IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2022, No. 4, pp. 828–861. DOI:10.46586/tches.v2022.i4.828-861

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

Guilherme Perin[1,2], Lichao Wu[2] and Stjepan Picek[1]

[1] Radboud University, Nijmegen, The Netherlands picek.stjepan@gmail.com
[2] Delft University of Technology, Delft, The Netherlands {G.Perin,L.Wu-4}@tudelft.nl

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 three different feature selection scenarios that could be realistically used in practical security evaluations. The scenarios range from the minimum possible number of features (worst-case security assumptions) to the whole available traces. 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. First, we show that feature selection with the worst-case security assumptions results in optimal profiling models that are highly dependent on the number of features and signal-to-noise ratio levels. Second, we demonstrate that attacking raw side-channel measurements with small deep neural networks also provides optimal models, that shortens the gap between worst-case security evaluations and online (realistic) profiling attacks. In all explored feature selection scenarios, the hyperparameter search always indicates a successful model with up to eight hidden layers for MLPs and CNNs, suggesting that complex models are not required for the considered datasets. Our results demonstrate the key recovery with less than ten attack traces for all datasets for at least one of the feature selection scenarios. Additionally, in several cases, we can recover the target key with a single attack trace.

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 in software (e.g., low-end IoT devices) or hardware (e.g., FPGAs or System-on-Chip) platforms. Among several proposed methods for side-channel analysis, differential power analysis (DPA) [KJJ99], correlation power analysis (CPA) [BCO04], and Mutual Information Analysis (MIA) [GBTP08] represent direct or non-profiling attacks. Nowadays, properly implemented (at least second-order) masking [CJRR99, RP10] and hiding [HOM06, CK10] (noise, shuffling, and timing desynchronization) countermeasures represent strong protection combinations to defeat non-profiling attacks and deliver successful security evaluation results. Under these circumstances, an evaluator checks whether an attacker would be able to reduce the entropy of the secret key with a specific number of side-channel

Received: 2022-04-15 Accepted: 2022-06-15 Published: 2022-08-31

measurements, which is limited by the acquisition capacity of the evaluator and, in some situations, countermeasures. On the other hand, profiling attacks assume a scenario where an adversary has a clone or 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 estimated from the amount of information the adversary has about the target implementation.

Template attacks [CRR02], stochastic attacks [SLP05], and machine learning-based attacks [LMBM13] are profiling attacks widely considered due to their efficiency in practice. These profiling methods allow an evaluator to estimate the worst-case security for a target device. For that, knowledge about random secret shares and/or implementation source code is taken into account to select points of interest (POI) from side-channel measurements.[1] The quality of a profiling model is usually estimated with the minimum number of attack traces to recover the secret key [SMY09] and/or by considering efficient metrics that measure attack complexity, such as Perceived Information (PI) [BHM+19]. Such information-theoretic metrics allow evaluators to deploy a fair comparison among different profiling methods and estimate how much information the device is leaking. Of course, an optimal profiling model is easier to be implemented if points of interest selected from secret shares contain sufficiently high Signal-to-Noise Ratio (SNR) values. Therefore, under the worst-case security assumptions, noise becomes the main artifact to reduce the attack's efficiency.

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. Our work differs from the first one as there, the authors do not explore the influence of the feature selection techniques on the performance of deep learning. What is more, since the authors concentrate on a polymorphic AES, any direct result comparison becomes difficult. Also, the target AES implementation from [MBC+20] is not masked, contrary to the evaluated datasets in our paper. At the same time, the difference from the second work ([LZC+21]) is even starker as we additionally do not require large neural networks to break the target.

For cases when masking countermeasures are implemented, the leaking points should coincide with points of interest representing the processing of secret mask shares. This is equivalent to concluding that POI selection would have a small impact on a security evaluation with the worst-case security assumptions. Whether this is true or not, we first need to understand to what extent 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]. This means that deep neural networks are required to automatically detect the location of points of interest in two secret shares. Whether deep neural networks can automatically fit more than two secret shares without feature selection is still an open research question.[2] *In the end, this still leaves unanswered the question of how influential POI selection is for the profiling attack performance with deep neural networks.* To answer this question, we define and explore the following feature selection scenarios:

- we follow the worst-case security assumptions with refined points of interest (RPOI)

[1] Increasing the number of shares will make finding the location of POIs more difficult if the masks are not known. Still, for a small number of shares and a limited understanding of implementation, an exhaustive approach [RGV12] should work.

[2] To the best of our knowledge, no published work demonstrated the practical possibility of a successful deep learning-based profiling attack against the second or higher-order masking schemes without some POI selection. In [MDP19a] and [Tim19], the authors demonstrated that deep neural networks could fit three secret shares from simulated traces. In [Tim19], the authors also show that a CNN fits 3 shares from very high SNR ChipWhisperer traces.

that are selected from the highest SNR peaks of two main secret shares required to implement a second-order profiled attack. The main goal of this analysis is to understand the effect of the number of points of interest and their SNR for deep learning-based profiling attacks. More precisely, by increasing the number of RPOIs, we also take the risk of adding low SNR points, making the model fit the noise instead of leakages. We add different levels of Gaussian noise to the datasets and demonstrate that even when RPOIs only contain low SNR, deep neural networks can reach optimal results, and the limitation is only related to the number of hyperparameter search attempts;

- we consider optimized points of interest (OPOI), where the attacked dataset consists of an optimized interval (or the concatenation of two intervals) that includes the main SNR peaks obtained from the two known secret shares. This scenario provides insights into the limitations of profiling attacks when attacked interval includes limited numbers of high SNR points but also includes several low SNR points. We demonstrate that this feature selection scenario, which is extensively used in related works, more likely results in suboptimal deep learning models from a hyperparameter search.
- we evaluate the performance of deep learning-based profiling attacks with non-optimized points of interest (NOPOI). In this case, an evaluator relaxes the assumptions about the adversary and applies profiling attacks over full trace intervals. As this scenario includes all points of interest where some have very low SNR, the only limitation to implementing optimal profiling models is the number of hyperparameter search attempts. Although the performance of our best-found models indicates performances similar to those obtained with the RPOI case (for some cases, we recover the secret key with a single attack trace), we argue that deep learning with NOPOI can deal with an extensive number of features, but it still does not always replace the performance of the worst-case security evaluations. Indeed, selecting RPOI tends to make the evaluation faster, as we find optimal profiling models with fewer hyperparameter search attempts.

The three aforementioned feature selection scenarios can be realistically adopted during security evaluations. The case of NOPOI is also of interest to security evaluation labs that often cannot follow the worst-case security assumptions for certification purposes. Although it is a common procedure to access source code and confidential target-related documents during security evaluations, attacks are usually conducted in a black-box fashion as secret randomness are not always available to evaluators. Another important aspect of our analysis is the extensive hyperparameter search for each feature selection scenario. For instance, recent publications comparing different profiling methods under black-box and worst-case security assumptions indicate that multilayer perceptron performs as good as other state-of-the-art methods such as Gaussian template attacks or Gaussian-based soft-analytical side-channel attacks and that there are significant differences in performance according to adversary assumptions [BS20, BDMS22]. Here, we demonstrate that multilayer perceptrons and convolutional neural networks **can** achieve optimal profiling performance (with key recovery requiring a single attack trace), and the limitation is only related to the number of hyperparameter searches. We show 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 three main reasons. First, this number of layers is based on attack performance provided by related works for the same evaluated datasets [KPH+19, ZBHV19, WAGP20, PCP20]. Second, adding more layers shows poor performance for the evaluated datasets and for the selected training settings (i.e., number of epochs, number of training traces). Third, the creation of smaller neural networks may generate inherently self-regularized models due to their limited fitting capacity, also providing benefits and fewer tuning experiments as we do not

require explicit regularization. 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 when feature selection is disregarded. 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 necessarily improving the models' complexities. Finally, we verify that portability issues can also be overcome when not using feature selection.

We provide the best-found MLP and CNN models for all feature selection scenarios in Github repository: https://github.com/AISyLab/feature_selection_dlsca.

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 model (mean, variance, and co-variance). 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 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 can have the advantage of relaxing 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. 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 Quantitative Analysis of Profiling Models

To verify if a cryptographic implementation leaks side-channel information, the best way is to estimate the worst-case security during a security evaluation. In this type of analysis, the evaluator has conditions to obtain optimal points of interest from side-channel measurements, build a profiling model with sufficient measurements, and estimate the minimum number of attack traces required to recover the secret. Realistically, security evaluators also face situations when a security test is conducted in a black-box approach.

In this case, the chances of implementing a profiling model that precisely estimates the security of implementation becomes more difficult.

Different metrics are used to compare profiling models. Guessing entropy and success rate became mainstream approaches to estimate the capacity of a model to recover the secret from a certain number of measurements [SMY09]. In [BHM+19], the authors provided Perceived Information (PI) as the information metric to measure the attack efficiency. PI measures how much information a model can obtain from testing side-channel measurements and indicates the complexity of an attack in terms of a number of measurements. The PI calculation is provided as follows:

$$\widehat{PI}(X,Y) = H(Y) + \sum_{y \in Y} p(y) \frac{1}{N_y} \sum_{i=1}^{N_y} \log_2 \hat{p}(y|x_i^y), \tag{1}$$

where $H(Y)$ is the entropy of Y and $\hat{p}(y|x_i^y)$ is probability of a model to predict a trace x_i^y with a class y and N_y is the number of attack traces labeled as y for a key candidate k. Later, in [MDP19b], the authors demonstrated that minimizing cross-entropy loss function in neural networks is similar to maximizing the perceived information from a profiling model. The PI calculation also allows the evaluator to estimate the minimum number of required attack traces $\hat{N}_{\beta,PI}$ to recover the key:

$$\hat{N}_{\beta,PI} \geq \frac{f(\beta)}{\widehat{PI}(X,Y)}, \tag{2}$$

where $f(\beta)$ is a small constant related to the expected success rate β. For instance, when targeting intermediate variables processed by a n bit devices, $f(\beta)$ is given by [dCGRP19]:

$$f(\beta) = n - (1-\beta)\log_2(2^n - 1) + \beta \log_2(\beta) + (1-\beta)\log_2(1-\beta) \tag{3}$$

Following these observations, in our experiments, we also use the $\widehat{PI}(X,Y)$ metric to compare the performance of deep learning models for the worst-case security assumptions. The small constant $f(\beta)$ is always computed for $\beta = 0.8$ (i.e., success rate of 80%). Negative values of $\widehat{PI}(X,Y)$ indicate a wrong profiling model from the information-theoretic perspective, and this can also be verified by the validation loss function that tends to grow during training. Finally, we also estimate the required number of attack traces to reach guessing entropy equal to 1, $N_{GE=1}$. Note, however, that even models showing negative $\widehat{PI}(X,Y)$ values can deliver successful key recovery. In these cases, we assume the model is sub-optimal. When the profiling model delivers positive values of $\widehat{PI}(X,Y)$, the quantities $\hat{N}_{\beta,PI}$ and $N_{GE=1}$ should be very close.

2.3 Datasets

ASCAD with a Fixed Key - `ASCADf`. The `ASCADf` dataset contains 60 000 side-channel measurements collected from an 8-bit ATMega device.[3] 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 (we attack the third key byte, which is the first masked one).

In our 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 the `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.

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

ASCAD with Random Keys - `ASCADr`. `ASCADr` contains 300 000 traces collected from a software implementation of AES 128,[4] where the first 200 000 measurements have random keys and 100 000 contain a fixed key. Each measurement contains 250 000 samples.

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 the `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. We again attack the third key byte as it is the first masked one.

DPAContest 4.2 - `DPAv4.2` The `DPAv4.2` dataset contains side-channel measurements obtained from a masked AES 128 software implementation. [5] The countermeasure is based on RSM (*Rotate S-box Masking*).

