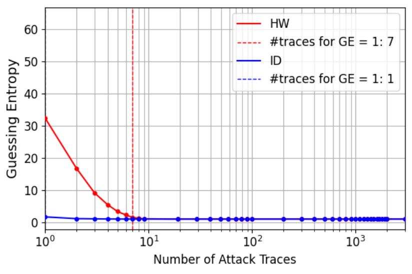


(b) CNN.

Figure 20: Semi-Optimized points of interest (SOPOI): best guessing entropy results for the **ASCADr** dataset with different leakage models.



(a) MLP.

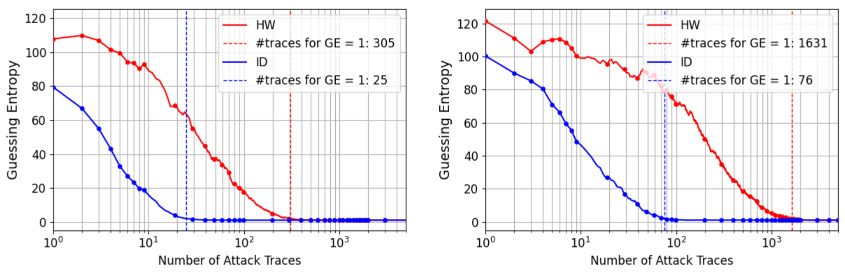


(b) CNN.

Figure 21: Non-optimized points of interest (NOPOI): best guessing entropy results for the **ASCADr** dataset with different leakage models.

search for MLP and CNN cases, with both the Hamming weight and Identity leakage models. *For the Identity leakage model, we can find the best neural network model that recovers the key with a single attack trace. Surprisingly, for the MLP case, the best model contains a single hidden layer, and this model can successfully recover the target key byte with only six attack traces. The best CNN was able to succeed with a single trace by having one convolution layer and three dense layers.* For the Hamming weight leakage model, we achieve guessing entropy equal to 1 after processing seven traces for the CNN case.

NOPOI with Trace Desynchronization. The desynchronization is artificially added to the dataset by shifting each trace to the left or the right. The number of shifted samples is randomly selected between -50 and 50. Afterward, each trace containing 250 000 samples



(a) CNN with data augmentation. **(b)** CNN without data augmentation.

Figure 22: Non-optimized points of interest (NOPOI) with desynchronization: best guessing entropy results for the **ASCADr** dataset with and without data augmentation.

Table 5: Minimum number of attack traces to get guessing entropy equal to 1 with the **ASCADr** dataset for all key bytes (results provided by MLP and CNN, for the Hamming weight and Identity leakage models).

Model Type	Leakage Model	Required Attack Traces (per key byte)
CNN	$Sbox(p_i \oplus k_i)$	1, 1, 2, 10, 1.0, 1, 5, 7, 4, 1, 3, 1, 19, 7, 1, 3
CNN	$HW(Sbox(p_i \oplus k_i))$	2, 2, 7, 6, 7, 45, 8, 8, 6, 6, 6, 88, 8, >5000, 6, 7
MLP	$Sbox(p_i \oplus k_i)$	1, 1, 1, 3, 1, 1, 1, 1, 1, 1, 1, 1, 3, 2, 1, 2
MLP	$HW(Sbox(p_i \oplus k_i))$	1, 2, 11, 10, 20, 6, 19, 13, 84, 13, 13, 11, 11, 14, 10, 13

is resampled into 25 000 samples. Results for the NOPOI scenario with desynchronized **ASCADr** dataset are shown in Figure 22. We apply random hyperparameter search with the same hyperparameters range from the previous experiments. Results from Figure 22a show best guessing entropy results for each leakage model when data augmentation is considered. Data augmentation randomly shifts the training traces by a maximum of 10 samples to the left and the right. As we can see, the correct key byte can be recovered after 25 attack traces with the Identity leakage model. Results in Figure 22b show the best guessing entropy obtained without data augmentation. With the Identity leakage model, we can recover the key with 76 attack traces. We can also recover the key for the Hamming weight leakage model with significantly more attack traces. As we can see, the usage of data augmentation again provides better results.

