The Need for MORE: Unsupervised Side-channel Analysis with Single Network Training and Multi-output Regression

Ioana Savu[1], Marina Krček[2], Guilherme Perin[3], Lichao Wu[4] and Stjepan Picek[4]

[1] NXP Semiconductors, Netherlands
[2] Delft University of Technology, Netherlands
[3] Leiden University, Netherlands
[4] Radboud University, Netherlands

Abstract. Deep learning-based profiling side-channel analysis has gained widespread adoption in academia and industry due to its ability to uncover secrets protected by countermeasures. However, to exploit this capability, an adversary must have access to a clone of the targeted device to obtain profiling measurements and know secret information to label these measurements. Non-profiling attacks avoid these constraints by not relying on secret information for labeled data. Instead, they attempt all key guesses and select the most successful one. Deep learning approaches form the foundation of several non-profiling attacks, but these methods often suffer from high computational complexity and limited performance in practical applications.
This work explores the performance of multi-output regression (MOR) models in side-channel analysis. We start with the recently proposed multi-output regression (MOR) approach for non-profiling side-channel analysis. Then, we significantly improve its performance by updating the 1) loss function, 2) distinguisher, and 3) employing a novel concept of validation set to reduce overfitting. We denote our approach as MORE - Multi-Output Regression Enhanced, which emphasizes significantly better attack performance than MOR. Our results demonstrate that combining the MORE methodology, ensembles, and data augmentation presents a potent strategy for enhancing non-profiling side-channel attack performance and improving the reliability of distinguishing key candidates.

Keywords: Side-channel Analysis · Deep learning · Non-profiling · Regression · Data Augmentation

1 Introduction

Side-channel attacks (SCAs) explore unintended leakage of information from secure devices. To evaluate the threats from such attacks, security certification methods are commonly applied to secure devices during the design and pre-market phases. This process usually considers two main types of side-channel attacks: profiling [CRR02] and non-profiling attacks [KJJ99, BCO04]. From the security evaluator's perspective, applying profiling attacks allows a more formal security assessment as it follows a threat model in which the attacker has full access to the clone version of the target device. On the other hand, non-profiling attacks directly use the side-channel measurements from the target device. While profiling attacks are more potent and considered the worst-case attack assumption [BDMS22], non-profiling attacks are more realistic from a practical point of view.

In recent years, the application of deep learning to side-channel analysis (DLSCA) has received significant attention from the academic community in the profiling SCA setting [PPM+22]. The research has predominantly concentrated on profiling SCA, whereas non-profiling techniques received less attention. Non-profiling deep learning-based SCA was first proposed by B. Timon in 2019 [Tim19], with an approach called Differential Deep Learning Analysis (DDLA). Although its performance is better than conventional non-profiling attacks such as CPA, it is mainly criticized for practical limitations. Indeed, to attack one key byte, DDLA needs to train a deep neural network 256 times (one network for each key hypothesis for commonly attacked byte-oriented cipher like AES) to brute force all possible key bytes. Such an attack may easily become impractical, considering a dataset with millions of measurements. From the adversary's perspective, this also means that the cost of a DDLA attack may easily become higher than the benefit of the attack itself. In [KHK22], the authors proposed several solutions to mitigate observed issues with DDLA. They presented a parallel network architecture to decrease time consumption. However, that increased the memory consumption, which the authors resolved using shared layers. Recently, the authors of [DLH+22] proposed using multi-output learning (MOL) [XST+19], where the model is trained to predict multiple outputs from a single input simultaneously. Their work considered multi-output classification (MOC) and multi-output regression (MOR) [BVBL15] architectures for non-profiling SCA. These models reduce the complexity of Timon's attack as they require a single neural network training. Moreover, the authors achieved lower execution time and better performance. Still, the performance remains far from desired (or achievable with profiling deep learning-based SCA).

This work identifies improvements over the MOR strategy for non-profiling SCA proposed in [DLH+22] that allow significantly better attack performance and constitute the core of our approach called MORE - Multi-Output Regressions Enhanced. Additionally, we investigate techniques like data augmentation and ensembles to enhance the attack performance for MORE further. Our main contributions are:

- We evaluate standard regression loss functions for multi-output regression models and propose new loss functions that significantly increase attack efficiency. Specifically, we show that Z-Score normalized Mean Square Error (MSE) allows better attack performance.
- We propose a novel method that computes the Pearson correlation coefficient between true and predicted labels to distinguish the correct key. The results show that it can more efficiently rank key candidates from multi-output predictions compared to [DLH+22].
- We propose to use a validation set approach to mitigate overfitting and enhance the attack performance. The validation set for MORE does not use labeled data as in profiling SCA, but the concept is similar since the validation set differs from the training set. To the best of our knowledge, this approach was not used before in non-profiling SCA.
- We compare neural network sizes for both MORE and profiling deep learning-based SCA, showing that profiling attacks commonly require smaller neural networks (as measured by the number of trainable parameters). Thus, for tuning MORE-based models, hyperparameter tuning becomes more crucial.
- We show that ensembles and data augmentation can further improve the performance of the MORE-based SCA.
- Our experiments consider different neural network architectures (multilayer perceptron and convolutional neural network), dataset preprocessing methods (trimmed interval, raw measurements), countermeasures (masking, hiding), and leakage models (Hamming Weight/Distance, Identity). Our results outperform MOR by obtaining $3.9\times$ higher success rate on average, making it possible to use non-profiling deep learning SCA significantly more efficiently than previous works.

We provide the source code of MORE implementation[1].

2 Preliminaries

We use calligraphic characters such as $\mathcal{X}$ to represent sets, while the related upper-case letters X denote random variables and random vectors $\mathbf{X}$ over $\mathcal{X}$. The associated lower-case letters x and $\mathbf{x}$ indicate realizations of X and $\mathbf{X}$, respectively. For functions, a sans-serif typeface is used (e.g., f). The symbol k represents a key byte candidate that derives its value from the key space $\mathcal{K}$. The correct key byte, or the one assumed to be accurate by the adversary, is denoted by k^*. A dataset $\mathbf{T}$ consists of traces $\mathbf{t_i}$, where each trace $\mathbf{t_i}$ is paired with a label y_i. A full set of labels with c classes is represented by $\mathcal{Y} = y^1, y^2, \cdots, y^c$. In the dataset $\mathbf{T}$, each trace $\mathbf{t_i}$ is connected to a plaintext/ciphertext $\mathbf{d}_i \in \mathcal{D}$ and a key $\mathbf{k}_i$, or $k_{i,j}$ and $d_{i,j}$ when focusing on partial key recovery for byte j. In this study, we target only a single key byte, so the byte vector notation is omitted in equations. The vector of parameters to be learned in a profiling model is denoted by $\boldsymbol{\theta}$.

2.1 Side-channel Analysis

Side-channel analysis (SCA) can be divided into two types based on the availability of a fully controlled cloned device: profiling SCA and non-profiling SCA. Profiling side-channel attacks map inputs (e.g., side-channel traces) to outputs (e.g., probability vector of key hypotheses). These attacks occur in two phases: profiling and attack phases. In the profiling phase, the adversary constructs a profiling model that draws its foundation from profiling traces, which encapsulate measurements of side-channel information sourced from a clone of the intended target device. Possessing this clone provides the attacker with comprehensive insights into the device, facilitating the learning and training of the model. In the attack phase, the trained model processes each attack trace obtained from the target device, producing a probabilities vector representing the likelihood of the associated leakage value or label j. Based on this probability vector, the adversary selects the best key candidate. If the adversary creates a good model, a minimal set of measurements from the targeted device might suffice to breach its security. Examples of such attacks include the template attack [CRR02], stochastic models [SLP05], and machine learning-based attacks [MPP16].