The original `DPAv4.2` contains 80 000 traces subdivided into 16 groups of 5 000 traces. Each group is defined with a separate and fixed key. Each measurement has 1 704 046 samples. In this work, we conduct our analyses on the first 400 000 samples since the selected interval contains side-channel leakages from the two S-box output and masking bytes.

CHES CTF 2018 - `CHES_CTF` The `CHES_CTF` dataset contains power measurements from four different devices, which allows us to verify our analyses with the portability scenario. The implementation is protected with the first-order Boolean masking. However, the authors of the dataset provide no information about mask shares and source code. From each device, 10 000 measurements were collected. Each trace contains 650 000 samples and represents the power consumption of the full AES encryption. In this work, we conduct our analysis on the first 150 000 sample points as this interval includes the first AES encryption round. Again, we explore the leakages from the S-box output.

3 Related Work

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 of the recent studies consider trimmed versions of the traces strategically selected from where the leakage is located (i.e., a small interval that includes at least some of the main SNR peaks - but not necessarily the highest peaks - from two secret shares) [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], the 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 datasets and `DPAV4.2`. 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 adversary

[4] https://github.com/ANSSI-FR/ASCAD/tree/master/ATMEGA_AES_v1/ATM_AES_v1_variable_key
[5] http://www.dpacontest.org/v4/42_doc.php

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
NOPOI	No	No POI selection and pre-processing is required.	All available

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 long short-term memory (LSTM), attention, and convolution blocks. This may convey a possibly alarming message that attacking raw measurements requires many neural network layers and very complex architectures. As we show in this paper, we can recover a target key byte with a single attack trace without feature selection by using relatively small MLP and CNN models.

Feature selection in profiling attacks is also closely connected to fast security evaluations with the worst-case security assumptions. From related works, we still see that the question of whether MLPs and CNNs can efficiently implement optimal profiling models for security evaluations with the worst-case security assumptions is not fully answered. More importantly, realistic security evaluations may also face limitations for feature selection, especially when datasets are noisy and randomness from the implemented target cannot be read. In this case, it is crucial to understand if MLPs and CNNs can fill the existing gap between the worst-case security evaluations and black-box or online attacks. Note that these remarks were also recently discussed in [BDMS22]. Therefore, in this paper, we provide a detailed analysis of deep learning-based profiling models to investigate how feature selection impacts security evaluations. We verify the efficiency of deep learning-based profiling models when feature selection is not possible or not considered.

4 Feature Selection Scenarios

This section describes three specific scenarios to select points of interest for a deep learning-based profiling SCA. We start from the worst-case security assumptions, 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 weaker adversary that does not know mask shares. This last scenario assumes that the adversary has no information about implementation and that the complete measured intervals (e.g., representing the processing of the first AES round) are attacked. We summarize the three scenarios in Table 1.

The attack intervals for each evaluated dataset are listed in Table 2.

4.1 Refined Points Of Interest (RPOI)

The analysis of the worst-case security provides conditions for an evaluator to implement a fast security evaluation and identify if a device leaks information when facing the strongest possible adversary. Thus, within the RPOI scenario, our goal is to verify the real potential of neural networks as profiling models when the worst-case security assumptions are in place. We assume an adversary with access to the mask shares and with sufficient knowledge of the implementation details (e.g., source code) to select optimal points of interest. As we target the first-order protected AES implementations, the two secret shares representing the points of interest are $s_r = S_{box}(p_i \oplus k_i) \oplus r$ and r for all evaluated datasets.

Table 2: Selected intervals for each feature selection scenario. '-' denotes that we did not explore that specific setting.

Dataset	RPOI	OPOI	NOPOI	Total
ASCADf	up to 1 000 SNR peaks from NOPOI interval	[45 400, 46 100]	[0, 100 000]	100 000
ASCADr	up to 1 000 SNR peaks from NOPOI interval	[80 945, 82 345]	[0, 250 000]	250 000
DPAv4.2	up to 1 000 SNR peaks from NOPOI interval	[170 000, 174 000] + [206 000, 210 000]	[250 000, 400 000]	1 700 000
CHES CTF	-	[0, 10 000] + [120 000, 150 000]	[0, 150 000]	650 000

To set a reference level for optimality, we also implement Gaussian template attacks (GTA). The entire analysis flow for the RPOI scenario is illustrated in Figure 1. Assuming that GTA represents the strongest model, we investigate if MLPs and CNNs can also achieve the same profiling model quality. Refined points of interest are selected from the highest SNR peaks from the two known secret shares during profiling. To provide a more detailed analysis of the effect that the number of features/RPOI has on MLPs and CNNs, we conduct analyses from 10 up to 1 000 RPOI, where the step is based on a logarithmic scale (i.e., 10, 20, ..., 100, 200, ..., 1 000). To ensure that we always include enough points of interest from both secret shares, half of RPOI are selected from the highest SNR peaks obtained from $s_r = S_{box}(p_i \oplus k_i) \oplus r$ and the other half are selected from highest SNR peaks obtained from the secret mask share r. Note that the points of interest are always selected from raw side-channel measurements, and the maximum number of points is aligned with the related works (cf. 700 for `ASCADf` and 1 400 for `ASCADr`). For each number of RPOI, we also apply Linear Discriminant Analysis (LDA) for dimensionality reduction. The LDA provides five dimensions for each of the two shares, making a total of ten dimensions for GTA. We tested other values for dimensions and verified that the total of ten dimensions provided the best results. For neural networks, we proceed in two ways: first, a grid search is applied to MLP (with a maximum of three hidden layers) and CNN models (with a maximum of one convolution layer and two dense layers) after the dimensionality reduction with LDA. Second, to see if MLPs and CNNs can also reach optimal profiling models without dimensionality reduction, we train the models right after feature selection, where the model selection also comes from the grid hyperparameter search. Finally, from the best-found models, we compute guessing entropy and perceived information.

As the target datasets in this paper represent side-channel measurements from AES protected with the first-order Boolean masking countermeasure, we also emulate a hiding countermeasure by artificially adding different levels of Gaussian noise to the traces. The main goal is to check whether deep neural networks can also treat noise besides masking and still provide optimal results. This way, we divide the refined points of interests into three types: *low* (≈ 0.1), *medium* (≈ 1), and *high* (≈ 10) Signal-to-Noise Ratio RPOI levels.

Objective 1. *Verify if the refined number of points of interest with and without dimensionality reduction 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 that the adversary has access to random secret shares and implementation knowledge. The main difference from the 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.

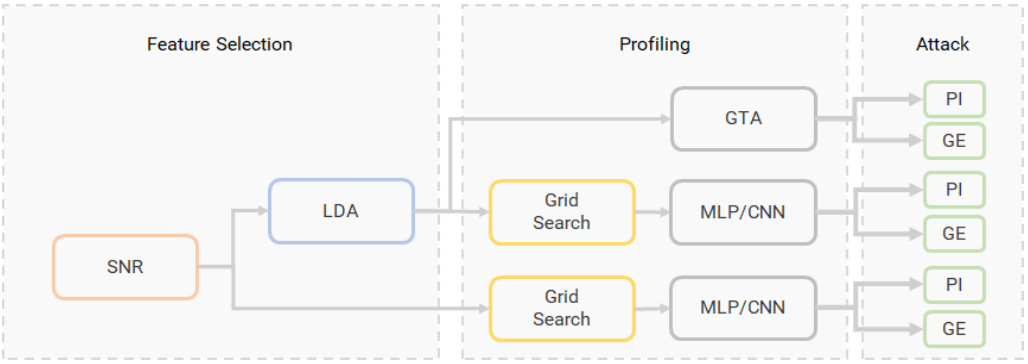


Figure 1: Flowchart for the RPOI scenario.

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, which contains some of the most leaking samples of two secret shares of the third key byte. 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 the `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., has low SNR values. A profiling model built from this interval should be, therefore, insensitive to noisy samples. Therefore, the main reason for defining OPOI besides the RPOI 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 intervals, including the main SNR peaks from the random secret shares, is beneficial for deep learning-based profiling attack performance.*

4.3 Non-optimized Points Of Interest (NOPOI)

To skip the feature selection process and to 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. The same principle is reinforced in [MBC+20], where the authors also attacked large trace intervals. Although results are competitive with state-of-the-art (they could 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, the 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 tens 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 counterintuitive from a practical perspective. The datasets evaluated in this paper (and also in [LZC+21]) have trace lengths of 100 000 (`ASCADf`), 250 000 (`ASCADr`), 1 7M (`DPAv42`), and 650 000 (`CHES_CTF`) samples. Coincidentally, [WAGP20] demonstrated the efficiency of deep learning architectures when the first layer is an *average-pooling* layer that, in the end, implements a window resampling of the input traces. The main idea from [WAGP20] was to keep the

same feature map dimensions as presented in [ZBHV19] in their CNN models, in which the first convolution layer was removed. Essentially, delivering this resampling task to the neural network is similar to performing trace resampling beforehand. Therefore, the proposed NOPOI scenario applies resampling to the input traces. Evidently, the resampling process may be advantageous or disadvantageous to the attack. Advantages appear when datasets contain high enough SNR leakages and when small misalignments still remain in the traces. Also, resampling results in smaller input neural network layers, which reduces training complexity and compensates for leakage dilution. Disadvantages will be evident when measurements contain low SNR leakages and are perfectly aligned.

We consider four different window sizes for resampling: 10, 20, 40, and 80. A window indicates the number of trace samples that are averaged into a single sample. Note that we always consider the overlap of 50% of window size in the resampling process. Obviously, in this case, a first average-pooling (as considered in [WAGP20]) is not necessary anymore. The resampling process is applied to the interval where the first AES encryption round is computed by the target device. This means that we do not perform the attack on fully raw trace intervals. For the ASCAD datasets, this step is not necessary, as the provided traces already represent only the first AES encryption round. For the `DPAv42` and `CHES CTF` datasets, we select the interval corresponding to the first AES encryption round, which is an obvious and intuitive process.[6] Note that in [LZC+21], the authors also trimmed the interval of `DPAv42` to the first encryption round. As shown in the results section, neural networks with up to only eight hidden layers are sufficient to achieve results that are better than state-of-the-art for some of the datasets. This suggests that the NOPOI scenario with trace resampling is an alternative solution to very deep models as proposed in [LZC+21].

Objective 3. *Verify if attacking the 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

The 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 as well as attacks on very large trace intervals) [LZC+21, WHJ+21].

The first approach requires more knowledge of the effect of hyperparameters on the learning process, and usually, several different configurations (which in a few cases include a grid search process as in [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 hyperparameter effects, even if optimized ranges are required for a more efficient hyperparameter 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 made by only looking at the model's performance for a few datasets.

[6] Attacking the full AES traces, which contain power consumption from all ten rounds, from `DPAv42` and `CHES CTF` datasets would be a waste of GPU power and memory resources.

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, we define the same ranges for hyperparameter search across all feature selection scenarios and datasets.

5.2 Hyperparameter Search

We follow the second approach listed in Section 5.1 for the model selection since we aim at 1) defining an algorithm selection process independent of the evaluated dataset and 2) verifying how a unique hyperparameter search process performs across multiple feature selection scenarios. We perform a random hyperparameter search for MLP and CNN models. Tables 3 and 4 list the covered search space for each hyperparameter. Random searches are applied to the OPOI and NOPOI feature selection scenarios. The grid search results in a small random search subspace and is applied to the RPOI case because finding optimal models is easier as we are already selecting the most leaking samples. Our random search space allows the selection of a deep neural network with up to eight hidden layers. As shown in the tables, the search space for random MLP search is close to 6 million, and for CNNs, the search space is higher than 1 billion options. For each specific number of points of interest, we search for 500 different models. This process is separately applied to MLP and CNN architectures for the Hamming weight and Identity leakage models.