Attacking the Full AES Key with the NOPOI Scenario. As for the **ASCADf** dataset, in this section, we also retrain the best-found model with the NOPOI scenario and the **ASCADr** dataset (without desynchronization) for the full AES key. This process is also executed for MLP and CNN, including the Hamming weight and Identity leakage models. Again, the best model is always initialized with the same weights as the best model found in the random hyperparameter search process. Table 5 displays the number of required attack traces to reach guessing entropy equal to 1 for each separate key byte. As we can see, in three out of four attack scenarios, we can recover the full AES key. In particular, for the MLP case with the Hamming weight leakage model, the full AES key for the **ASCADr** dataset is recovered with less than three attack traces. *This is the best-reported attack results so far on the* **ASCADr** *dataset in the literature.* For comparison, in [BCS21], the authors reported their best profiling attack on the **ASCADr** dataset required at least 32 attack traces. Note also that our attack does not assume any knowledge from mask shares.

6.3 Summary of Results

Table 6 summarizes the results obtained in the experiments conducted in this section. This table provides the minimum number of attack traces required for guessing entropy

Table 6: Minimum number of attack traces to get guessing entropy equal to 1 with the **ASCADf** and **ASCADr** datasets on different feature selection scenarios.

Dataset	Model Type	Feature Selection	POIs (HW/ID)	Attack Traces (HW/ID)
ASCADf	MLP	RPOI	100/100	1436/58
ASCADf	CNN	RPOI	100/100	2994/80
ASCADf	MLP	OPOI	700/700	480/104
ASCADf	CNN	OPOI	700/700	744/87
ASCADf	MLP	SOPOI	3500/3500	16/4
ASCADf	CNN	SOPOI	3500/3500	20/4
ASCADf	MLP	NOPOI	10000/10000	6/4
ASCADf	CNN	NOPOI	10000/10000	8/ **1**
ASCADf	CNN	NOPOI desync	10000/10000	532/36
ASCADr	MLP	RPOI	100/100	360/28
ASCADr	CNN	RPOI	100/100	458/40
ASCADr	MLP	OPOI	1400/1400	328/129
ASCADr	CNN	OPOI	1400/1400	538/78
ASCADr	MLP	SOPOI	4500/3500	16/4
ASCADr	CNN	SOPOI	5000/4500	21/8
ASCADr	MLP	NOPOI	25000/25000	6/ **1**
ASCADr	CNN	NOPOI	25000/25000	7/ **1**
ASCADr	CNN	NOPOI desync	10000/10000	1631/73

equal to 1 for one target key byte (the third key byte of AES key). It is important to observe that scenarios that consider larger attack trace intervals require fewer traces to succeed. Following the same observation, the best attack results happen when considering the maximum analyzed number of points of interest for the RPOI scenario. In particular, the NOPOI scenario for both datasets requires a single attack trace to recover the key with the Identity leakage model. Results for OPOI are aligned with state-of-the-art results [ZBHV19, WAGP20], where the best case reaches successful key recovery with a number of attack traces close to 100.

7 Conclusions and Future Works

For the first-order protected software AES implementations, feature selection has a very small impact from the perspective of searching for an efficient deep learning model. Indeed, the larger the attacked interval (around the first AES round for datasets considered in this paper), the better the performance of the best-found model from a random hyperparameter search. Our results indicate that for the considered datasets, random hyperparameters search for MLP and CNN models can find highly efficient and small models (with up to eight hidden layers) that can recover the target key byte with a single trace. With the same hyperparameter search process, we successfully and efficiently recovered the key from scenarios with trace desynchronization and without any feature selection process. We found neural network models that recover the key with a single attack trace. Finally, the same best model that found one key byte can be directly applied to the remaining key bytes with similar performance. Thus, we can conclude that the number of features does not directly affect the required model size for successful key recovery. For many cases, a single hidden layer in MLP models and a single convolution layer plus one dense layer in CNN models are sufficient to provide a successful key recovery.

As future works, we plan to evaluate the performance of different feature selection scenarios against newer and more protected datasets, such as ASCADr2 [MS21]. In this case, we would like to anticipate that our statements about the relevance of knowing the mask shares for security evaluations could lead to different conclusions, as already evidenced in [MS21] (in which a successful profiling attack was only possible by assuming the knowledge of at least one mask share).

References

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.

[BCS21] Olivier Bronchain, Gaëtan Cassiers, and François-Xavier Standaert. Give me 5 minutes: Attacking ASCAD with a single side-channel trace. *IACR Cryptol. ePrint Arch.*, page 817, 2021.

[BPS$^+$20] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. Deep learning for side-channel analysis and introduction to ASCAD database. *J. Cryptographic Engineering*, 10(2):163–188, 2020.