The focus of this work is on non-profiling side-channel attacks. Non-profiling SCA relies on the correlation between key-related intermediate values and leakage measurements. To execute such attacks, an adversary collects traces during cryptographic operations. If the chosen key-related intermediate value correlates significantly with leakage measurements, these measurements can be used to verify guesses for a part of the key. For each potential value of the relevant key part (or the key itself), the adversary separates the traces into groups based on the intermediate value predicted by the current key guess. The current key guess is likely accurate if each group is noticeably distinct from the others (with the definition of 'difference' depending on the attack methods). Non-profiling attacks assume less powerful adversaries without access to a clone of the targeted device, so they might require millions of measurements to extract secret information. Simple power analysis (SPA), differential power analysis (DPA) [KJJ99] and correlation power analysis (CPA) [BCO04] are common examples of such attacks.

The most common metrics for assessing the performance of the SCA are Guessing Entropy (GE) and Success Rate (SR). For further information about these metrics, we refer readers to [SMY09]. **Guessing Entropy (GE)** is the average position of the correct key k^* in the guessing vector $\mathbf{g} = [g_1, g_2, ..., g_{|K|}]$. The guessing vector is the output of the attack phase, where the key candidates are sorted in decreasing order of probability. The

[1]`https://anonymous.4open.science/r/MORE/README.md`

most likely key is g_1, while the least likely is $g_{|K|}$. **The success rate (SR)** of order o is the average probability that the correct key k^* is located within the first o elements of the guessing vector $\mathbf{g}$.

2.2 Machine Learning and SCA

Machine learning algorithms can be classified into several categories based on their learning approach. Supervised learning algorithms utilize labeled training data. These algorithms train a model f to predict labels for previously unseen data by analyzing the data and labels. Most supervised learning methods follow the Empirical Risk Minimization (ERM) framework, where the model parameters $\boldsymbol{\theta}$ are obtained by solving the following optimization problem:

$$\arg\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i}^{n} \mathcal{L}(\mathsf{f}_{\boldsymbol{\theta}}(\mathbf{t_i}), y_i), \tag{1}$$

where $\mathcal{L}$ represents the loss function and n is the size of the dataset T. Supervised machine learning occurs in two phases: training and testing. Classification and regression are two common problem types in supervised machine learning that differ primarily in the nature of their output.

- **Classification** problems involve categorizing input data into discrete classes or categories. In these problems, the output is a discrete value representing the class or category that an input data point belongs to. Various algorithms can be used for classification, such as logistic regression, decision trees, support vector machines, and neural networks.
- **Regression** problems involve predicting continuous output values based on input data. In these problems, the output is a continuous numeric value, often representing a measurement or quantity. Common regression algorithms include linear regression, polynomial regression, ridge regression, and support vector regression.

There is a natural relationship between machine learning algorithms based on learning style and side-channel analysis. Profiling attacks depend on a labeling function that utilizes secret information from a device under the adversary's control. Since supervised machine learning involves a labeling function with secret information, it falls within the profiling attack setting. Similarly, although the true label is unknown, non-profiling attacks hypothesize labels without knowing the secret information. Thus, supervised learning can also be applied to this type of attack using the measurements and the hypothesized labels.

2.2.1 Common Regression Loss Functions

We define common loss functions for regression tasks [WMZT20] that we use in this work. As mentioned, we hypothesize the labels in non-profiling attacks by calculating them from the key byte hypothesized values. The following definitions refer to those hypothesized labels as actual, true, or correct. However, these are not correct labels as used in profiling SCA as we do not know the correct key, but we use all possible key bytes and obtain what the labels would be given a specific key value.

Mean Squared Error (MSE). MSE is a commonly used measure of the difference between actual and predicted values. It measures the average of the square of the differences between the actual and predicted values according to the following expression:

$$\mathcal{L}_{MSE} = \frac{1}{n} \sum_{i=1}^{n} \left\| y_i - \hat{y_i} \right\|, \tag{2}$$

where n is the size of the dataset, while true and predicted labels are represented with y_i and $\hat{y_i}$, respectively. MSE is used in regression problems, where the goal is to predict a continuous value based on input variables. Lower MSE values indicate a better fit between the actual and predicted values.

Mean Absolute Error (MAE). MAE calculates the average of the absolute differences between the actual and predicted values as follows:

$$\mathcal{L}_{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y_i}|. \tag{3}$$

Unlike MSE, MAE does not square the differences and is more robust to outliers. As such, MAE tends to be a less efficient loss function than MSE for our non-profiling attack, but we evaluate MAE to show empirically that our assumptions are valid.

Huber Error. Huber Error is a loss function used in regression problems that balances the mean squared error and the mean absolute error. This loss function is defined as:

$$\mathcal{L}_{Huber} = \begin{cases} \frac{1}{2n}\sum_{i=1}^{n} \|y_i - \hat{y_i}\| & \text{if } |y_i - \hat{y_i}| \leq \delta \\ \frac{\delta}{n}\sum_{i=1}^{n}(|y_i - \hat{y_i}| - 0.5\delta) & \text{otherwise,} \end{cases} \tag{4}$$

where the δ parameter specifies the transition point between the quadratic and linear regions of the loss function. Larger δ values make the function more robust to outliers, which we want to avoid. In our experiments, δ is set to 1 based on the preliminary investigations, and it makes a good balance between MAE and MSE, where it should be less robust to outliers but also sensitive to small errors.

2.3 Non-profiling Multi-Output Regression-based Side-channel Attacks

Do et al. developed a novel method called Multi-output regression (MOR) DLSCA to decrease the computational efforts of DDLA [DLH+22]. Rather than training 256 models, where each aims to classify hypothetical labels with probabilities accurately, MOR utilizes the concept of multi-output regression, which seeks to *regress* the prediction outputs to the actual label values. Specifically, a model is trained to map input leakage traces to the actual values of all possible $y_i(k)$, which denotes the key-related intermediate data (label) given a specific key byte k. The most likely key k^* is determined by identifying the smallest loss measured by MSE.

$$k^* = \arg\min_{k} \frac{1}{n}\sum_{i}^{n} \left\| y_i(k) - \mathsf{f}_{\boldsymbol{\theta}}^{k}(\mathbf{t_i}) \right\|, \;\; k \in \mathcal{K}, \tag{5}$$

$\mathsf{f}_{\boldsymbol{\theta}}^{k}(\mathbf{t_i})$ represents the model's prediction value to approximate $y_i(k)$. In SCA, the regression output is a key-related intermediate data, e.g., the output of $\mathsf{Sbox}(d_i \oplus k)$, and can be further parameterized by the leakage models such as the Identity (ID) or Hamming Weight (HW) (for details about leakage models, see Section 2.5).

2.4 Datasets

This section introduces the datasets that are used in the experiments. We note that the dataset sizes (number of traces available for attack) are reported with each corresponding experiment.