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`. Additionally, we allow the search for different weight initialization options, which are `random`, `glorot`, and `he uniform`. Neural networks can also contain kernel regularization `l1` or `l2` and their possible values are randomly selected from $\{5e^{-3}, 1e^{-3}, 5e^{-4}, 1e^{-4}, 5e^{-5}, 1e^{-5}\}$. If dropout layers are included as a regularization method, the dropout rate is randomly selected between 0.05 and 0.5, with a step of 0.05.

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 are 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 used in CNNs is the batch normalization layer after each pooling layer.

Table 3: Hyperparameter search options and ranges for MLPs. The number of epochs is set to 100.

Hyperparameter	Random Search	Grid Search
Optimizer	`Adam`, `RMSprop`	`Adam`, `RMSprop`
Dense Layers	1, 2, 3, 4, 5, 6, 7, 8	1, 2, 3
Neurons	10, 20, 50, 100, 200, 300, 400, 500	20, 50, 100, 200
Activation Function	`SeLU`, `ReLU`	`SeLU`, `ReLU`
Learning Rate	1e-3, 5e-3, 1e-4, 5e-4	1e-3, 1e-4
Batch Size	100 to 1 000 (step: 100)	200, 400
Weight Initialization	`random`, `glorot` or `he uniform`	`glorot`
Regularization	`None`, `Dropout`, `l1` or `l2`	`None`
Total Search Space	**5 971 968**	**192**

Table 4: Hyperparameter search options and ranges for CNNs. The number of epochs is set to 100.

Hyperparameter	Random Search	Grid Search
Optimizer	`Adam`, `RMSprop`	`Adam`, `RMSprop`
Convolution Layers	1, 2, 3, 4	1
Convolution Filters	$4 * 2^{i-1}$, $8 * 2^{i-1}$, $12 * 2^{i-1}$, $16 * 2^{i-1}$ (i = conv. layer index)	5, 10
Convolution Kernel	26 to 52 with a step of 2	2, 4
Convolution Stride	Convolution Kernel / 2	1
Pooling Type	`maxpooling`, `avgpooling`	`avgpooling`
Pooling Size	2, 4, 6, 8, 10	2
Pooling Stride	Pooling Size	2
Dense Layers	1, 2, 3, 4	1, 2
Neurons	10, 20, 50, 100, 200, 300, 400, 500	50, 100
Activation Function	`SeLU`, `ReLU`	`SeLU`, `ReLU`
Learning Rate	1e-3, 5e-3, 1e-4, 5e-4	1e-3, 1e-4
Batch-Size	100 to 1 000 (step: 100)	200, 400
Weight Initialization	`random`, `glorot` or `he uniform`	`glorot`
Regularization	`None`, `Dropout`, `l1` or `l2`	`None`
Total Search Space	**>1B**	**128**

6 Results

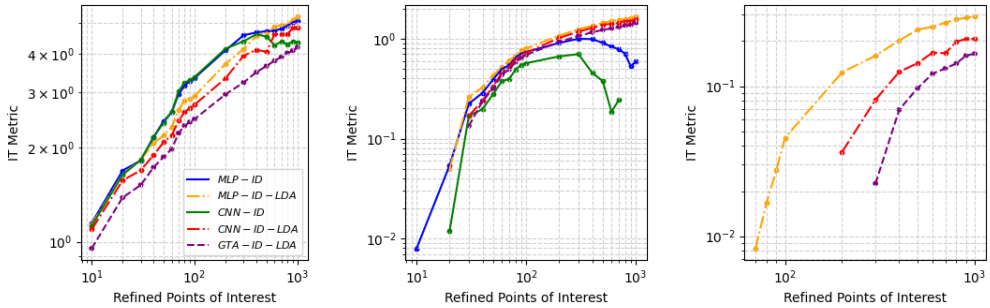
The results presented in this section were obtained from a high-performance computing environment (HPC), and our analysis was deployed on a maximum of four NVIDIA GeForce GTX 2080Ti GPUs. Neural network training always trains on one GPU, allowing us to deploy four experiments in parallel. Depending on the dataset and model size, a single neural network training time may range from a few seconds to approximately one hour. The dataset and model size also impact RAM memory usage, ranging from hundreds of MB up to a hundred GB. The framework is implemented in Python 3.7, and we use TensorFlow 2.0 and Keras 2.1.6. Note that the legend includes lines for the number of attack traces to reach GE of 1 for both the HW and ID leakage models. When the attack does not work (i.e., cannot break target with 3 000 attack traces), the line is not visible, but we keep the legend to indicate that the attack does not work. Additionally, when the attack works perfectly (i.e., we break the target with a single attack trace), the lines are not visible, but the attack performance can be seen in the legend.

6.1 ASCADf

RPOI. In this section, we directly apply the workflow depicted in Figure 1 for the `ASCADf` dataset with the RPOI feature selection scenario and different SNR levels. Figures 2 and 4 show the perceived information results obtained from MLPs, CNNs, and Gaussian template attacks (GTA). Results are provided with the selection of different numbers of refined points of interest. For neural networks, we provide results with and without LDA for dimensionality reduction. The corresponding estimated number of attack traces, $\hat{N}_{\beta,PI}$, are shown in Figures 3 and 5. In the particular case of high SNR, neural networks show superior performance to GTA, especially for the Hamming weight leakage model. Furthermore, we can see that the more features, the better the model performance, as more leaky points of interest are included in the process.

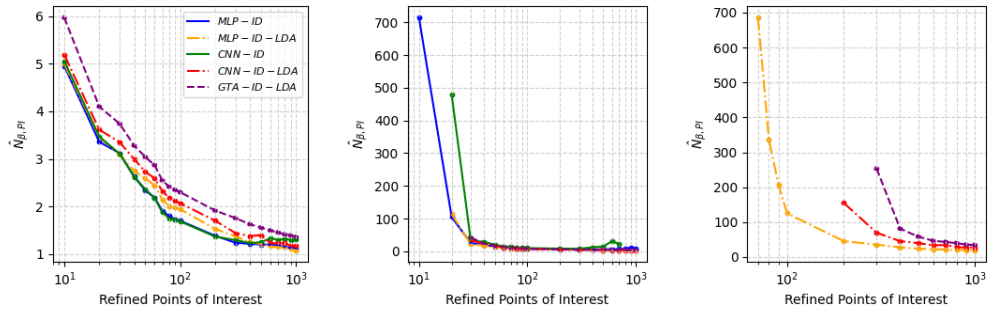
As the `ASCADf` dataset has a "small" number of profiling traces (50 000), it is expected that medium and low SNR cases show sub-optimal performance. For the dataset with medium SNR (Figures 2b and 4b), LDA became necessary to achieve higher PI values as the number of features increases. When LDA is not considered for MLP and CNN models, increasing the number of selected RPOIs tends to reduce the quality of the model. This indicates that for this particular case, dimensionality reduction is necessary to find optimal models.

For the low SNR case, as shown in Figures 2c and 4c for the Identity and Hamming



(a) High SNR (≈ 10). **(b)** Medium SNR (≈ 1). **(c)** Low SNR (≈ 0.1).

Figure 2: Perceived information (PI) computed from profiling attacks (MLP, CNN, and GTA) implemented with different numbers of Refined Points of Interest (RPOI). The figures show results for the `ASCADf` dataset with the Identity leakage model.



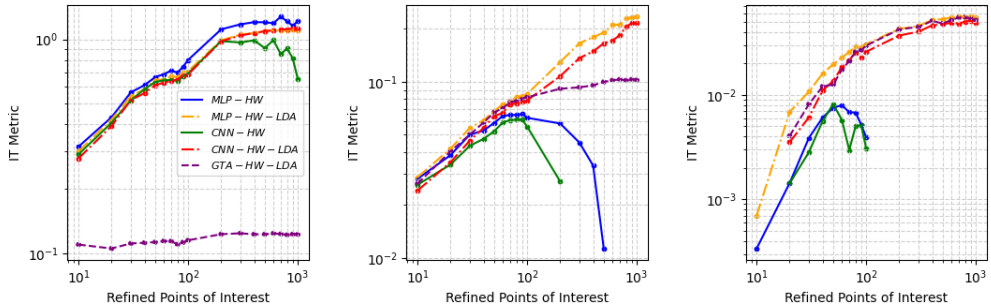
(a) High SNR (≈ 10). **(b)** Medium SNR (≈ 1). **(c)** Low SNR (≈ 0.1).

Figure 3: Number of required attack traces to reach GE equal to 1 from profiling attacks (MLP, CNN, and GTA) implemented with different numbers of Refined Points of Interest (RPOI). The figures show results for the `ASCADf` dataset with the Identity leakage model.

weight leakage models, respectively, the importance of LDA is even more visible. For the Identity leakage model case, MLP with LDA shows better performance compared to CNNs and GTA. In this case, excluding LDA from the process delivered no profiling models where the PI values are positive. For the Hamming weight leakage model, skipping LDA allowed us to find models with positive PI only if the number of RPOIs is lower than 100. In terms of computation time efforts, the grid search process when considering 1 000 points of interest takes approximately 11 hours to complete with a single GPU.

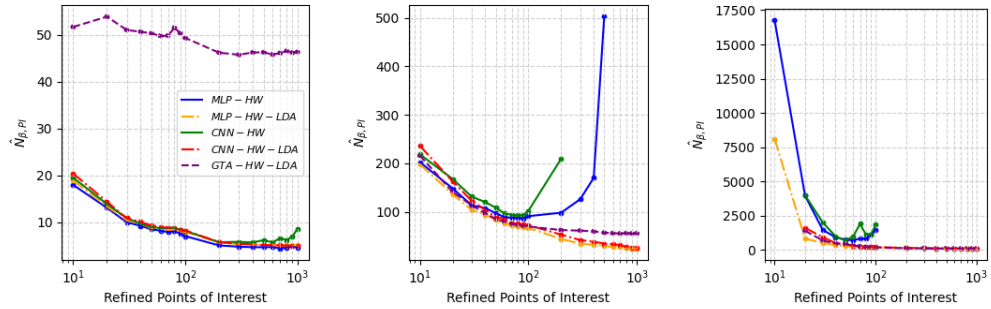
To conclude, under the worst-case security assumptions, we can find neural networks with superior performance compared to GTA. Obviously, this requires a costly hyperparameter search. Therefore, there is a trade-off between model performance and computation cost.

OPOI. The results for the OPOI scenario are commonly reported in the literature with different deep neural network configurations [BPS+20, ZBHV19, WAGP20, PCP20]. We apply the random hyperparameter search process where the number of searches is limited to 500 per leakage model. The guessing entropy results for best-found models with the OPOI scenario are shown in Figure 6. Results are aligned with state-of-the-art, especially for the Identity leakage model, which indicates that our hyperparameter search space is set to model performance similar to alternative hyperparameter tuning techniques (e.g.,



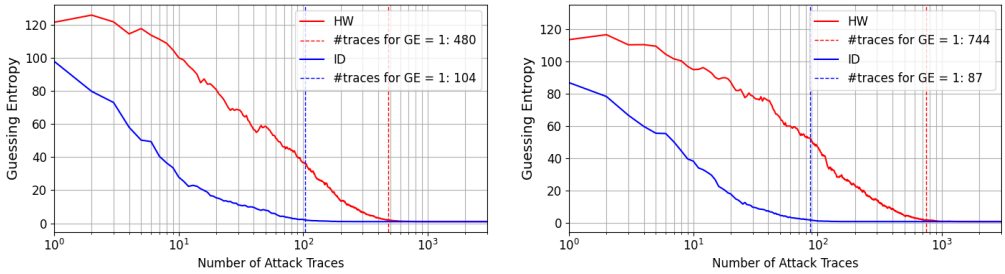
(a) High SNR (≈ 10). **(b)** Medium SNR (≈ 1). **(c)** Low SNR (≈ 0.1).