[CDP17] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems – CHES 2017*, pages 45–68, Cham, 2017. Springer International Publishing.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.

[CK10] Jean-Sébastien Coron and Ilya Kizhvatov. Analysis and improvement of the random delay countermeasure of CHES 2009. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 95–109. Springer, 2010.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.

[GBTP08] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual information analysis. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, volume 5154 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2008.

[HOM06] Christoph Herbst, Elisabeth Oswald, and Stefan Mangard. An AES smart card implementation resistant to power analysis attacks. In Jianying Zhou, Moti Yung, and Feng Bao, editors, *Applied Cryptography and Network Security, 4th International Conference, ACNS 2006, Singapore, June 6-9, 2006, Proceedings*, volume 3989 of *Lecture Notes in Computer Science*, pages 239–252, 2006.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, pages 388–397, London, UK, UK, 1999. Springer-Verlag.

[KPH$^+$19] Jaehun Kim, Stjepan Picek, Annelie Heuser, Shivam Bhasin, and Alan Hanjalic. Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 148–179, 2019.

[LMBM13] Liran Lerman, Stephane Fernandes Medeiros, Gianluca Bontempi, and Olivier Markowitch. A Machine Learning Approach Against a Masked AES. In *CARDIS*, Lecture Notes in Computer Science. Springer, November 2013. Berlin, Germany.

[LZC$^+$21] Xiangjun Lu, Chi Zhang, Pei Cao, Dawu Gu, and Haining Lu. Pay attention to raw traces: A deep learning architecture for end-to-end profiling attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):235–274, 2021.

[MBC$^+$20] Loïc Masure, Nicolas Belleville, Eleonora Cagli, Marie-Angela Cornelie, Damien Couroussé, Cécile Dumas, and Laurent Maingault. Deep learning side-channel analysis on large-scale traces - A case study on a polymorphic AES. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider, editors, *Computer Security - ESORICS 2020 - 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14-18, 2020, Proceedings, Part I*, volume 12308 of *Lecture Notes in Computer Science*, pages 440–460. Springer, 2020.

[MDP19] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. Gradient visualization for general characterization in profiling attacks. In Ilia Polian and Marc Stöttinger, editors, *Constructive Side-Channel Analysis and Secure Design - 10th International Workshop, COSADE 2019, Darmstadt, Germany, April 3-5, 2019, Proceedings*, volume 11421 of *Lecture Notes in Computer Science*, pages 145–167. Springer, 2019.

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

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

[MS21] Loïc Masure and Rémi Strullu. Side channel analysis against the anssi's protected AES implementation on ARM. *IACR Cryptol. ePrint Arch.*, 2021:592, 2021.

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

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

[RP10] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010.

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

[SA08] François-Xavier Standaert and Cédric Archambeau. Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, volume 5154 of *Lecture Notes in Computer Science*, pages 411–425. Springer, 2008.

[SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. A stochastic model for differential side channel cryptanalysis. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, pages 30–46, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[Tim19] Benjamin Timon. Non-profiled deep learning-based side-channel attacks with sensitivity analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):107–131, 2019.

[WAGP20] Lennert Wouters, Victor Arribas, Benedikt Gierlichs, and Bart Preneel. Revisiting a methodology for efficient cnn architectures in profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(3):147–168, Jun. 2020.

[WHJ$^+$21] Yoo-Seung Won, Xiaolu Hou, Dirmanto Jap, Jakub Breier, and Shivam Bhasin. Back to the basics: Seamless integration of side-channel pre-processing in deep neural networks. *IEEE Trans. Inf. Forensics Secur.*, 16:3215–3227, 2021.

[WPP20] Lichao Wu, Guilherme Perin, and Stjepan Picek. I choose you: Automated hyperparameter tuning for deep learning-based side-channel analysis. *IACR Cryptol. ePrint Arch.*, 2020:1293, 2020.

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

A Summary of Results

This section provides the percentage of successful models from the hyperparameter search process applied to each feature selection scenario. In general, the chances to find deep learning models that provide successful key recovery increase when the number of hidden layers is reduced. Note that for most cases, a single hidden layer in MLP models and a single convolution layer plus one dense layer in CNN models is sufficient to provide successful key recovery.

Table 7: Influence of the number of hidden layers on the number of successful MLP models (key recovery) for **ASCADf**.