ASCADf. This dataset contains electromagnetic (EM) traces collected from an 8-bit AVR-based microcontroller that implements a first-order Boolean masked AES-128 [BPS+20]. In our experiments, we consider two versions of this dataset. The first version consists of the trimmed interval of 700 features that includes the second-order leakages of the third S-box in the first encryption round, which we refer to as ASCADf. We also run experiments with the raw measurements from the NOPOI scenario presented in [PWP22], which contains 10 000 features, and we refer to it as `ASCADf_nopoi_10000` dataset.[2]

ASCADr. This dataset was collected from the same measurement setup as ASCADf. However, the difference is that the encryption keys are randomized for the profiling phase [BPS+20]. For this work, we use the part of this dataset commonly used as the attack set in DLSCA, where all plaintexts are encrypted with the same key. We consider two versions of this dataset. The first version consists of the trimmed interval of 1 400 features that includes the second-order leakages of the third S-box in the first encryption round, which we refer to as ASCADr. We also run experiments with the raw measurements from the NOPOI scenario from [PWP22], which contains 25 000 features, and we refer to it as `ASCADr_nopoi_25000` dataset.[3]

AES_RD. This dataset contains traces collected from a smart card with an 8-bit Atmel AVR microcontroller. As a countermeasure, random delays using the Floating Mean method were added to the traces to protect against simple side-channel attacks [CK09]. The plaintexts are encrypted using the same fixed key, and the collected traces have 3 500 features each.

AES_HD. This dataset targets an unprotected hardware implementation of AES-128 written in VHDL in a round-based architecture. Using a Xilinx Virtex-5 FPGA of a SASEBO GII evaluation board, side-channel traces were collected using a high sensitivity near-field EM probe positioned above a decoupling capacitor on the power line [KPH+19]. The same key is used for all collected traces, which consist of 1 250 features.

2.5 Leakage Models

This work considers two commonly used leakage models - Identity (ID) and Hamming Weight (HW). Since the datasets consider AES encryption, we consider the S-box function outputs. S-box is fixed and publicly known for AES but depends on d_i and k. The ID leakage model is the intermediate value from the S-box function $\mathsf{Sbox}(d_i \oplus k)$, while for the HW model, the Hamming weight value of the intermediate is used: $HW(S_{box}(d_i \oplus k))$. For the AES_HD dataset, we use the Hamming Distance leakage model instead of the Hamming weight. As commonly done for AES, we compute the XOR between the final output and the S-box input of the last round [KPH+19].

3 Related Work

The development of non-profiling deep learning side-channel analysis began with B. Timon's introduction of the deep learning-based distinguisher: DDLA [Tim19] in 2019. The DDLA method involves training a neural network for each key guess and selecting the one with the best result as the correct key guess. While effective, DDLA is computationally expensive and requires a non-bijective leakage function. Kuroda et al. conducted a follow-up study on DDLA, examining neural network structures and attack points, and concluded that simpler architectures and a broader range of points of interest were preferable [KFYF21]. Alipour et al. investigated DDLA's performance against a hiding-based AES countermeasure using correlated noise generation and found DDLA was less effective than CPA in this context [APB+20]. Kwon et al. proposed improvements to DDLA to increase its speed and enhance neural network architectures' performance [KHK22]. Do et al. further analyzed DDLA to understand its behavior in more complex scenarios and explored different data preparation techniques and neural network architectures to improve performance [DHDP22].

[2] Raw measurements contain 100 000 sample points. We resampled it into 10 000 to reduce the input dimension that significantly impacts the profiling model size.

[3] Raw measurements contain 250 000 sample points. For the same reason as ASCADf, they are resampled into 25 000.

Hoang et al. introduced a non-profiling SCA technique using multi-output classification, achieving up to 30× faster and 20% better results than DDLA [HDD22]. Do et al. investigated multi-output classification (MOC) and multi-output regression (MOR) models for non-profiling SCA [DLH+22]. They concluded that MOC reduced execution time compared to DDLA and showed that MOR, although slower than MOC, worked for the Identity leakage model. Both MOC and MOR outperformed DDLA by at least 25%. Finally, Do et al. further improved the MOR concept by using identity labeling [DHD23]. The authors reported results up to 40× faster than DDLA and at least 30% better success rate. Wu et al. recently proposed a non-profiling deep learning-based attack called PLDL based on the bijective relationship between plaintext/ciphertext and secret information [WPP23]. The authors reported the attack as powerful and easily rivaling the performance of profiling deep learning-based SCA. Note that multiple works use unsupervised deep learning techniques like autoencoders, see, e.g. [RAD20, WP20]. Still, since such methods are used as preprocessing for profiling SCA, we do not consider them further. Since MOR is already shown to perform much better than DDLA, we do not compare our MORE approach with DDLA but only with MOR. Finally, since PLDL is a two-stage classification-based attack (more similar to a profiling attack as there is still a notion of the labeling function), we do not consider it in this work.

4 MOR Enhanced (MORE)

4.1 Loss Functions and Distinguishers

The learning process implemented by a multi-output regression model should minimize the empirical error for all outputs. For the application to non-profiling SCA, the MOR model should minimize the empirical error for only one of the outputs because only one output is related to the correct key candidate. Thus, it is important to find the most efficient loss function that is also more sensitive (i.e., less robust) to *outliers*. Outliers are usually variables whose characteristics are distant from the mean of the data group. This characteristic is desirable from an adversary's perspective because the outlier is the attacker's best guess for the correct key candidate. In this case, if we consider a loss function, the wrong keys would have a similar loss value, where the correct key should be an outlier having a lower loss value than the rest.

The loss functions outlined in Section 2.2.1 are typically employed in regression models. As mentioned, we aim to develop a learning process where the loss function is highly sensitive to outliers. A functional model should produce prediction errors for all incorrect key candidates significantly larger than those for the correct key. To this end, we propose to use the Z-Score normalization [PS15] to build the loss function. Specifically, Z-Score normalization aids outlier detection by making it easier to identify data points that deviate significantly from the mean. In the context of our learning process, normalizing the predictions before computing the loss function accentuates the differences between data points. The increased sensitivity can help the learning process focus more on the correct key candidate, considered an outlier, than the incorrect key candidates, potentially leading to improved performance in MOR-based non-profiling attacks. As such, we propose combining the aforementioned loss functions with the Z-Score normalization. For instance, we can rewrite MSE defined in Eq. (2) to Eq. (6):

$$\mathcal{L}_{Z-scoreMSE} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \mu(\mathbf{y})}{\sigma(\mathbf{y})} - \frac{\hat{y}_i - \mu(\hat{\mathbf{y}})}{\sigma(\hat{\mathbf{y}})}\right)^2, \tag{6}$$

where the original y_i and $\hat{y}_i$ are normalized and MSE is calculated with normalized values. μ and σ denote the mean and standard deviation functions, respectively, and n is the size of the dataset.

Additionally to the three common regression loss functions and their normalized versions, we propose another loss function, **Pearson Correlation loss**, for which we also test its normalized version. Wu et al. [WWK+23] show that assessing the correlation between key distribution[4] and guessing vector can considerably enhance the learning rate of the profiling model. In our case, we do not use label distance, as we can directly utilize the hypothesized labels. Thus, we introduce a novel loss function that aims to maximize the linear correlation between the hypothesized (actual) and predicted labels for all possible keys, defined as follows:

$$\mathcal{L}_{Pearson} = 1 - |\rho(\mathbf{y}(k), \hat{\mathbf{y}}(k))|, \; k \in \mathcal{K}, \tag{7}$$

where ρ represents the Pearson correlation. $\mathbf{y}(k)$ and $\hat{\mathbf{y}}(k)$ represent vectors of the actual and predicted labels for all n traces given a key byte k. Compared with other loss functions, $\mathcal{L}_{Pearson}$ considers the correlation, leveraging inter-key dependence that could lead to more efficient learning. Thus, this paper benchmarks eight loss functions where, in figures, we differentiate between the non-normalized and normalized ones by adding a prefix 'z_score' or 'z' before the loss function names 'mse', 'mae', 'huber', and 'corr'.