Figure 4: Perceived information (PI) computed from profiling attacks (MLP, CNN, and GTA) implemented with different numbers of Refined Points of Interest (RPOI). The figures show results for the `ASCADf` dataset with the Hamming weight leakage model.



(a) High SNR (≈ 10). **(b)** Medium SNR (≈ 1). **(c)** Low SNR (≈ 0.1).

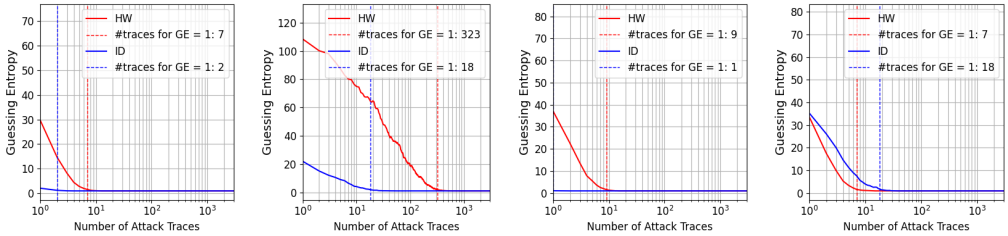
Figure 5: Number of required attack traces to reach GE equal to 1 from profiling attacks (MLP, CNN, and GTA) implemented with different numbers of Refined Points of Interest (RPOI). The figures show results for the `ASCADf` dataset with the Hamming weight leakage model.



(a) MLP. **(b)** CNN.

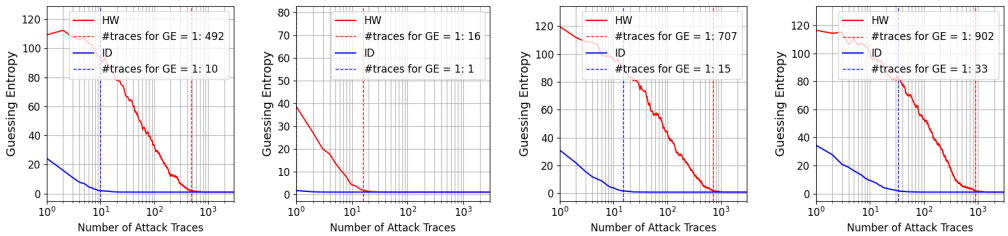
Figure 6: Optimized points of interest (OPOI): best guessing entropy results for the `ASCADf` dataset with different leakage models.

reinforcement learning and grid search).



(a) MLP (w=80). **(b)** CNN (w=80). **(c)** MLP (w=40). **(d)** CNN (w=40).

Figure 7: Non-optimized points of interest (NOPOI): best guessing entropy results for the `ASCADf` dataset with different leakage models and resampling windows of 80 and 40 samples.



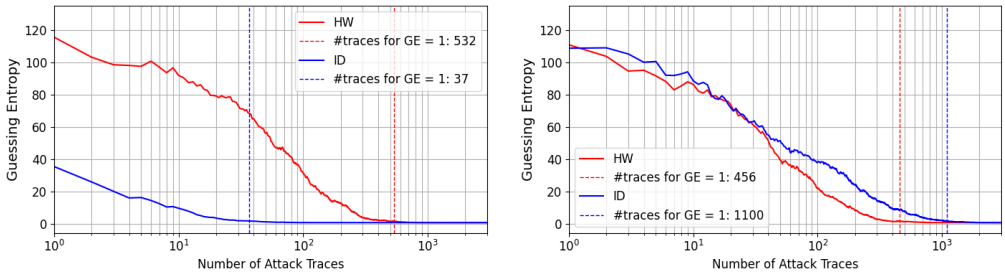
(a) MLP (w=20). **(b)** CNN (w=20). **(c)** MLP (w=10). **(d)** CNN (w=10).

Figure 8: Non-optimized points of interest (NOPOI): best guessing entropy results for the `ASCADf` dataset with different leakage models and resampling windows of 20 and 10 samples.

NOPOI. Again, in the NOPOI scenario, we deploy a random hyperparameter search. As specified in Section 4.3, we apply trace resampling to the raw traces with different window sizes. Figure 7 shows the performance of the best-found MLP and CNN models when windows of 80 and 40 are considered for resampling. In this case, the original trace lengths that consist of 100 000 samples are reduced to 1 250 and 2 500 samples, respectively. Figure 8 shows results for the best found MLP and CNN models when resampling windows of 20 and 10 are considered, which results in trace lengths of 10 000 and 20 000, respectively. As we can see in Figures 7c and 8b, the best models with the Identity leakage model could successfully recover the correct key byte after processing a single attack trace. Results with a resampling window of 10, as shown in Figures 8c and 8d, show slightly worse results, indicating that reducing the number of features with resampling is beneficial for the considered small models.

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

In [LZC+21], the authors could defeat the same `ASCADf` dataset with six attack traces, which from a practical perspective, is similar to our findings. Obviously, to find out top-performing models, a long hyperparameter tuning process is required. For the MLP with the Identity leakage model, where the best results were found, our full random search process for 500 models took 33 hours to complete with a single GPU. At the same time, the best architecture from [LZC+21] for the `ASCADf` dataset required approximately 24 hours to be trained, and information about hyperparameter tuning is not provided in the paper.



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

Figure 9: 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. Next, we also evaluate the possibility of key recovery with the NOPOI scenario for `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. Afterward, we again applied trace resampling, but this time only for a window size of 20, as this was the best resampling window found for CNN models in the previous section.

Figure 9 shows results for `ASCADf` with desynchronized side-channel traces. In Figure 9a, the results were obtained when we applied data augmentation to the training process by randomly shifting the training traces to the left or to the right by up to 10 samples. We again train the model for 100 epochs, and for each epoch, 500 artificial measurements are generated, which also results in 50 000 profiling traces as the original dataset. 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 a 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. Note that we consider the same CNN hyperparameter ranges as in the previous experiments, indicating that trace desynchronization can be bypassed without necessarily increasing the hyperparameter search complexity.

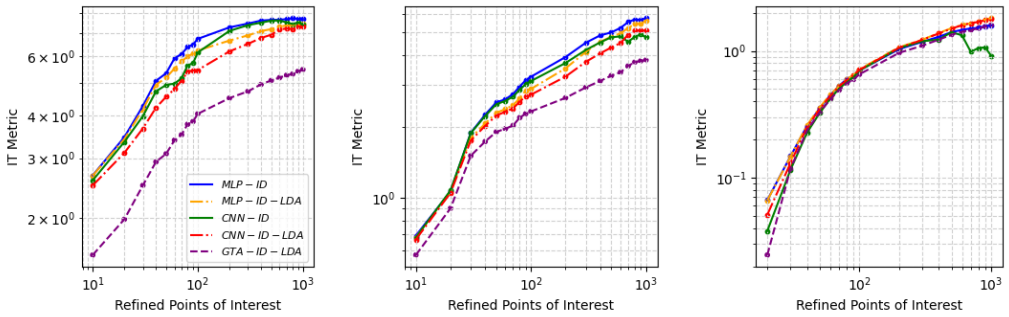
Attacking the Full AES Key with the NOPOI Scenario. In this section, we retrain the best-found models with the NOPOI scenario (without desynchronization) for the full AES key. We apply this process for MLP and CNN, including the Hamming weight and Identity leakage models. The best model is always initialized with the same weights as the best model found in the random hyperparameter search process. Table 5 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 the `ASCADf` dataset is recovered with less than 20 attack traces. For the best CNN models, we are unable to recover the full key with the same model trained on key byte 2, even if we can recover several key bytes with less than ten attack traces. Note, however, that the two first key bytes are unprotected.

6.2 ASCADr

RPOI. Figure 10 shows the PI values for different numbers of refined points of interest when the Identity leakage model is considered. In this case, the dataset contains more profiling traces (i.e., 200 000), and we observe that more features are beneficial for all GTA, MLP, and CNN models, even without LDA. We can mainly conclude here that dimensionality reduction becomes irrelevant for MLP and CNN models, and they also

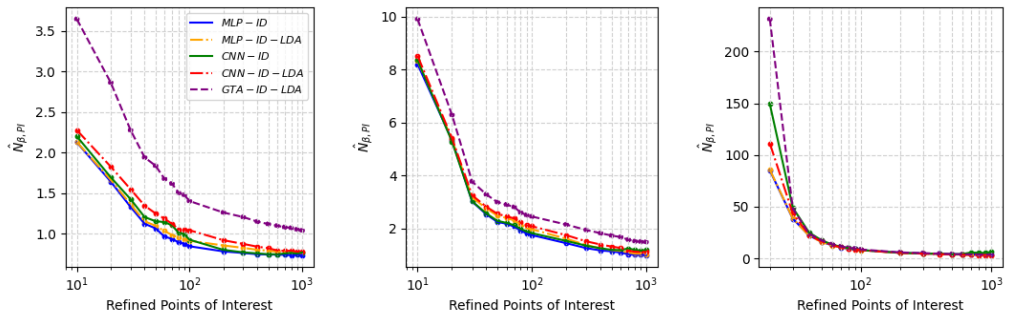
Table 5: Minimum number of attack traces to obtain guessing entropy equal to 1 with the `ASCADf` dataset for all key bytes (results provided by MLP and CNN, for the Hamming weight and Identity leakage models).

Model Type	Leakage Model	Required Attack Traces (per key byte)
CNN	$Sbox(p_i \oplus k_i)$	1, 1, 1, >3 000, 2, >3 000, >3 000, >3 000, >3 000.0, >3 000, >3 000, 1, >3 000, >3 000, 3, >3 000
CNN	$HW(Sbox(p_i \oplus k_i))$	3, 3, 14, 6, 8, 7, >3 000, 8, >3 000, 7, >3 000, 11, >3 000, 40, 10, >3 000
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, >3 000, 20, 7, 7



(a) High SNR (≈ 10). **(b)** Medium SNR (≈ 1). **(c)** Low SNR (≈ 0.1).

Figure 10: Perceived information (PI) computed from profiling attacks (MLP, CNN, and GTA) implemented with different numbers of Refined Points of Interest (RPOI). The figures show results for the `ASCADr` dataset with the Identity leakage model.

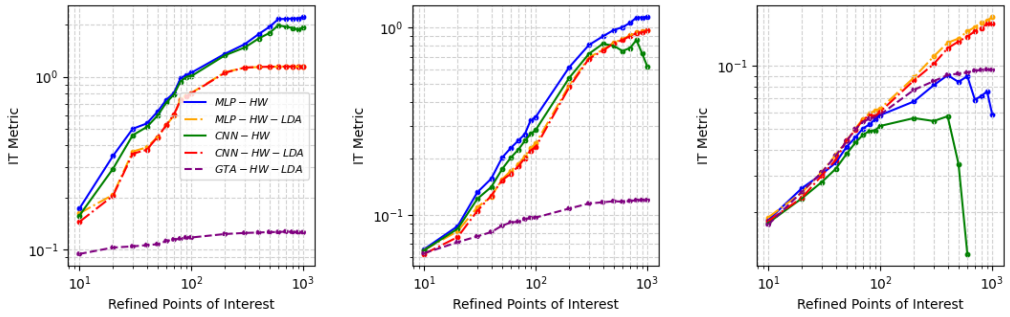


(a) High SNR (≈ 10). **(b)** Medium SNR (≈ 1). **(c)** Low SNR (≈ 0.1).