Model	Leakage					Number of Layers					
Type	Model	POI	1	2	3	4	5	6	7	8	Total
MLP	HW	10	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
MLP	HW	20	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
MLP	HW	30	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
MLP	HW	40	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
MLP	HW	50	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
MLP	HW	60	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
MLP	HW	70	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
MLP	HW	80	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
MLP	HW	90	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
MLP	HW	100	0.0%	0.0%	0.0%	0.0%	1.6%	1.5%	3.8%	1.6%	1.0%
MLP	HW	700	56.6%	82.0%	77.3%	70.6%	89.2%	62.7%	66.2%	58.2%	70.6%
MLP	HW	1500	74.5%	82.1%	83.8%	74.0%	78.0%	75.9%	69.6%	56.5%	73.3%
MLP	HW	2000	90.1%	88.9%	78.0%	83.9%	82.5%	78.8%	69.1%	63.0%	80.2%
MLP	HW	2500	83.6%	87.3%	77.8%	80.0%	72.9%	69.8%	57.6%	66.7%	74.2%
MLP	HW	3000	83.9%	89.5%	84.5%	85.1%	77.9%	77.0%	76.1%	58.2%	79.2%
MLP	HW	3500	87.8%	80.9%	89.8%	85.5%	74.5%	78.4%	77.8%	67.3%	80.3%
MLP	HW	4000	87.9%	73.8%	85.7%	85.5%	77.4%	80.9%	75.0%	68.4%	79.4%
MLP	HW	4500	93.5%	84.2%	85.0%	82.9%	72.5%	71.7%	75.5%	58.8%	78.2%
MLP	HW	5000	88.1%	68.9%	73.0%	87.5%	83.8%	87.2%	50.0%	70.5%	75.3%
MLP	HW	10000	37.1%	36.1%	37.9%	37.5%	23.1%	24.6%	15.6%	11.9%	28.5%
MLP	ID	10	77.4%	80.3%	66.2%	76.9%	59.6%	63.5%	55.1%	51.5%	66.2%
MLP	ID	20	72.7%	73.6%	70.1%	66.0%	60.0%	75.0%	57.7%	53.0%	66.6%
MLP	ID	30	78.0%	90.3%	92.5%	83.0%	76.1%	85.9%	85.9%	80.8%	83.8%
MLP	ID	40	74.2%	92.5%	86.0%	86.4%	85.7%	78.5%	82.8%	62.1%	80.6%
MLP	ID	50	69.5%	82.4%	94.5%	85.5%	85.7%	79.1%	78.6%	77.4%	81.8%
MLP	ID	60	60.8%	77.8%	90.6%	84.1%	80.0%	77.2%	75.7%	79.5%	78.4%
MLP	ID	70	63.3%	93.5%	86.7%	88.7%	75.6%	70.8%	70.1%	73.9%	77.8%
MLP	ID	80	78.3%	80.6%	86.0%	89.9%	87.7%	82.8%	82.3%	79.7%	83.4%
MLP	ID	90	78.9%	88.0%	89.3%	86.8%	85.5%	79.3%	74.6%	83.0%	83.2%
MLP	ID	100	72.9%	87.5%	88.5%	91.2%	89.6%	82.4%	75.8%	87.1%	84.4%
MLP	ID	700	70.8%	73.0%	77.1%	73.4%	61.4%	57.6%	57.6%	54.0%	66.0%
MLP	ID	1500	78.3%	64.3%	57.1%	55.2%	38.9%	40.0%	31.7%	17.2%	47.1%
MLP	ID	2000	71.8%	59.2%	50.8%	47.8%	56.2%	38.7%	28.3%	34.3%	49.2%
MLP	ID	2500	75.0%	52.2%	54.7%	38.4%	37.5%	24.6%	21.0%	25.8%	40.2%
MLP	ID	3000	70.4%	69.4%	46.9%	37.3%	44.2%	24.5%	25.0%	29.7%	43.4%
MLP	ID	3500	77.4%	76.1%	64.3%	50.0%	56.9%	50.0%	51.0%	45.8%	59.1%
MLP	ID	4000	83.3%	66.0%	74.4%	70.3%	50.0%	57.5%	61.5%	56.1%	63.8%
MLP	ID	4500	76.4%	59.5%	59.5%	54.3%	60.0%	46.2%	47.7%	36.4%	56.4%
MLP	ID	5000	75.0%	55.0%	60.0%	62.2%	57.9%	41.4%	33.3%	36.8%	52.6%
MLP	ID	10000	13.8%	1.8%	0.0%	0.0%	0.0%	0.0%	1.6%	0.0%	2.3%

Table 8: Influence of the number of hidden layers on the number of successful CNN models (key recovery) for **ASCADf**.