Since we defined different loss functions that we evaluate in our experiments, we also introduce considered key distinguishers. In a non-profiling attack, the most likely key is commonly obtained by sorting the key candidates according to a distinguisher. First, we propose a novel key distinguisher, **Pearson correlation distinguisher**, based on the Pearson correlation coefficient between the true labels vector for each key candidate $\mathbf{y}(k)$ and its predicted labels $\hat{\mathbf{y}}(k)$. The most likely key k^* has the maximum correlation coefficient among all key candidates, defined with Eq. (8).

$$k^* = \arg\max_k \rho(\mathbf{y}(k), \hat{\mathbf{y}}(k)), \; k \in \mathcal{K}. \tag{8}$$

The Pearson correlation distinguisher and Pearson loss function use the same function to calculate the correlation. However, when used as the loss function, the correlation function is inverted for training. Besides, we consider a **loss function distinguisher**. Indeed, MOR models leverage the loss value for each key candidate k. The most likely key k^* is the one with minimum loss function value among all key candidates, defined in Eq. (9).

$$k^* = \arg\min_k \mathcal{L}(k), \; k \in \mathcal{K}. \tag{9}$$

Note that in the case where we use the loss function for the distinguisher, and the loss function for training the model was the Pearson correlation loss function, the distinguisher minimizes the inverted correlation function, which converges to maximizing correlation as is done using the Pearson correlation distinguisher.

4.2 Objective Function for Hyperparameter Tuning

Our MOR models are deep neural networks, so hyperparameter tuning is necessary, especially when attacking datasets with low signal-to-noise ratio (SNR). Hyperparameter tuning is already a complex problem in profiling SCA where an objective function can be defined (i.e., with metrics extracted from a labeled validation set) [WPP22]. In profiling SCA, the main goal is to find a deep neural network that provides better generalization, which implies a more efficient attack. That requires computing SCA metrics, such as guessing entropy or success rate and selecting the model, which requires fewer validation traces to recover the correct key candidate. Unfortunately, the secret key is unknown in a non-profiling setting. Therefore, finding a model that provides the best SCA metrics is not possible.

[4]Key distribution measures differences between key candidates based on label distance.

Section 4.1 considers two distinguishers, based on Pearson correlation and loss function, to rank the key candidates. In our experiments, we verified that the first approach based on correlation provides faster convergence, which means that a MOR model may require fewer training epochs to recover the correct key candidate. Therefore, we consider all key candidates' maximum Pearson correlation value as the objective function for hyperparameter tuning for our approach - MORE.

4.3 Validation Set to Mitigate Overfitting

It is essential to realize that MOR models also show overfitting, meaning they do not generalize well on the previously unseen data. As a result, the training loss for every key candidate may decrease, increasing the difficulty of distinguishing the correct key as the outlier. The same occurs when we compute the Pearson correlation between predicted and true labels from training traces. To avoid selecting the wrong key byte as the correct one (minimum loss value or maximum Pearson correlation from all key byte candidates), one can compute the objective function from a separate set of traces not considered during training. The idea is similar to using *validation* traces in a profiling SCA setting. Still, the concept differs, as a validation set, in this case, does not have known labels. Typically, in non-profiling SCA, as is done in MOR, the model is trained on all collected traces, and the same dataset is used to obtain the most likely correct key byte using a distinguisher objective function (such as the loss function). We propose a different approach, dividing the collected traces into two datasets - one for training and one for validation. Just like in supervised learning, these two datasets are distinct. Once the model is trained on the training dataset, we select the most likely key byte from the validation set. To the best of our knowledge, this concept was not used in other deep learning non-profiling SCA methods.

Figure 1 illustrates the aforementioned overfitting case. The example is from a MOR-based MLP when the dataset is `ASCADf_nopoi_10000` with the Identity leakage model, trained with the Z-Score MSE loss function. The top two graphs show the Pearson correlation (left) and loss (right) values for training traces; the bottom two are for the validation traces. The line in red color represents loss values for the correct key, while the gray color is for the wrong key candidates. Using training traces, the correct key candidate is not the one with the highest correlation or minimum Z-Score MSE loss value. The attack would fail using training traces to distinguish the most likely key, which is how the key is selected in the original MOR attack [DLH+22]. On the other hand, by predicting on a separate set of validation traces for the same trained MOR model, the correct key candidate is easily identified as the outlier with both loss and correlation metrics. This shows that extracting the most likely key from validation traces could offer a better approach in MOR-based non-profiling attacks. Thus, we test this concept in the following section in the search for the best loss function.

4.4 Experimental Results

4.4.1 Loss Function Benchmark

The experiments in this section aim to identify the best loss function for a MOR model using the ASCAD datasets. We consider trimmed and resampled versions of ASCADf and ASCADr datasets to improve the results' generality. First, we consider the trimmed ASCAD datasets. We investigate the success rate of the MOR approach by considering all loss functions: MSE, MAE, Huber, correlation (Corr), and their Z-score normalized variants. We randomly drew 1 000 models from the hyperparameter search space defined in Table 1 using only MLP models as the datasets are not desynchronized. We computed the success rate (SR) based on whether these models could find the correct key byte.

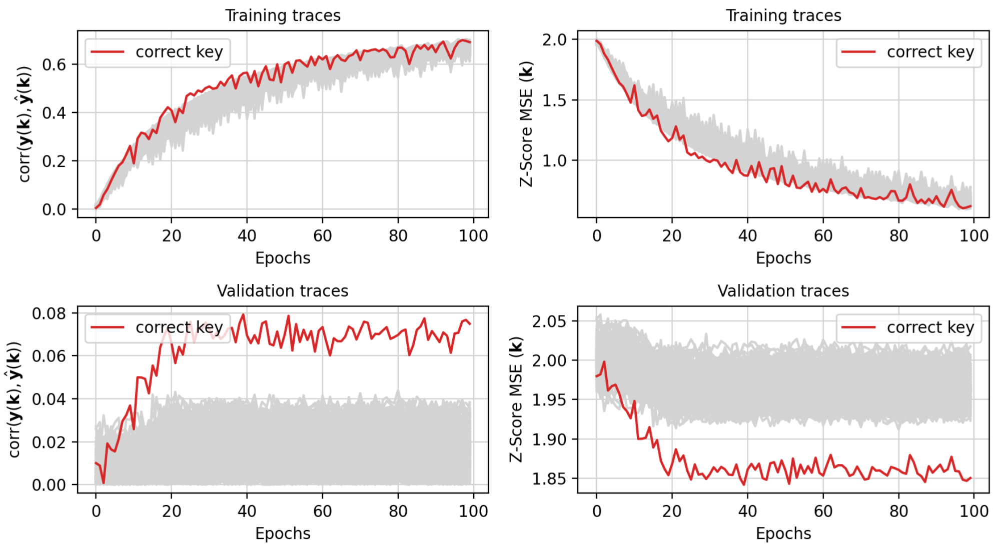


Figure 1: Training and validation performance of MOR for the ASCADf dataset using Pearson correlation and Z-score MSE metrics. The training set consists of 48 000 traces, while the validation set has 12 000 traces. Gray areas represent the wrong keys.

Table 1: Hyperparameter search space for finding the best MLP and CNN models.