Figure 11: Number of required attack traces to reach GE equal to 1 from profiling attacks (MLP, CNN, and GTA) implemented with different numbers of Refined Points of Interest (RPOI). The figures show results for the `ASCADr` dataset with the Identity leakage model.

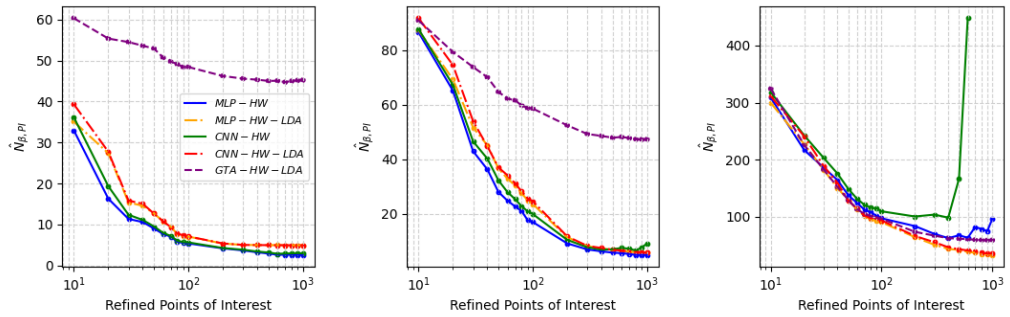
provide superior results compared to GTA. Interestingly, for the low SNR case, all profiling models perform in a very similar way, indicating that neural networks can also deal with higher amounts of noise. One possible reason for this is that with more noise, there are fewer models that fit well, so more models behave similarly, i.e., sub-optimally. Figure 11 shows the minimum number of estimated attack traces, $\hat{N}_{\beta,PI}$, to reach success rate of 80%.

The results obtained for the Hamming weight leakage model are shown in Figure 12. For the high and medium SNR cases, the performance of MLP and CNN models is far superior to GTA, and the more features, the better. Moreover, we can see that excluding LDA from the process provides even better performance. As the `ASCADr` dataset has



(a) High SNR (≈ 10). **(b)** Medium SNR (≈ 1). **(c)** Low SNR (≈ 0.1).

Figure 12: Perceived information (PI) computed from profiling attacks (MLP, CNN, and GTA) implemented with different numbers of Refined Points of Interest (RPOI). The figures show results for the `ASCADr` dataset with the Hamming weight leakage model.

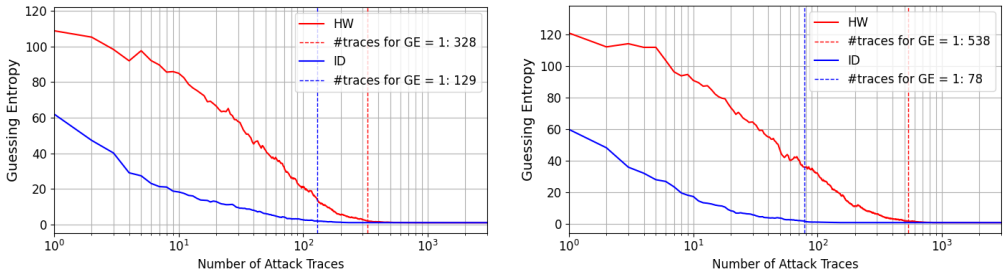


(a) High SNR (≈ 10). **(b)** Medium SNR (≈ 1). **(c)** Low SNR (≈ 0.1).

Figure 13: Number of required attack traces to reach GE equal to 1 from profiling attacks (MLP, CNN, and GTA) implemented with different numbers of Refined Points of Interest (RPOI). The figures show results for the `ASCADr` dataset with the Hamming weight leakage model.

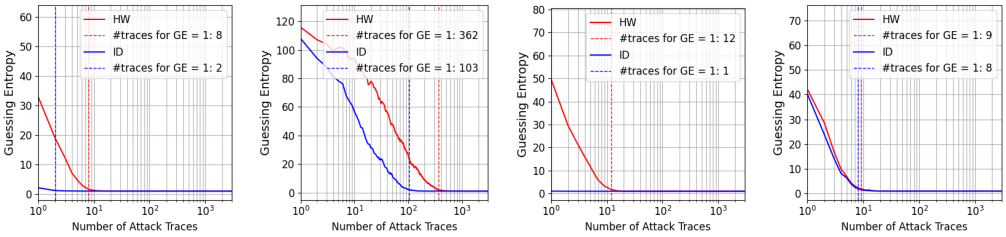
more profiling traces compared to the `ASCADf` dataset, neural networks can provide better performance without dimensionality reduction if noise levels are lower. Indeed, when noise is not the limiting factor to building well-performing neural networks, more features can help as they provide more information for the neural networks (and those features that are not useful will be discarded due to the implicit feature selection). Stated differently, less noise allows neural networks to recognize important features more easily and discard the irrelevant ones, which is well-known in the ML community [SSBD14, AB09]. For low SNR, as the dataset becomes noisier, we see a degradation in the performance of neural networks for more features when LDA is not taken into account. In this case, and for increased numbers of refined points of interest, LDA becomes crucial to achieving positive PI values. The reflected performance in terms of the estimated number of attack traces to achieve a success rate of 80% is shown in Figure 5. A grid search process for `ASCADr` dataset, when considering 1 000 points of interest, takes approximately 23 hours to complete with two parallel GPUs.

OPOI. Figure 14 shows the guessing entropy for the best models for the `ASCADr` dataset with the OPOI feature selection scenario. The results obtained are aligned with state-of-



(a) MLP. **(b)** CNN.

Figure 14: Optimized points of interest (OPOI): best guessing entropy results for the `ASCADr` dataset with different leakage models.



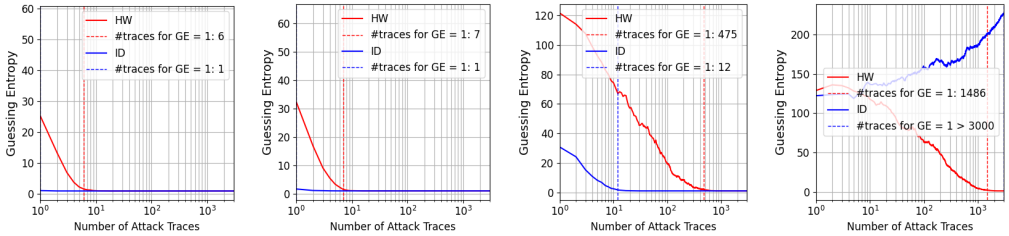
(a) MLP (w=80). **(b)** CNN (w=80). **(c)** MLP (w=40). **(d)** CNN (w=40).

Figure 15: Non-optimized points of interest (NOPOI): best guessing entropy results for the `ASCADr` dataset with different leakage models and resampling windows of 80 and 40 samples.

the-art results [RWPP21] for the `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 with MLP with up to two hidden layers and CNN with up to 4 hidden layers.

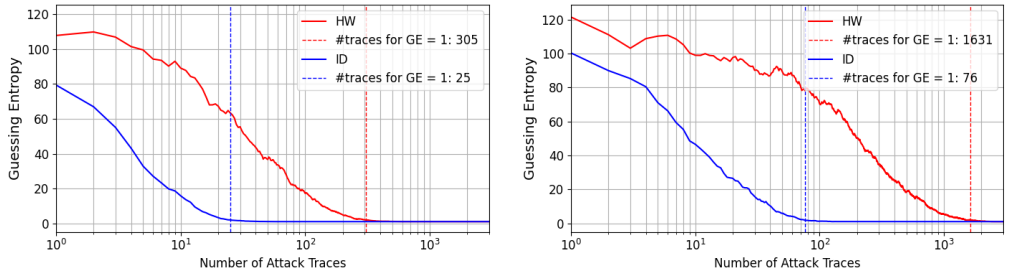
NOPOI. Here, we consider the `ASCADr` dataset with 250 000 sample traces. We also apply the resampling process with four different window options. Figure 15 shows guessing entropy for the best-found models from the random hyperparameter search for the MLP and CNN cases, with both the Hamming weight and Identity leakage models. This figure shows results when raw traces that represent the power consumption of the first AES encryption round are resampled with windows of 80 and 40 samples. For the Identity leakage model, we can find the best neural network model that recovers the key with a single attack trace when a resampling window of 40 samples is adopted and two attack traces when a resampling window of 80 samples is considered. Surprisingly, for the MLP case with the Identity leakage model, the best model contains a single hidden layer, and this model can successfully recover the target key byte with only eight attack traces. The best CNN could succeed with a single trace by having one convolution layer and three dense layers.

In Figure 16, we provide the guessing entropy results for the best models when resampling windows of 20 and 10 are applied to the raw traces. Note that for the resampling window of 20, the best found MLP and CNN models can recover the target key byte with a single attack trace if the Identity leakage model is considered. When a resampling window of 10 is applied, the best MLP found model recovers the key byte with 12 attack traces, as shown in Figure 16c. Note that a smaller resampling window delivers



(a) MLP (w=20). **(b)** CNN (w=20). **(c)** MLP (w=10). **(d)** CNN (w=10).

Figure 16: Non-optimized points of interest (NOPOI): best guessing entropy results for the `ASCADr` dataset with different leakage models and resampling windows of 80 and 40 samples.



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

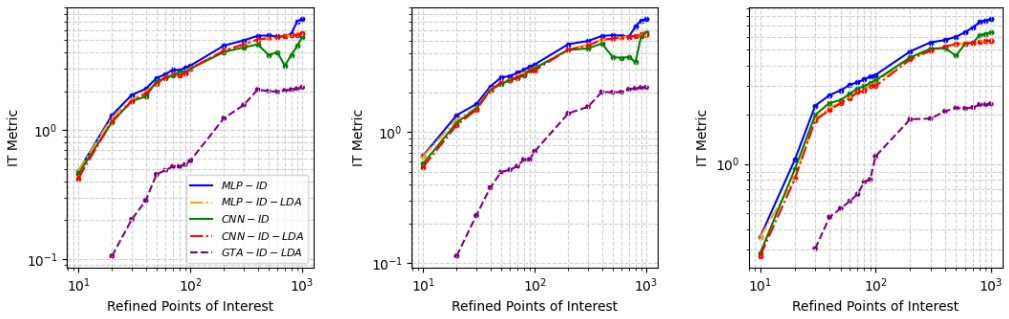
Figure 17: Non-optimized points of interest (NOPOI) with desynchronization: best guessing entropy results for the `ASCADr` dataset with and without data augmentation.

inferior profiling models with the current hyperparameter search. This indicates that when attacking the full trace intervals, the resampling window can play an important role if we want to keep profiling models smaller. A resampling window of 10 results in trace lengths of 50 000 samples for the `ASCADr` dataset. To reach optimal profiling models for larger resampling windows, a larger random hyperparameter search would be necessary.

NOPOI with Trace Desynchronization. The desynchronization is artificially added to the `ASCADr` dataset by shifting each raw 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 is resampled into 25 000 samples by using a resampling window of 20 with a window overlap of 50% as this window provided the best CNN results in the NOPOI scenario. Results with the desynchronized `ASCADr` dataset are shown in Figure 17. We apply a random hyperparameter search with the same hyperparameter range from the previous experiments. Results from Figure 17a show the 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. We again train each searched model for 100 epochs, and for each epoch, we generate 2 000 synthetic traces. As we can see, the correct key byte can be recovered after 25 attack traces with the Identity leakage model. Results in Figure 17b 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 but with significantly more attack traces. The usage of data augmentation again provides better results.

Table 6: 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, >5 000, 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



(a) High SNR (≈ 10). **(b)** Medium SNR (≈ 1). **(c)** Low SNR (≈ 0.1).