Model	Leakage		Number of Layers								
Type	Model	POI	1	2	3	4	5	6	7	8	Total
CNN	HW	10	-	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
CNN	HW	20	-	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
CNN	HW	30	-	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
CNN	HW	40	-	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
CNN	HW	50	-	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
CNN	HW	60	-	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
CNN	HW	70	-	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
CNN	HW	80	-	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
CNN	HW	90	-	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
CNN	HW	100	-	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
CNN	HW	700	-	27.8%	26.1%	40.8%	37.4%	29.0%	30.2%	31.6%	33.2%
CNN	HW	1500	-	50.0%	35.8%	37.1%	47.0%	37.3%	35.7%	28.6%	39.6%
CNN	HW	2000	-	39.4%	57.6%	47.7%	53.6%	40.2%	50.0%	38.9%	48.4%
CNN	HW	2500	-	50.0%	40.7%	53.6%	41.7%	42.7%	32.0%	24.0%	42.9%
CNN	HW	3000	-	55.6%	68.5%	55.8%	67.2%	50.0%	50.0%	38.5%	57.5%
CNN	HW	3500	-	65.4%	85.1%	75.5%	70.2%	60.0%	53.5%	65.2%	69.5%
CNN	HW	4000	-	86.4%	76.6%	70.8%	61.6%	60.5%	59.2%	50.0%	65.8%
CNN	HW	4500	-	69.7%	71.7%	78.2%	53.7%	57.0%	62.9%	50.0%	63.3%
CNN	HW	5000	-	81.2%	67.6%	75.0%	65.0%	52.6%	52.6%	43.3%	63.3%
CNN	HW	10000	-	23.3%	13.8%	16.5%	8.1%	6.1%	5.5%	0.0%	10.5%
CNN	ID	10	-	100.0%	89.5%	89.8%	87.1%	72.1%	71.7%	84.2%	84.5%
CNN	ID	20	-	87.5%	80.0%	73.4%	66.7%	76.0%	72.2%	65.4%	73.0%
CNN	ID	30	-	95.7%	92.9%	88.0%	79.3%	70.4%	82.1%	73.9%	81.7%
CNN	ID	40	-	87.0%	81.7%	78.6%	81.5%	74.0%	67.7%	60.9%	76.7%
CNN	ID	50	-	91.7%	86.4%	72.5%	75.0%	72.5%	73.8%	85.2%	77.3%
CNN	ID	60	-	81.8%	87.7%	78.9%	68.7%	64.4%	54.7%	75.0%	71.9%
CNN	ID	70	-	83.9%	82.1%	66.7%	75.5%	59.8%	72.6%	59.5%	70.2%
CNN	ID	80	-	75.0%	86.9%	83.3%	75.0%	64.4%	62.5%	40.9%	72.8%
CNN	ID	90	-	66.7%	78.2%	78.6%	70.1%	63.0%	60.7%	51.3%	68.7%
CNN	ID	100	-	86.5%	87.9%	85.4%	79.5%	67.7%	59.0%	52.9%	76.5%
CNN	ID	700	-	31.4%	31.7%	37.5%	30.9%	36.3%	34.1%	43.8%	34.2%
CNN	ID	1500	-	50.0%	32.7%	27.1%	26.0%	15.4%	21.3%	18.2%	25.3%
CNN	ID	2000	-	25.7%	19.1%	33.7%	30.2%	19.5%	20.8%	11.5%	25.1%
CNN	ID	2500	-	23.8%	33.8%	28.9%	21.1%	20.2%	19.3%	14.8%	23.7%
CNN	ID	3000	-	37.0%	50.0%	27.7%	20.8%	21.4%	20.0%	20.7%	26.3%
CNN	ID	3500	-	52.0%	61.9%	55.0%	38.1%	37.8%	34.4%	36.7%	44.3%
CNN	ID	4000	-	44.4%	47.1%	40.4%	39.8%	34.5%	33.3%	20.0%	38.2%
CNN	ID	4500	-	15.8%	58.5%	40.5%	35.0%	34.4%	31.3%	28.0%	37.5%
CNN	ID	5000	-	26.9%	45.3%	36.5%	35.4%	30.2%	18.6%	25.9%	33.4%
CNN	ID	10000	-	11.1%	3.4%	1.0%	2.4%	2.3%	0.0%	0.0%	2.5%

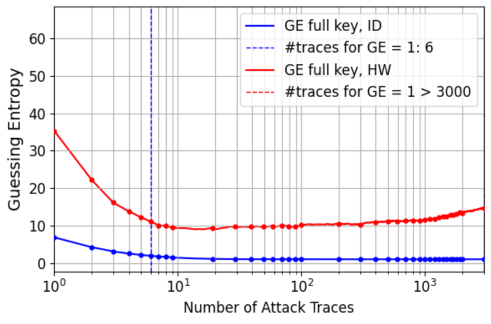
Table 9: The influence of the number of hidden layers on the number of successful models (key recovery) for **ASCADr**.