Hyperparameter	MLP			CNN		
	Min		Max		Step	
Learning Rate	0.00005		0.005		0.00025	
Mini-batch	50		200		50	
Dense Layers	1	4	1	1	2	1
Neurons	200		1 000		100	
Convolutional Layers	/			1	4	1
Filters	/			4, 8, 12, 16 or 32		
Kernel Size	/			5, 10, 20, 30 or 40		
Pool Type	/			Average or Max		
Activation	ReLU, ELU, or SELU					
Optimizer	Adam or RMSprop					
Weight Initializer	random_uniform, he_uniform, glorot_uniform, random_normal, he_normal, or glorot_normal					

Table 2 shows the success rate of the MOR approach for the ASCADf dataset using both distinguishers (loss and Pearson correlation). Moreover, we also report the results using both leakage models. We show the results using the training and validation set for distinguishing between the correct and the other key candidates. Z-score MSE, denoted by z-MSE, achieves excellent performance and is the best-performing loss function regardless of the key distinguisher, dataset used, or leakage model. z-MSE results are followed by the Z-score correlation (z-Corr) loss functions. z-MSE has at least $\approx 19\%$ higher success rate than all other loss functions. On the other hand, compared to the standard loss functions and their Z-score variants, z-MSE has at least $\approx 65\%$ higher success rate. The success rate with the ID leakage model is lower than when the HW model is utilized. Thus, we further increased the number of training traces to 40 000, with 10 000 validation traces, and obtained 85% SR for z-MSE on training and 86% for validation data.

We conducted the same experiments for the ASCADr dataset, and we provide the results in Table 3. As before, z-MSE achieves the best success rate in all tested scenarios. Considering the ASCADr datasets, z-MSE achieves at least $\approx 74\%$ higher success rate than other standard loss functions and their Z-score variants. Compared to correlation losses, it achieves a minimum 36% higher SR. The validation set concept improved SR for the z-MSE loss function in almost all cases. Other loss functions did not attain the same improvement using the validation set. However, using the Pearson correlation as the key distinguisher for the ASCADr dataset, using the validation set improved the SR for six out of eight tested functions. We mentioned that we achieve faster convergence using this key distinguisher, so we could be observing overfitting to the training data in these cases. Therefore, using a validation set mitigates this effect and selects the correct key byte, leading to better SR. We obtain the same success rate (SR) when using the correlation loss function with both distinguishers because, in this scenario, distinguishers essentially use the same function. The loss distinguisher minimizes inverted correlation, while the correlation distinguisher maximizes correlation function, ending in the same results. The functions of the loss and correlation distinguisher for when the z-Corr loss is used differ only in the Z-score normalization. Here, the SR results also stay the same, as the relations remain the same with and without the normalization.

Table 2: Success rate of MOR for the ASCADf dataset using both leakage models and different loss functions. The training set is 16 000, while the validation set is 4 000 when the HW leakage model (LM) is used, and 32 000 and 8 000, respectively, for the ID leakage model. We considered the loss and Pearson Correlation (Pearson) of validation (Vl) and training (Tr) set for the distinguishers.

Dstgh	LM	Set	MSE	z-MSE	MAE	z-MAE	Huber	z-Huber	Corr	z-Corr
loss	HW	Vl	15.68%	**94.95%**	8.14%	3.75%	16.29%	18.46%	72.8%	77.1%
		Tr	27.31%	**92.5%**	9.21%	6.25%	18.35%	8.54%	75.3%	77.5%
	ID	Vl	0%	**74.1%**	0.6%	0.3%	0%	0.8%	41.2%	44.0%
		Tr	25.7%	**77.6%**	35.8%	14%	36.1%	10.4%	45.4%	50.0%
Pearson	HW	Vl	45.9%	**94.9%**	37.7%	6.7%	48.5%	29%	72.8%	77.1%
		Tr	51%	**92.5%**	43.3%	10.1%	55.9%	35.7%	75.3%	77.5%
	ID	Vl	25.7%	**73.9%**	29.2%	22.8%	29.0%	13.7%	41.2%	44.0%
		Tr	37.5%	**77.6%**	43.4%	26.1%	44.5%	16.1%	45.4%	50.0%

Table 3: Success rate of MOR for the ASCADr dataset using both leakage models and different loss functions. The training set is 32 000, while the validation set is 8 000 with both leakage models (LM). We considered the loss and Pearson Correlation (Pearson) of validation (Vl) and training (Tr) set for the distinguishers.

Dstgh	LM	Set	MSE	z-MSE	MAE	z-MAE	Huber	z-Huber	Corr	z-Corr
loss	HW	Vl	3.9%	**91.2%**	1.5%	1.0%	2.2%	8.2%	44.7%	50.0%
		Tr	30.9%	**77.2%**	15.9%	9.0%	20.8%	12.2%	42.9%	44.6%
	ID	Vl	1.2%	**22.2%**	1.3%	0%	0%	0.4%	14.7%	13.8%
		Tr	4.5%	**9.4%**	2.6%	3.6%	0%	2.2%	6.9%	3.4%
Pearson	HW	Vl	36.6%	**90.2%**	28.4%	8.7%	39.0%	18.4%	44.7%	50.0%
		Tr	31.5%	**77.2%**	27.5%	8.7%	33.4%	21.6%	42.9%	44.6%
	ID	Vl	10.8%	**20.5%**	8.8%	3.9%	10%	3.5%	14.7%	13.8%
		Tr	4.3%	**9.4%**	3.5%	4.8%	2%	5.4%	6.9%	3.4%

The Z-score normalized MSE performs the best since it creates the most significant numerical gap between the correct and wrong key candidate bytes. Figure 2 shows loss values for each key candidate for four different Z-score normalized loss functions. The key candidate 224 corresponds to the third (correct) key byte for the ASCADf dataset.

The difference between the loss value corresponding to the correct key and the loss values for all the other key candidates is the largest for Z-score MSE. Moreover, the Z-score Correlation loss yields similar results to those of Z-score MSE, while both Z-score Huber and Z-score MAE result in a much smaller gap between loss values of correct and wrong key candidates, the smallest gap being obtained in the case of Z-score MAE. This is aligned with our experiments, as Z-score Huber performs better than Z-score MAE in terms of the success rate of the attacks (see Table 2 and Table 3). Z-score Correlation performs worse than Z-score MSE but significantly better than the other loss functions.

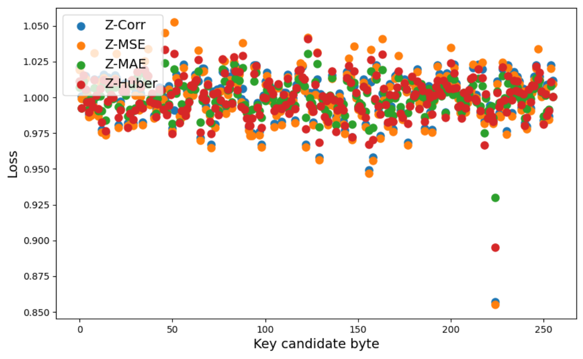


Figure 2: The normalized Z-score loss functions and their values for each key candidate. The loss values are scaled (divided by their mean).