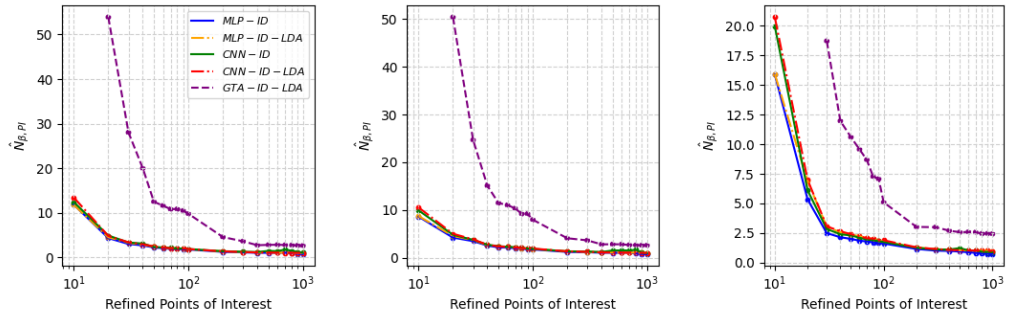
Figure 18: Perceived information (PI) computed from profiling attacks (MLP, CNN, and GTA) implemented with different numbers of Refined Points of Interest (RPOI). The figures show results for the `DPAv4.2` dataset with the Identity leakage model.

Attacking the Full AES Key with the NOPOI Scenario. Like for the `ASCADf` dataset, 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 6 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 result so far on the `ASCADr` dataset in the literature.* The attack presented in [LZC+21] for the NOPOI-like scenario on `ASCADr` was able to recover the key with 8 attack traces. Again, from a practical perspective, both results are similar. To achieve our best model, the 500 models random search process took approximately 2,5 days by running the analysis on four parallel GPUs. In the case of [LZC+21], the hyperparameter tuning efforts are not mentioned, and a single model training on `ASCADr` took 55 minutes.

6.3 DPAv4.2

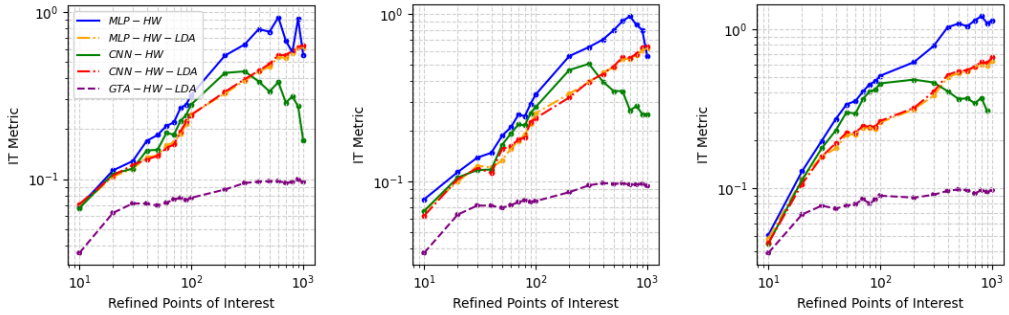
RPOI Results for the `DPAv4.2` dataset and the Identity leakage model, with the RPOI feature selection scenario, are shown in Figures 18 and 19. MLP and CNN models tend to provide better results when increasing the number of refined points of interest. This observation is also valid for the Gaussian template attack results, even if their performance is inferior when compared to neural networks with and without LDA for dimensionality reduction. Moreover, for the medium and low SNR cases, the performance of profiling models shows no degradation in the perceived information and corresponding estimated number of required attack traces to reach a success rate of 80%.

Results for the Hamming weight leakage model are shown in Figures 20 and 21 for



(a) High SNR (≈ 10). **(b)** Medium SNR (≈ 1). **(c)** Low SNR (≈ 0.1).

Figure 19: Number of required attack traces to reach GE equal to 1 from profiling attacks (MLP, CNN, and GTA) implemented with different numbers of Refined Points of Interest (RPOI). The figures show results for the `DPAv4.2` dataset with the Identity leakage model.



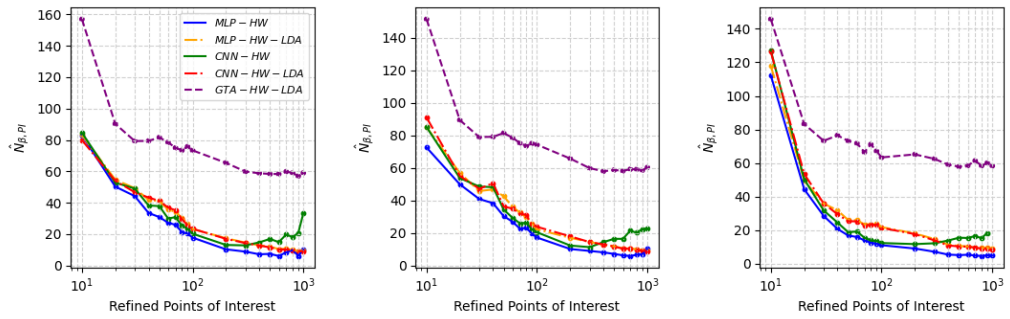
(a) High SNR (≈ 10). **(b)** Medium SNR (≈ 1). **(c)** Low SNR (≈ 0.1).

Figure 20: Perceived information (PI) computed from profiling attacks (MLP, CNN, and GTA) implemented with different numbers of Refined Points of Interest (RPOI). The figures show results for the `DPAv4.2` dataset with the Hamming weight leakage model.

perceived information and required number of attack traces. Again, we see that neural networks, with and without LDA preprocessing, show superior results compared to Gaussian template attacks, regardless of the SNR levels. The total amount of time to grid search on 1 000 points of interest took approximately 13 hours with a single GPU.

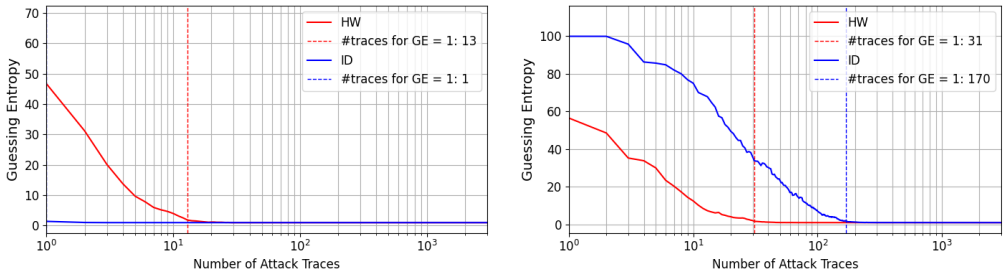
OPOI As the OPOI scenario assumes that an adversary can locate points of interest by knowing the random secret shares, we concatenated two continuous intervals of 400 samples into a final attacked interval of 800 samples. Each of the 400 sample intervals contain the leakage of one of the secret shares, i.e., $s_r = Sbox(p_0 \oplus k_0) \oplus r_0$ and r_0. The results obtained from the random hyperparameter search with the best found MLP indicate that we can recover the correct key byte with a single attack trace if the Identity leakage model is considered. With the Hamming weight leakage model, the best model retrieves the key with 13 attack traces. For CNNs, we can retrieve the correct key with 31 and 179 attack traces for the Hamming weight and Identity leakage model, respectively. Therefore, we demonstrate that optimal models can be found with the OPOI scenario for this dataset.

NOPOI For this scenario, we select the trace interval that clearly contains the leakages from the first AES encryption round, as specified in Table 2. Thus, we also apply four



(a) High SNR (≈ 10). **(b)** Medium SNR (≈ 1). **(c)** Low SNR (≈ 0.1).

Figure 21: Number of required attack traces to reach GE equal to 1 from profiling attacks (MLP, CNN, and GTA) implemented with different numbers of Refined Points of Interest (RPOI). The figures show results for the `DPAv4.2` dataset with the Hamming weight leakage model.

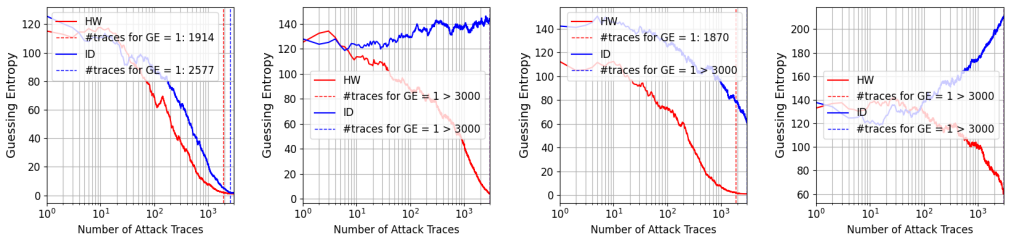


(a) MLP. **(b)** CNN

Figure 22: Refined points of interest (OPOI): best guessing entropy results for the `DPAv4.2` dataset with different leakage models.

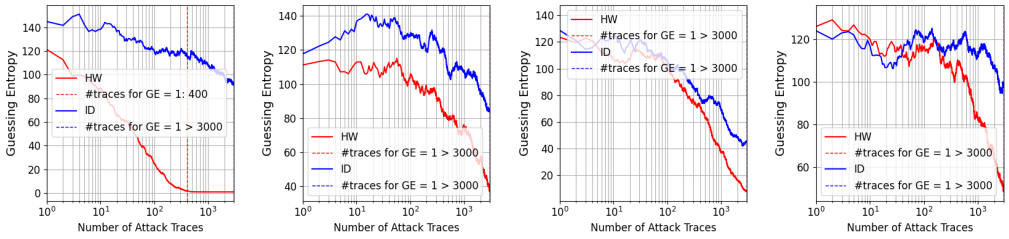
different resampling windows of 10, 20, 40, and 80 samples. Results for the best MLP and CNN models found when resampling windows of 80 and 40 are considered are shown in Figure 23. MLP and the Hamming weight leakage model combination tends to provide better results when compared to CNNs and the Identity leakage model cases. In this case, we cannot find optimal profiling models that recover the correct key byte with tens of attack traces. However, the same random hyperparameter search process delivers successful profiling attacks, which is an advantage from the security evaluation perspective because we do not necessarily need to go for more complex hyperparameter search spaces for this dataset when masks are unknown.

Figure 24 shows the best MLP and CNN models when resampling windows of 20 and 10 are considered. This time, the best MLP with the Hamming leakage model can retrieve the key with only 400 attack traces. Results reported in [LZC+21] for this same dataset provides a model that recovers the correct key byte with only 8 attack traces. Note, however, that their best model was fine-tuned for this dataset, and the hyperparameter efforts to generate this model are omitted from the paper, which makes the direct comparison more difficult. Nevertheless, in terms of computation time, the architecture from [LZC+21] requires approximately 14 hours to be trained. Our 500 model hyperparameter search process took in total 42 hours to complete. There is a large difference in performance between both best models. Still, we use the same hyperparameter search process for all datasets, which simplifies the process and indicates that relatively small models can deliver



(a) MLP (w=80). **(b)** CNN (w=80). **(c)** MLP (w=40). **(d)** CNN (w=40).

Figure 23: Non-optimized points of interest (NOPOI): best guessing entropy results for the `DPAV4.2` dataset with different leakage models and resampling windows of 80 and 40 samples.



(a) MLP (w=20). **(b)** CNN (w=20). **(c)** MLP (w=10). **(d)** CNN (w=10).