Model	Leakage		Number of Layers								
Type	Model	POI	1	2	3	4	5	6	7	8	Total
MLP	HW	10	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
MLP	HW	20	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
MLP	HW	30	0.0%	2.4%	4.9%	8.3%	9.8%	6.2%	13.9%	14.8%	7.1%
MLP	HW	40	30.0%	26.3%	28.0%	32.4%	23.7%	31.6%	20.0%	22.2%	26.6%
MLP	HW	50	78.8%	77.4%	75.5%	72.5%	49.2%	66.0%	47.9%	58.5%	64.9%
MLP	HW	60	83.8%	81.6%	80.8%	70.6%	75.0%	52.4%	53.8%	57.5%	68.8%
MLP	HW	70	95.3%	92.5%	87.9%	88.9%	73.3%	73.5%	87.0%	74.3%	84.3%
MLP	HW	80	97.4%	93.3%	95.6%	90.0%	90.0%	78.0%	75.6%	65.2%	85.5%
MLP	HW	90	97.8%	97.8%	93.3%	95.7%	89.8%	88.0%	86.1%	71.8%	90.4%
MLP	HW	100	100.0%	100.0%	84.2%	93.1%	97.6%	91.7%	85.0%	84.9%	92.1%
MLP	HW	1400	73.4%	89.4%	78.9%	74.6%	62.1%	60.7%	66.1%	47.1%	68.2%
MLP	HW	1500	83.3%	85.1%	88.9%	62.9%	65.7%	58.8%	59.5%	32.1%	67.8%
MLP	HW	2000	70.0%	81.0%	84.8%	65.0%	57.8%	52.5%	51.4%	43.6%	63.0%
MLP	HW	2500	92.9%	77.1%	60.0%	51.2%	59.3%	51.5%	33.3%	53.8%	57.7%
MLP	HW	3000	64.3%	47.6%	78.6%	87.5%	66.7%	45.5%	30.8%	25.0%	54.6%
MLP	HW	3500	69.2%	82.9%	65.7%	71.0%	47.1%	53.8%	29.4%	30.8%	57.3%
MLP	HW	4000	85.7%	81.2%	81.2%	72.7%	54.5%	45.2%	58.8%	46.4%	66.5%
MLP	HW	4500	95.0%	81.8%	68.0%	83.3%	82.6%	70.8%	67.9%	41.7%	73.2%
MLP	HW	5000	100.0%	100.0%	72.7%	68.4%	69.2%	71.0%	50.0%	40.0%	70.3%
MLP	HW	25000	78.9%	50.0%	50.9%	34.4%	40.9%	32.6%	31.8%	31.6%	43.9%
MLP	ID	10	93.9%	91.9%	92.1%	89.5%	80.6%	69.4%	56.8%	61.3%	79.7%
MLP	ID	20	88.4%	100.0%	93.8%	93.9%	79.3%	82.9%	80.0%	79.2%	87.3%
MLP	ID	30	77.8%	91.3%	73.3%	87.5%	84.2%	86.1%	89.3%	73.5%	82.7%
MLP	ID	40	80.0%	100.0%	89.2%	93.5%	92.3%	84.2%	79.4%	87.2%	88.2%
MLP	ID	50	89.3%	83.3%	94.3%	89.7%	80.0%	87.5%	71.0%	76.5%	84.1%
MLP	ID	60	90.6%	88.9%	96.4%	100.0%	89.3%	88.5%	80.0%	79.4%	88.8%
MLP	ID	70	79.3%	94.6%	88.9%	88.6%	90.3%	93.5%	79.4%	82.5%	87.1%
MLP	ID	80	72.2%	85.2%	91.9%	84.4%	74.2%	83.9%	54.8%	87.1%	79.3%
MLP	ID	90	81.8%	100.0%	83.9%	88.2%	70.0%	84.4%	82.5%	71.4%	82.8%
MLP	ID	100	80.9%	80.8%	87.5%	91.2%	81.2%	80.5%	81.1%	70.0%	81.5%
MLP	ID	1400	53.8%	46.7%	32.7%	39.7%	34.7%	25.9%	22.9%	26.8%	35.3%
MLP	ID	1500	40.0%	37.9%	32.0%	43.5%	47.1%	17.4%	24.0%	6.5%	29.8%
MLP	ID	2000	42.4%	28.0%	24.4%	21.6%	18.4%	27.5%	15.6%	19.4%	24.5%
MLP	ID	2500	31.8%	32.3%	20.9%	20.0%	21.2%	7.7%	24.2%	16.7%	22.5%
MLP	ID	3000	23.1%	8.1%	7.7%	11.5%	14.8%	11.1%	0.0%	2.9%	9.1%
MLP	ID	3500	13.9%	3.7%	26.5%	8.3%	16.7%	13.3%	14.3%	7.4%	13.6%
MLP	ID	4000	26.3%	25.0%	26.1%	26.5%	5.6%	17.2%	15.8%	20.0%	21.1%
MLP	ID	4500	46.7%	21.7%	31.8%	33.3%	16.1%	20.0%	18.2%	6.2%	23.5%
MLP	ID	5000	60.0%	50.0%	7.1%	11.1%	6.2%	7.1%	27.8%	12.5%	19.8%
MLP	ID	25000	35.1%	5.2%	4.3%	3.3%	6.7%	1.4%	4.0%	1.3%	7.2%