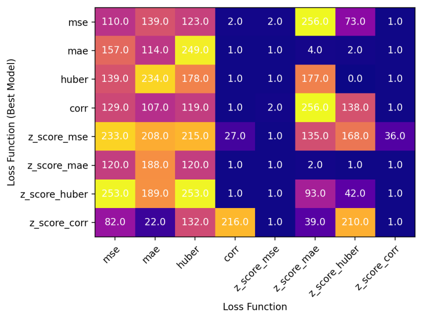
Considering the significant improvement in the observed success rate of the MOR-based SCA using z-MSE, we further test the adaptability of the functions using the following strategy:

1. We run random hyperparameter search processes for both `ASCADf_nopoi_10000` and `ASCADr_nopoi_25000` datasets. The hyperparameter search space is shown in Table 1. A separate random search is launched for each leakage model (ID or HW), model type (MLP or CNN), and loss function (defined in Section 4.1) combination. In total, we deploy 64 random search processes. In each of these searches, we try 100 different hyperparameter combinations, which sums up to 6 400 models. We conduct this search to assess the performance of diverse options properly.[5]
2. For each random search process, we select the best MOR model by sorting the models according to the objective function defined in Section 4.2. Experiments were executed using both distinguishers (loss function and Pearson correlation). This sorting procedure is done for each dataset, leakage model, model type, and loss function combination.
3. We re-train each of the best models with all other loss functions. The main idea is to see which loss function performs better across different hyperparameter configurations. The reported key rank is computed from a validation set described in Section 4.3, which are measurements not used during training. We opted to use the validation set as it improved the results for the best-performing loss function and the ASCADr dataset with the correlation distinguisher.

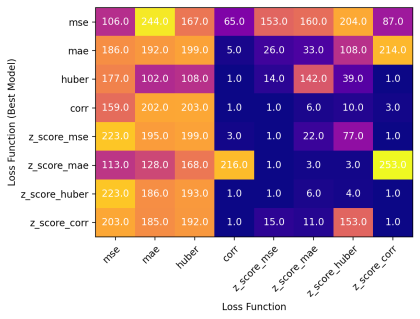
Figure 3 illustrates the *key rank* outcomes, derived using loss distinguisher defined with Eq. (9), of the top-performing model obtained with various loss functions represented on the y-axis. The optimal model is retrained with other loss functions depicted on the x-axis, and the key rank is presented in each cell. The darker colors in the graphs indicate better

[5]The attack commonly skips this procedure by selecting the loss function based on experience or shared knowledge and focuses on tuning other hyperparameters.

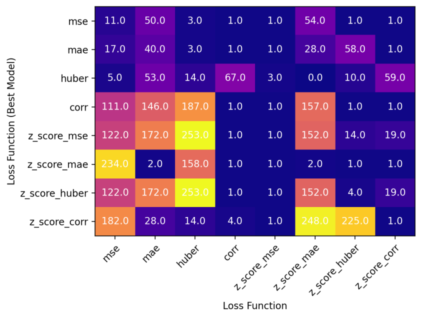
performance with lower key ranks. In these experiments, we use 40 000 traces for training and 10 000 for validation in the case of ASCADf, and 80 000 traces for training and 20 000 for validation in the case of ASCADr. The columns show that Z-Score MSE demonstrates superior adaptability across different hyperparameter configurations for various MOR models trained with other loss functions. Overall, the Z-score versions of the loss functions show greater adaptability than the standard loss functions.



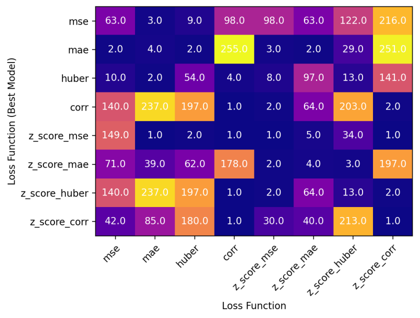
(a) `ASCADf_nopoi_10000`, HW



(b) `ASCADf_nopoi_10000`, ID



(c) `ASCADr_nopoi_25000`, HW



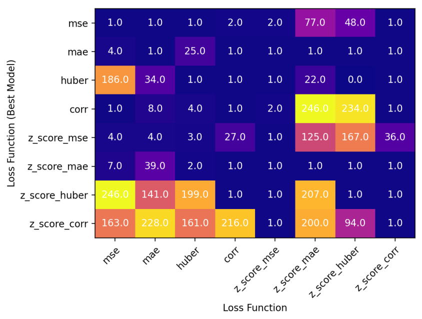
(d) `ASCADr_nopoi_25000`, ID

Figure 3: Loss function adaptation to different models. Each grid cell is the key rank result obtained with loss function distinguisher (Eq. (9)). The y-axis indicates the loss function set for finding the best model in a random search, while the x-axis indicates loss functions used during retraining.

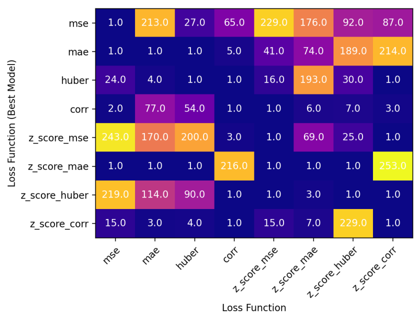
Figure 4 presents the key rank outcomes determined using a correlation distinguisher defined with Eq. (8), in which key candidates are ranked based on the highest correlation coefficient. We used the same number of training and validation traces as for the previous experiment. These findings emphasize that Z-Score MSE exhibits better adaptability to varying hyperparameter configurations when the HW leakage model is employed for both datasets. Conversely, when utilizing the ID leakage model with the `ASCADf_nopoi_10000` dataset, the performance of correlation and Z-Score MSE loss is comparable. The median and mean for correlation loss are 2 and 36.63, respectively, while for Z-Score MSE, the median is 8, and the mean is 38.13. For the `ASCADr_nopoi_25000` dataset with the ID leakage model, MSE, MAE, and Huber demonstrate slightly better results when averaged than Z-Score MSE. However, MORE adopts the Z-score MSE as its preferred loss function due to its significantly higher success rate when compared to other loss functions and its remarkable adaptability to various models.

4.4.2 MORE vs. MOR

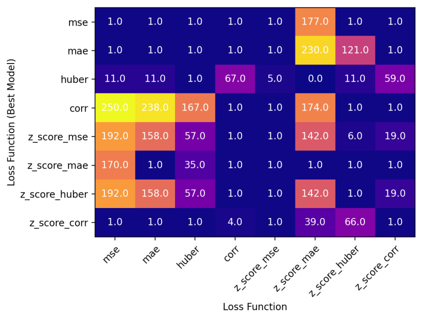
The main aspects of MORE that improve performance over the MOR models are



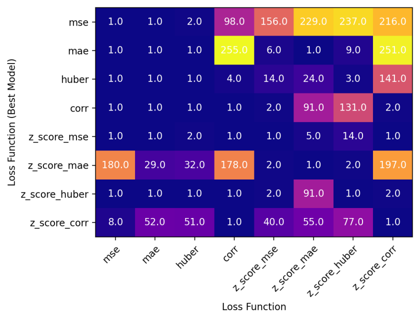
(a) `ASCADf_nopoi_10000`, HW



(b) `ASCADf_nopoi_10000`, ID



(c) `ASCADr_nopoi_25000`, HW



(d) `ASCADr_nopoi_25000`, ID

Figure 4: Loss function adaptation to different models. Each grid cell is the key rank results obtained with Pearson Correlation distinguisher (Eq. (8)). The y-axis indicates the loss function set for finding the best model in a random search, while the x-axis indicates loss functions used during retraining.

- Pearson correlation for the key distinguisher.
- Z-score normalized MSE as loss function.
- Validation set for finding the correct secret information.

Table 4: Average success rate comparison between MOR and MORE. For ASCADf and ASCADr, we used 16 000 training and 4 000 validation traces with the HW (HD) leakage model, and 32 000 training and 8 000 validation with ID. For AES_HD and AES_RD, we used 16 000 training and 4 000 validation traces for both leakage models since no further improvement was observed when increasing the number of traces.