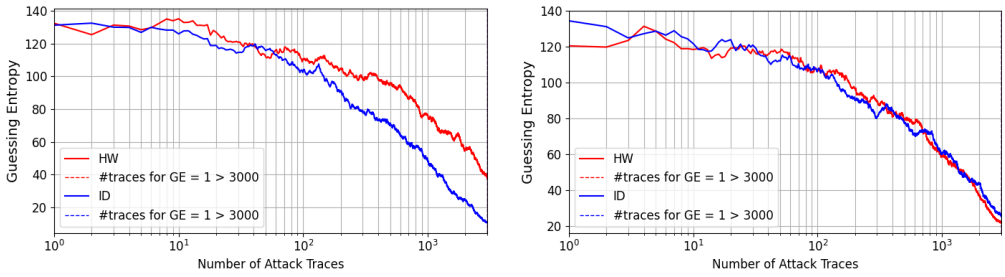
Figure 24: Non-optimized points of interest (NOPOI): best guessing entropy results for the `DPAV4.2` dataset with different leakage models and resampling windows of 20 and 10 samples.

successful results if an extensive hyperparameter tuning process is considered.

NOPOI with Trace Desynchronization The best CNN model found with the NOPOI scenario for the `DPAV4.2` dataset considered window resampling of 80. Thus, we consider this setting to verify if the same random hyperparameter search process can recover the key when trace desynchronization is applied. Results are shown in Figure 25. As we can see, with and without trace augmentation, our best CNN models are unable to reach guessing entropy equal to 1 after processing 3 000 attack traces. However, we see a clear decreasing trend in guessing entropy, which indicates that the key recovery would be possible by adding more attack traces. The lower guessing entropy is obtained with data augmentation with the Identity leakage model, where 700 synthetic traces are generated for each epoch. Data augmentation is again a process where we randomly shift traces to the right and to the left during training by 10 samples.

Attacking the Full AES Key with the NOPOI Scenario We select the best MLP and CNN models from the NOPOI scenario that were obtained by attacking a single key byte, and we attack the rest of the AES key bytes. Models are retrained and initialized with the same weights for all key bytes. When considering MLP and the Hamming weight leakage model, the best MLP for the NOPOI case could recover the full AES key. When the Identity leakage model is applied to MLP, we recover 2 out 16 key bytes. The best found CNN models are unable to recover the AES key bytes with less than 3 000 attack traces.

Results for the desynchronization case or full key recovery are not provided for NOPOI-like scenarios in related works, which limits our discussion in terms of performance comparisons.



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

Figure 25: Non-optimized points of interest (NOPOI) with desynchronization: best guessing entropy results for the `DPAV4.2` dataset with different leakage models with and without data augmentation.

Table 7: Minimum number of attack traces to get guessing entropy equal to 1 with the `DPAV4.2` dataset for all key bytes (results provided by MLP and CNN, for the Hamming weight and Identity leakage models).

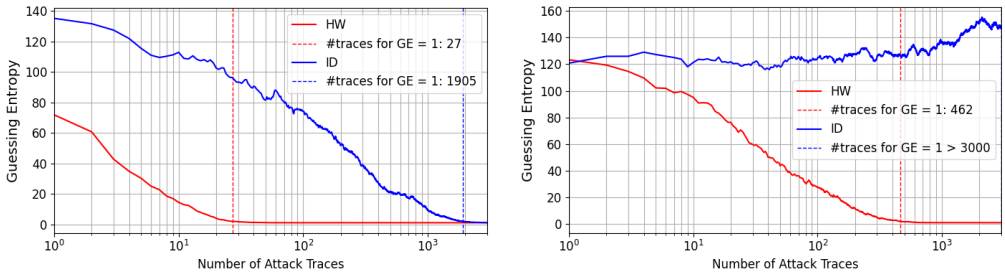
Model Type	Leakage Model	Required Attack Traces (per key byte)
CNN	$Sbox(p_i \oplus k_i)$	>3 000 (all key bytes)
CNN	$HW(Sbox(p_i \oplus k_i))$	>3 000 (all key bytes)
MLP	$Sbox(p_i \oplus k_i)$	2 427, 3 000, 146, 3 000, 3 000, 3 000, 3 000, 3 000, 3 000, 3 000, 3 000, 3 000, 3 000, 3 000, 3 000, 3 000
MLP	$HW(Sbox(p_i \oplus k_i))$	477, 759, 292, 1 666, 1 958, 346, 593, 592, 765, 536, 2 373, 702, 430, 299, 1 209, 2 854

6.4 CHES_CTF: A Portability Case

RPOI The CHES CTF 2018 dataset was released without information on mask shares used in the Boolean masking protection. Therefore, we do not provide RPOI analysis for this dataset as the refined selection of points of interest based on secret mask shares is not possible.

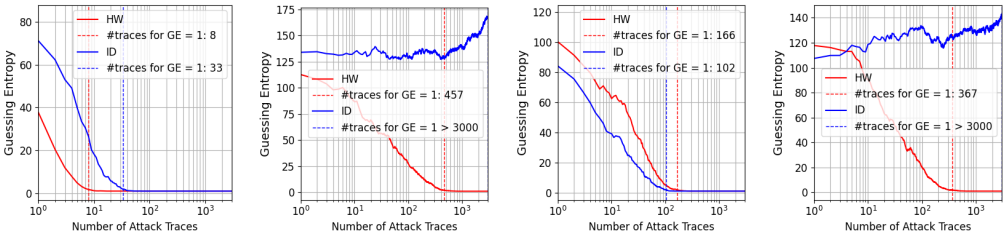
OPOI For the OPOI scenario, we select trace intervals as indicated in Table 2. As mask shares are not known, to select trimmed intervals, we observe input gradient peaks from models that succeed with key recovery. Of course, the interval is not optimized as for the ASCAD datasets. However, we obtain successful attack results. Note that we also apply the resampling process with a window of 20. Figure 26 shows results for the OPOI feature selection scenario for MLP and CNN models. Results are provided for the Identity and Hamming leakage models. The `CHES_CTF` dataset shows better results for the Hamming weight leakage model for this particular scenario. For the case of CNN with the Identity leakage model, we were unable to find a model through hyperparameter search that recovers the correct key in the attack phase. For the case of MLP, results are much better, especially for the Hamming weight, where only 27 attack traces were necessary to recover the correct key byte.

NOPOI We also attack the `CHES_CTF` dataset by skipping feature selection. When using MLP as a profiling model, both the Identity and Hamming weight leakage models allowed us to find a well-performing model that recovers the correct key byte with 18 and 53 attack traces, respectively. These results are shown in Figure 28a. For this particular case, the 500 models random search took approximately 69 hours to complete. On the other hand, when using CNN models, we were unable to find a model that successfully recovered the correct key byte with less than 3 000 attack traces, even if, for the Hamming weight model, the attack would be successful if more attack traces would be considered, as shown in



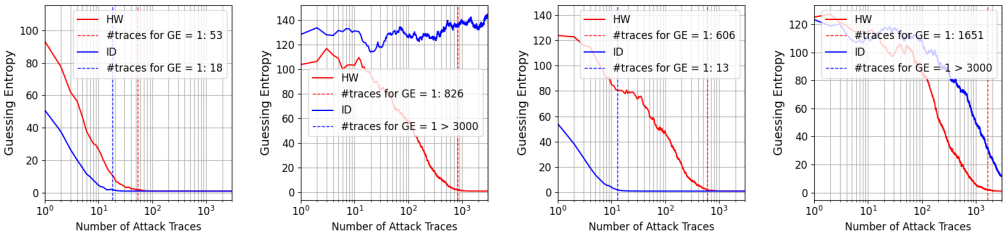
(a) MLP. **(b)** CNN

Figure 26: Non-optimized points of interest (OPOI): best guessing entropy results for the `CHES_CTF` dataset with different leakage models.



(a) MLP (w=80). **(b)** CNN (w=80). **(c)** MLP (w=40). **(d)** CNN (w=40).

Figure 27: Non-optimized points of interest (NOPOI): best guessing entropy results for the `CHES_CTF` dataset with different leakage models and resampling windows of 80 and 40 samples.

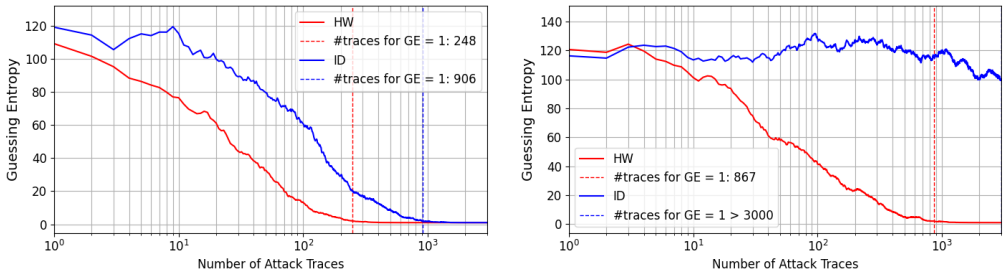


(a) MLP (w=20). **(b)** CNN (w=20). **(c)** MLP (w=10). **(d)** CNN (w=10).

Figure 28: Non-optimized points of interest (NOPOI): best guessing entropy results for the `CHES_CTF` dataset with different leakage models and resampling windows of 20 and 10 samples.

Figure 28b.

NOPOI with Trace Desynchronization In the NOPOI case with the `CHES_CTF` dataset, the best CNN model was found when a resampling window of 40 was considered. This way, to verify whether the same hyperparameter search configuration can also recover the key from the `CHES_CTF` dataset with desynchronization, we only consider the resampling window of 40. As for other datasets, we add artificial shifts, randomly chosen between -50 and 50 samples per trace. We search for CNN models with and without data augmentation, which consists of creating 300 synthetic traces per epoch, and randomly shifting them between up to 10 samples to left or right. Models are again always trained for 100 epochs.



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

Figure 29: Non-optimized points of interest (NOPOI) with desynchronization: best guessing entropy results for the `CHES_CTF` dataset with different leakage models with and without data augmentation.

Table 8: Minimum number of attack traces to get guessing entropy equal to 1 with the `CHES_CTF` 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	$Sbox(p_i \oplus k_i)$	>3 000, >3 000, 2 237, 1 390, >3 000, >3 000, 2 150, 1 747, 2 999, >3 000, 2 559, >3 000, 2 540, 2 508, 2 557, 2 795
CNN	$HW(Sbox(p_i \oplus k_i))$	>3 000, 787, 374, 411, 2 518, 1 788, 1 115, 747, 1 142, 1 594, 619, 1 027, 980, 1 109, 317, 555
MLP	$Sbox(p_i \oplus k_i)$	>3 000, 2 189, 14, 15, >3 000, 14, 16, 202, >3 000, 2 168, 13, 131, 31, 20, >3 000, 691
MLP	$HW(Sbox(p_i \oplus k_i))$	288, 227, 8, 7, 182, 190, 141, 7, 281, 13, 7, 178, 247, 123, 8, 8

Figure 29 shows the results for the best CNN models with (left) and without (right) data augmentation. The best attack results are obtained when data augmentation is applied, which shows the same trend as for previous datasets. In particular, for this dataset, we obtain the best result for the Hamming weight leakage model.

Attacking the Full AES Key with the NOPOI Scenario Here, we select the best MLP and CNN models from the NOPOI scenario (without desynchronization) for both Hamming weight and Identity leakage models and verify how each best model is able to retrieve the remaining key bytes. Results are shown in Table 8. For the CNN case with the Identity leakage model, the best CNN model can successfully recover 10 out of 16 key bytes with less than 3 000 attack traces. For the Hamming weight case, the CNN results are better, and we can recover 15 out 16 key bytes. On the other hand, the best MLP models showed successful full AES key recovery for the Hamming weight leakage model. For the Identity leakage model case, we could recover 12 out of 16 key bytes with the same MLP model. Note that we always re-initialize the neural network with the same initial weights from the best model found during the NOPOI hyperparameter search scenario. We hypothesize that MLP shows somewhat better performance than CNN as the trained model does not fit the data so well, consequently generalizing a bit better.