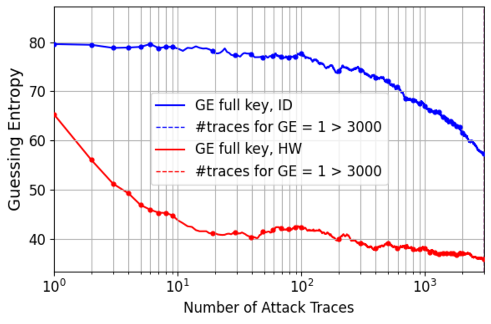
Table 10: Influence of the number of hidden layers on the number of successful models (key recovery) for **ASCADr**.

Model Type	Leakage Model	POI	Number of Layers								Total
			1	2	3	4	5	6	7	8	
CNN	HW	10	-	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
CNN	HW	20	-	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
CNN	HW	30	-	0.0%	0.0%	0.0%	0.0%	1.1%	0.0%	0.0%	0.2%
CNN	HW	40	-	5.1%	1.8%	2.9%	4.3%	2.2%	1.6%	4.0%	3.1%
CNN	HW	50	-	20.7%	17.6%	22.2%	23.6%	20.2%	18.0%	11.4%	20.3%
CNN	HW	60	-	18.8%	23.4%	30.9%	24.2%	26.2%	25.0%	25.0%	25.5%
CNN	HW	70	-	40.0%	53.6%	51.9%	50.0%	50.6%	34.8%	44.0%	48.0%
CNN	HW	80	-	69.0%	70.8%	67.0%	61.4%	64.4%	46.8%	50.0%	62.9%
CNN	HW	90	-	78.6%	61.4%	71.7%	67.2%	71.2%	59.1%	48.0%	66.7%
CNN	HW	100	-	69.4%	77.0%	74.3%	75.2%	60.5%	54.4%	65.2%	68.6%
CNN	HW	1400	-	40.0%	47.3%	55.3%	34.7%	49.3%	36.9%	16.7%	43.2%
CNN	HW	1500	-	36.0%	41.7%	51.8%	34.6%	45.7%	48.4%	46.2%	43.6%
CNN	HW	2000	-	25.0%	34.9%	37.7%	28.6%	23.9%	15.8%	12.5%	28.4%
CNN	HW	2500	-	28.6%	41.2%	48.6%	27.1%	24.5%	14.8%	13.3%	29.8%
CNN	HW	3000	-	10.0%	35.5%	31.8%	20.4%	17.2%	35.0%	0.0%	25.9%
CNN	HW	3500	-	25.0%	27.1%	32.5%	20.6%	23.1%	15.4%	18.2%	24.1%
CNN	HW	4000	-	52.2%	69.1%	52.4%	51.7%	31.9%	34.6%	20.7%	45.6%
CNN	HW	4500	-	53.6%	60.9%	60.2%	53.1%	40.7%	18.0%	25.0%	48.2%
CNN	HW	5000	-	65.6%	68.6%	59.0%	50.7%	47.9%	37.9%	22.7%	52.3%
CNN	HW	25000	-	20.0%	25.0%	50.0%	38.9%	25.0%	12.5%	0.0%	29.4%
CNN	ID	10	-	100.0%	100.0%	97.6%	93.9%	98.7%	94.2%	89.7%	96.3%
CNN	ID	20	-	100.0%	100.0%	99.0%	96.6%	93.5%	92.0%	81.2%	96.2%