	HW/HD		ID	
Dataset	MOR	MORE	MOR	MORE
ASCADf	27.3%	92.5%	25.7%	74.1%
ASCADr	33.9%	77.2%	4.5%	22.2%
AES_HD	10.2%	77.3%	11.3%	58.0%
AES_RD	22.01%	75.8%	0.7%	1.1%

We compare MOR and MORE methods using success rate calculated based on randomly chosen MLP models from the hyperparameter space presented in Table 1, where the optimizer is set to Adam optimizer, and the weight initialization method is fixed to *he_uniform*. Thus, the search space for MLP is reduced from 51 840 possible combinations compared to 4 320 by fixing the optimizer and weight initialization method. This is still a significantly larger search space than the one introduced in [DHD23] (192 possible combinations). Table 4 shows the success rate for both models on ASCADf, ASCADr, AES_HD, and AES_RD datasets with the ID and HW (HD) leakage models. Considering the HW (HD) leakage model, MORE presents at least 2.3× higher SR obtained for the ASCADr dataset, and, with the ID leakage model, it is at least 1.6× better, which is the result obtained on the AES_RD dataset, where both models obtain low SR. The best improvement with both leakage models is on the AES_HD dataset, with 7.6× and 5.1× higher SR using HW and ID leakage models, respectively. Overall, on average, MORE provides a 3.9× higher success rate.

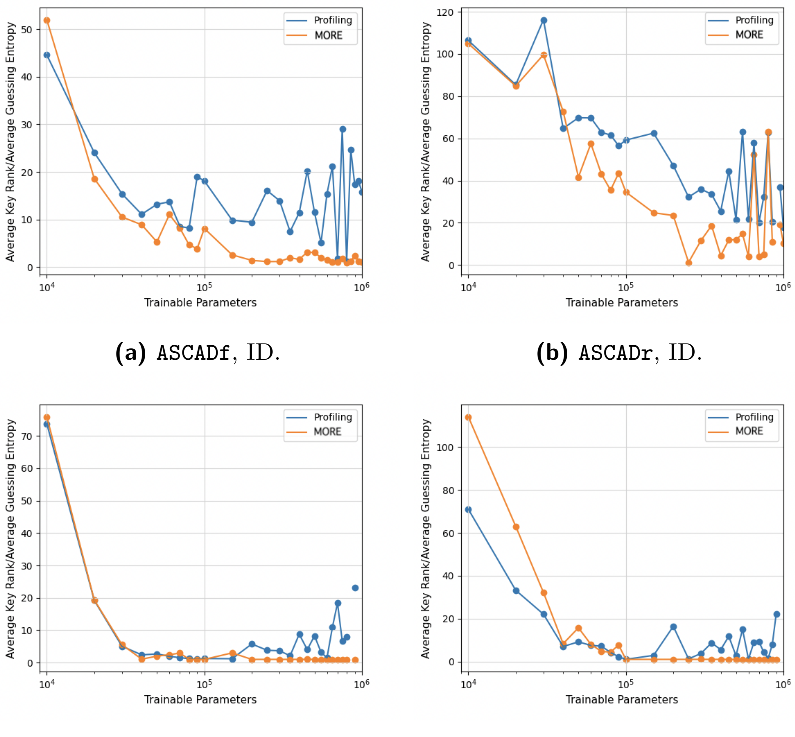
4.4.3 On the Size of the MORE Networks

In profiling SCA, smaller networks commonly work well enough on the public datasets most often used in DLSCA literature. A MORE-based neural network must address a multi-output regression problem. The trained model is designed to simultaneously predict 256 distinct outputs (each represented by a set of labels corresponding to each key byte candidate). Ultimately, this task is more complicated than profiling, where the attacker can access a labeled dataset. Therefore, we anticipate that the MORE-based network should have a more complex structure than neural networks typically considered for profiling settings. Moreover, that implies hyperparameter search spaces should be defined to provide networks with enough capacity to learn a more challenging task of non-profiling SCA.

In this section, we experimentally compare the performance of various neural networks for MORE (non-profiling) and profiling attacks. For each leakage model and model type (MLP or CNN), we randomly generate up to 3 000 neural network architectures, with trainable parameters ranging from 10 000 to 1 000 000, since the complexity (size) of neural network models can be measured in the number of trainable parameters when the same type of networks are compared, such as MLPs or CNNs. The hyperparameter search space is defined in Table 1, with fixed Adam optimizer and *he_uniform* initializer. However, the randomly generated MLP and CNN models can have up to eight hidden layers to enable the required number of trainable parameters. For CNN networks, we allow a maximum of four convolution layers and four fully connected layers. Each randomly generated architecture is trained as both MORE and a profiling model. The comparison criterion is the final guessing entropy (key rank) relative to the number of trainable parameters. For this analysis, we consider the ASCADf and ASCADr datasets. In the case of the ASCADf dataset in the profiling setup, 50 000 traces are considered for training, while 100 000 are utilized for the ASCADr dataset. For the attack, we used 5 000 attack traces. In the non-profiling case, for ASCADf, we use 40 000 for training and 10 000 traces for validation, while for ASCADr, we have 80 000 for training and 20 000 traces for validation.

Figure 5 displays the MLP performance of MORE and profiling models for both datasets and the two leakage models. The y-axis represents either the average key rank (for MORE) or the average guessing entropy (for profiling attack) for a specific range of trainable parameters. The results in this figure suggest that MORE-based models generally exhibit better performance when they include more trainable parameters. In contrast, the profiling task demonstrates an optimal range with enhanced key recovery performance. As models become excessively large (approaching 1 million trainable parameters) for a profiling task, overfitting signs emerge for the two examined datasets, which is not observed for MORE-based models, particularly visible with the HW leakage model. This implies that employing larger models for MORE tasks could be an appropriate choice. Consequently, models with fewer trainable parameters prove more efficient in a profiling attack, indicating that the structure of the profiling model could be easier to define, and more time can be spent on fine-tuning other hyperparameters if necessary.

Figure 6 presents the performance of MORE and profiling models using randomly generated CNNs. Once more, the results show that larger MORE-based CNN models outperform large CNN-based profiling models. This pattern is even more pronounced when considering the HW leakage model. Similar to the results with MLPs, the figure



(a) ASCADf, ID. **(b)** ASCADr, ID. **(c)** ASCADf, HW. **(d)** ASCADr, HW.

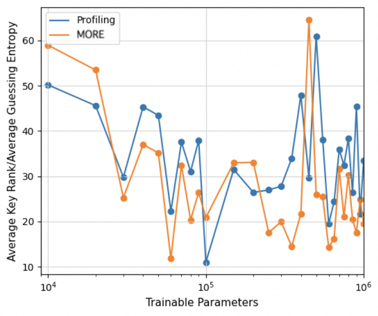
Figure 5: The performance of MORE and profiling models concerning the different number of trainable parameters in MLPs.

suggests that MORE-based CNN models for non-profiling tasks should have more trainable parameters than profiling models. To enable the creation of complex networks, the hyperparameter search space should be large enough to allow for such networks. Commonly, the search space for profiling SCA is small, as few layers provide enough network learning capacity to break the public datasets. Thus, following these experiments, we suggest using more extensive hyperparameter search spaces when tuning the networks for non-profiling SCA tasks, such as MORE-based SCA (original MOR and MORE).