In [GJS19], the authors presented solutions for the CHES CTF 2018 challenges, which include the same `CHES_CTF` dataset evaluated in our paper. In that case, the authors applied a decision tree classifier and a SAT solver over the key schedule process. As a result, it was verified that the key schedule process is implemented without masking protection, which provided successful key recovery results with a single attack trace. Obviously, the attack implemented in [GJS19] is much simpler than ours and exploits the significant Hamming weight leakages from the processing of (unprotected) sub-key bytes, which is an implementation flaw. However, when considering a leakage model from the first S-box output, first-order leakages are not present, and, therefore, the profiling model

needs to implement a second-order attack. This way, we provide results for `CHES_CTF` by implementing the same attack process as for the previous datasets as a way to further validate our arguments when feature selection (as for RPOI and OPOI cases) is not considered.

6.5 Summary of Results

In our analyses, we provided extensive experimental results for four different datasets that contain side-channel measurements from software implementations of the AES encryption process protected with the first-order Boolean masking. Overall, the four datasets contain lengthy traces, and one of our goals is to understand how feature selection impacts the performance of deep learning-based profiling models. As iterated during the discussion of the results, we always deploy the same *grid* hyperparameter search settings for the RPOI scenarios with all datasets (excluding `CHES_CTF` as it does not contain information about mask shares) and the same *random* hyperparameter search settings for the OPOI and NOPOI scenarios with all four datasets.

Here, we provide a summary of our results. Table 9 provides the minimum number of attack traces required for guessing entropy equal to 1 for one target key byte, the number of points of interest from the best-found model, the percentage of profiling models that delivered successful attack results (Search Success column), and the number of trainable parameters for the best found model. For the ASCAD and `DPAV4.2` datasets, the RPOI results indicate that this feature selection scenario tends to provide the best possible models, and the search success is close to 100%. As we are targeting higher SNR peaks, the models are always trained on very leaky samples, which justifies the ease of finding optimal models from the grid search processes. Note that our neural networks implement optimal profiling models that recover the target key byte with a single attack trace.

The best models found from the OPOI feature selection show a similar trend for the `ASCADf`, `ASCADr`, and `CHES_CTF` datasets: attacking continuous trace intervals where the main SNR peaks from two mask shares are located tends to provide inferior profiling models when compared to the RPOI and NOPOI scenarios. For the `DPAV4.2` dataset, OPOI results are superior when compared to the NOPOI case. This is expected as the `DPAV4.2` dataset contains very lengthy traces and attacking larger intervals, as is the case of NOPOI, includes a massive amount of noise. Although we are able to find successful profiling models for NOPOI with the `DPAV4.2` dataset, hyperparameter search attempts should be larger than 500 in order to reach better models.

For the `ASCADf`, `ASCADr`, and `CHES_CTF` datasets, the NOPOI scenario delivers optimal profiling models, where the key is recovered with less than ten attack traces. In particular, for both ASCAD datasets, we found optimal models that retrieve the correct key byte with a single attack trace. For the `CHES_CTF` dataset, the best found model recovers the key with only eight attack traces. From the perspective of the number of trainable parameters, we see that different feature selection scenarios result in models of different sizes (as expected), but even the largest models can still be considered relatively small (especially compared to deep neural networks used in different domains but also architectures used in [LZC+21]). We also presented the attack results for all key bytes. While there are several cases where we can efficiently break the whole (or most of the) key, there is also a fluctuation of the attack performance (aligned with [EST+22]). This fluctuation is most likely due to the measurement or cryptographic implementations, and our conclusions are consistent in interpreting these results.

Therefore, this extensive experimental evaluation indicates that it is possible to keep hyperparameter search space unchanged in security evaluations across multiple datasets and feature selection, where the variable to be taken into account is the number of search attempts. This variable is also defined as the *learnability* in [PHPG21].

Table 9: Points of interest, minimum number of attack traces to get guessing entropy equal to 1, model search success (when GE=1), and number of trainable parameters for all datasets and feature selection scenarios. Settings where we obtain $GE = 1$ are denoted in bold style.

Dataset	Neural Network Model	Feature Selection Scenario	Amount of POIs (HW/ID)	Attack Traces (HW/ID)	Search Success (%) (HW/ID)	Trainable Parameters (HW/ID)
ASCADf	MLP	RPOI	200/100	5/**1**	99.22%/96.86%	82 209/429 256
ASCADf	CNN	RPOI	400/200	5/**1**	99.23%/99.08%	499 533/158 108
ASCADf	MLP	OPOI	700/700	480/104	82.80%/68.80%	16 309/10 266
ASCADf	CNN	OPOI	700/700	744/87	55.53%/35.33%	594 305/62 396
ASCADf	MLP	NOPOI	2 500/2 500	7/**1**	74.50%/39.00%	2 203 009/5 379 256
ASCADf	CNN	NOPOI	10 000/10 000	7/ **1**	15.40%/2.45%	545 693/439 348
ASCADf	CNN	NOPOI desync	10 000/10 000	532/36	2.44%/2.64%	268 433/64 002
ASCADr	MLP	RPOI	200/20	3/**1**	99.23%/100%	565 209/639 756
ASCADr	CNN	RPOI	400/30	5/**1**	100%/100%	575 369/636 224
ASCADr	MLP	OPOI	1 400/1 400	328/129	71.40%/37.25%	31 149/34 236
ASCADr	CNN	OPOI	1 400/1 400	538/78	47.92%/23.95%	270 953/87 632
ASCADr	MLP	NOPOI	25 000/25 000	6/ **1**	44.39%/7.02%	5 243 209/12 628 756
ASCADr	CNN	NOPOI	25 000/25 000	7/ **1**	19.17%/4.35%	369 109/721 012
ASCADr	CNN	NOPOI desync	25 000/25 000	305/73	0.71%/1.04%	22 889/90 368
CHES_CTF	MLP	OPOI	4 000/4 000	27/1 905	54.24%/0.09%	1 383 609/213 106
CHES_CTF	CNN	OPOI	4 000/4 000	462/>3 000	2.99%/0.00%	666 429/593 780
CHES_CTF	MLP	NOPOI	3 750/30 000	8/13	69.75%/11.11%	198 209/6 091 856
CHES_CTF	CNN	NOPOI	7 500/7 500	238/>3 000	7.89%/0.00%	216 673/1 319 552
CHES_CTF	CNN	NOPOI desync	7 500/7 500	248/906	12.22%/3.05%	374 233/564 436
DPAV4.2	MLP	RPOI	900/100	3/**1**	100%/95.25%	956 009/287 956
DPAV4.2	CNN	RPOI	700/200	8/**1**	93.05%/97.54%	34 709/697 112
DPAV4.2	MLP	OPOI	800/800	11/**1**	93.33%/71.62%	48 159/251 856
DPAV4.2	CNN	OPOI	800/800	31/170	33.33%/2.63%	1 158 857/424 956
DPAV4.2	MLP	NOPOI	15 000/3 750	400/2 577	5.49%/0.14%	1 501 009/1 603 056
DPAV4.2	CNN	NOPOI	3 750/15 000	2 767/>3 000	0.16%/0.00%	578 033/89 384
DPAV4.2	CNN	NOPOI desync	3 750/3 750	>3 000	0.00%/0.00%	34 836/72 677

7 Conclusions and Future Work

Depending on computational power (which is reflected in the number of hyperparameter search attempts) and security assumptions (i.e., the knowledge of secret random masks and source code), the security evaluators may define different profiling model strengths: from the optimal and sub-optimal to wrong models. This paper shows that when conducting security evaluations on the first-order protected software AES implementations, feature selection also has a significant impact when searching for an efficient deep learning-based profiling model. Therefore, we investigated three feature selection scenarios ranging from the worst-case security assumptions to online attacks that relax most of the adversary assumptions. Our results indicate that for the considered datasets, an extensive hyperparameters search for MLP and CNN models can find highly efficient and relatively small models (with up to eight hidden layers and a few million trainable parameters) that can recover the target key byte with a single trace for most of the datasets. We observed that different datasets lead to the variation of model performances for different feature selection scenarios. From the evaluated cases, we concluded that the ASCAD and `CHES_CTF` datasets lead to similar outcomes, where attacking optimized features from higher SNR peaks (RPOI) or relaxing feature selection and attacking all samples (NOPOI) lead to optimal profiling model recovering the key with less than ten attack traces (one attack trace for the ASCAD cases). For `DPAV4.2`, we observed the same trend for the RPOI case. Relaxing feature selection with the NOPOI case for the `DPAV4.2` dataset led to the key recovery with model performance relatively inferior compared to other datasets. However, different performances for different datasets are expected as we consider the same hyperparameter search process for all datasets, simplifying a security evaluation. Finally, we successfully and efficiently recovered the key from scenarios with trace desynchronization when feature selection is disregarded.

In future work, we plan to evaluate the performance of different feature selection

scenarios against newer and more protected datasets, such as ASCADv2 [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). Additionally, we showed that small neural networks could efficiently break various datasets with only a few attack traces. Still, it would also be interesting to consider the profiling complexity in terms of the number of profiling traces in different feature selection scenarios.

Acknowledgements

This work received funding in the framework of the NWA Cybersecurity Call with project name PROACT with project number NWA.1215.18.014, which is (partly) financed by the Netherlands Organisation for Scientific Research (NWO). Additionally, this work was supported in part by the Netherlands Organization for Scientific Research NWO project DISTANT (CS.019).
The authors thank our anonymous reviewers and our shepherd for their valuable comments and suggestions.

References

[AB09] Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, USA, 1st edition, 2009.

[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.

[BDMS22] Olivier Bronchain, François Durvaux, Loïc Masure, and François-Xavier Standaert. Efficient profiled side-channel analysis of masked implementations, extended. *IEEE Transactions on Information Forensics and Security*, pages 1–1, 2022.

[BHM+19] Olivier Bronchain, Julien M. Hendrickx, Clément Massart, Alex Olshevsky, and François-Xavier Standaert. Leakage certification revisited: Bounding model errors in side-channel security evaluations. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 713–737. Springer, 2019.

[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.

[BS20] Olivier Bronchain and François-Xavier Standaert. Side-channel countermeasures' dissection and the limits of closed source security evaluations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(2):1–25, 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.

[dCGRP19] Eloi de Chérisey, Sylvain Guilley, Olivier Rioul, and Pablo Piantanida. Best information is most successful mutual information and success rate in side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):49–79, 2019.

[EST+22] Maximilian Egger, Thomas Schamberger, Lars Tebelmann, Florian Lippert, and Georg Sigl. A second look at the ascad databases. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 75–99. Springer, 2022.

[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.

[GJS19] Aron Gohr, Sven Jacob, and Werner Schindler. CHES 2018 side channel contest CTF - solution of the AES challenges. *IACR Cryptol. ePrint Arch.*, page 94, 2019.

[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.

[MDP19a] 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.

[MDP19b] 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*, 2020(1):348–375, Nov. 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.

[PHPG21] Stjepan Picek, Annelie Heuser, Guilherme Perin, and Sylvain Guilley. Profiled side-channel analysis in the efficient attacker framework. In Vincent Grosso and Thomas Pöppelmann, editors, *Smart Card Research and Advanced Applications - 20th International Conference, CARDIS 2021, Lübeck, Germany, November 11-12, 2021, Revised Selected Papers*, volume 13173 of *Lecture Notes in Computer Science*, pages 44–63. Springer, 2021.

[RGV12] Oscar Reparaz, Benedikt Gierlichs, and Ingrid Verbauwhede. Selecting time samples for multivariate dpa attacks. In *Proceedings of the 14th International Conference on Cryptographic Hardware and Embedded Systems*, CHES'12, page 155–174, Berlin, Heidelberg, 2012. Springer-Verlag.

[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.

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

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, USA, 2014.

[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.