CNN	ID	30	-	100.0%	100.0%	100.0%	98.1%	98.9%	90.4%	85.3%	97.1%
CNN	ID	40	-	100.0%	100.0%	98.7%	94.8%	93.0%	96.4%	94.9%	96.2%
CNN	ID	50	-	100.0%	98.3%	98.8%	94.9%	93.8%	90.2%	84.0%	95.1%
CNN	ID	60	-	100.0%	100.0%	100.0%	94.0%	94.4%	94.2%	91.3%	96.3%
CNN	ID	70	-	100.0%	98.1%	96.5%	96.7%	92.5%	80.4%	86.4%	93.5%
CNN	ID	80	-	100.0%	100.0%	97.1%	92.6%	84.1%	88.7%	77.8%	92.2%
CNN	ID	90	-	100.0%	100.0%	98.4%	92.2%	86.8%	91.5%	86.4%	93.1%
CNN	ID	100	-	100.0%	98.2%	95.1%	91.6%	88.7%	86.7%	83.3%	92.2%
CNN	ID	1400	-	33.3%	24.4%	26.7%	24.0%	26.6%	10.6%	10.0%	22.6%
CNN	ID	1500	-	11.1%	24.5%	18.0%	27.8%	29.2%	10.0%	0.0%	22.1%
CNN	ID	2000	-	14.3%	21.1%	10.9%	19.7%	13.0%	20.8%	16.7%	16.5%
CNN	ID	2500	-	25.0%	8.0%	12.2%	15.6%	15.6%	4.0%	0.0%	12.6%
CNN	ID	3000	-	0.0%	19.2%	11.4%	3.6%	12.0%	15.4%	0.0%	10.6%
CNN	ID	3500	-	7.4%	17.6%	25.4%	20.7%	20.5%	9.4%	21.7%	19.2%
CNN	ID	4000	-	37.5%	23.8%	23.4%	19.4%	17.8%	16.3%	4.8%	20.5%
CNN	ID	4500	-	20.7%	18.8%	15.2%	21.0%	18.3%	8.8%	5.9%	17.1%
CNN	ID	5000	-	3.8%	25.0%	16.9%	14.8%	18.5%	10.4%	2.9%	15.0%
CNN	ID	25000	-	0.0%	20.0%	6.2%	0.0%	0.0%	0.0%	0.0%	3.5%

B Guessing Entropy Results for Full Key with NOPOI Scenario

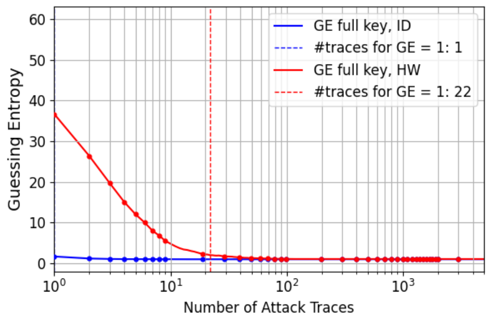


(a) MLP.

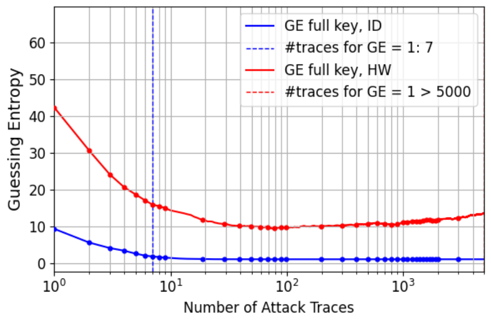


(b) CNN

Figure 23: Non-optimized points-of-interest (NOPOI) on the full AES key: best guessing entropy results for the **ASCADf** dataset with different leakage models.



(a) MLP.



(b) CNN

Figure 24: Non-optimized points-of-interest (NOPOI) on the full AES key: best guessing entropy results for the **ASCADr** dataset with different leakage models.