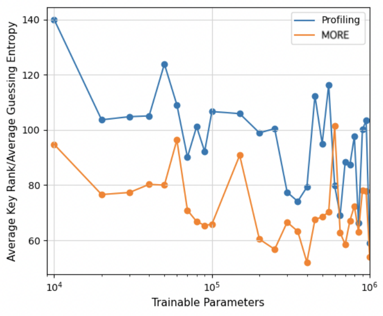
5 Getting More from MORE

While our experiments already show that MORE can significantly improve the attack performance over MOR, there is still a question of whether the performance can be further enhanced. Next, we will investigate two well-known concepts used in profiling deep learning-based SCA: ensembles and data augmentation. Both concepts were shown to improve the performance of profiling models significantly, e.g., [PCP20, CDP17, PCBP20]. Thus, using these methods, we expect to improve MORE-based non-profiling SCA further.[6]

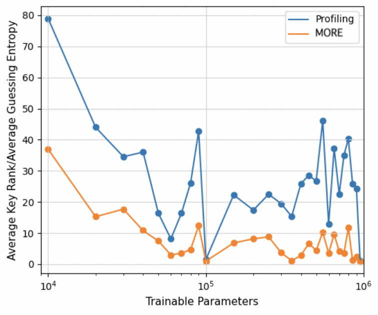
[6]Data augmentation is used as a vital part of a method proposed by Wu et al. [WPP23] by adding a layer to generate random perturbation right after the input, while ensembles were not yet explored for non-profiling SCA, to the best of our knowledge.



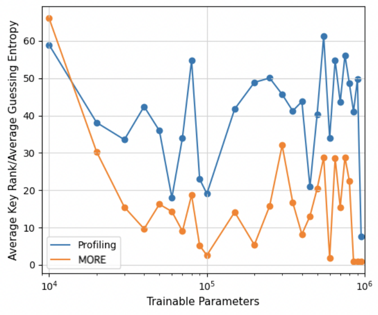
(a) `ASCADf`, ID.



(b) `ASCADr`, ID.



(c) `ASCADf`, HW.



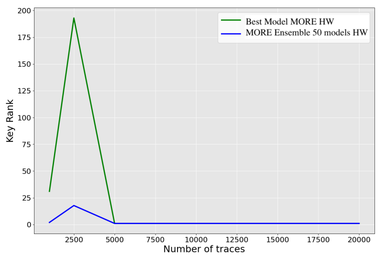
(d) `ASCADr`, HW.

Figure 6: The performance of MORE and profiling models concerning the different numbers of trainable parameters in CNNs.

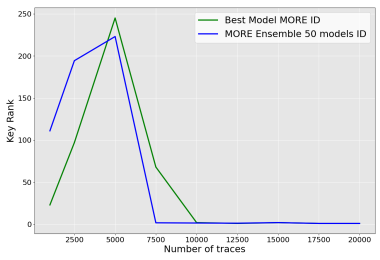
5.1 MORE Ensembles for SCA

This section provides experimental results for the bootstrap aggregating ensemble methodology [Bre96] with MORE approach. Bootstrap aggregating (also known as bagging) is an ensemble technique in machine learning that combines multiple models to improve performance and reduce overfitting. The idea is to create multiple sub-samples (bootstraps) of the original dataset chosen randomly, train a separate model on each sub-sample, and then combine the predictions of the individual models. We consider the ID and the HW/HD leakage models for ASCADf, ASCADr, AES_RD, and AES_HD datasets. Based on the preliminary tuning phase, we set the number of epochs to 40 and varied the number of available traces for the MORE model. From the available traces, 80% are training traces, while the remaining 20% are validation traces. The key rank is calculated based on the different number of input traces and shown in the graphs. We test a single best MORE model and ensembles comprising 50 models. Each model is randomly selected from the same hyperparameter space detailed in Table 1, except for the optimizer being set to Adam and the weight initialization set to *he_uniform*. The number of models in an ensemble is decided based on the preliminary experiments with different numbers of models for the ensemble. We do not show results for CNN ensembles as we could not observe any significant improvements when using more than one model. We compare the ensemble's key rank to the key rank of the best model selected according to the highest value of the Pearson correlation objective function, as introduced in Section 4.

In Figure 7a, we plot the key rank for ASCADf with the HW leakage model while varying the number of traces. The best model chosen according to the Pearson correlation distinguisher displays higher key ranks with fewer traces than the ensemble. However, both methods perform similarly when the number of traces exceeds 5 000. Figure 7b demonstrates the key rank for the ID leakage model. In this case, the ensemble recovers the secret key more quickly, using fewer traces, than the single best model. Similarly to the results on ASCADf, ensembles on ASCADr for the HW leakage model show slightly better performance when less than 10 000 traces are used, while with more traces, the single best model and ensemble consistently retrieve the correct key. We observed no improvement for the ID leakage model using ensembles due to the low success rate we obtained. We omit these results due to lack of space and similar results.



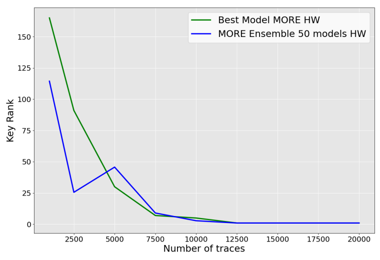
(a) `ASCADf`, Key rank, HW leakage



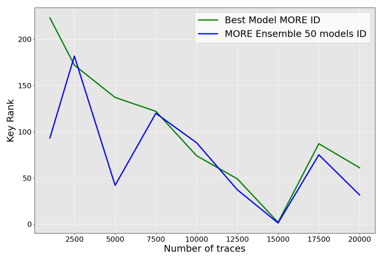
(b) `ASCADf`, Key rank, ID leakage

Figure 7: Key rank for ASCADf for the HW and ID leakage models.

Figure 8 shows the key rank of the ensemble and the best model for AES_RD. As seen in Figure 8a, the ensemble performs slightly better than the best model using fewer traces. Figure 8b shows that overfitting for ID is not entirely avoided, not even when using ensembles. In this case, there was no significant benefit when employing the ensembles. The performance based on the key rank is similar between the two setups.



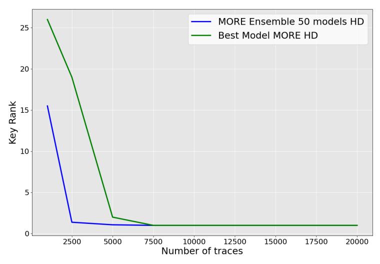
(a) `AES_RD`, Key rank, HW leakage



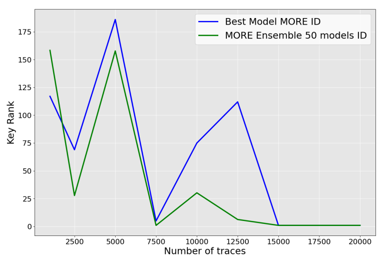
(b) `AES_RD`, Key rank, ID leakage

Figure 8: Key rank for AES_RD for the HW and ID leakage models.

Figure 9 considers MORE experimental setup for AES_HD and plots the key ranks for the HD and ID leakage models varying the number of traces. The MORE ensemble shows improvement for both leakage models compared to the best model. It reaches the minimum key rank faster in the case of the HD leakage model and decreases the key rank with the ID leakage model when less than 15 000 traces are used. The effect visible with the ID leakage model comes from retraining the models for the different number of available traces we tested and a rather noisy data.



(a) `AES_HD`, Key rank, HD leakage



(b) `AES_HD`, Key rank, ID leakage

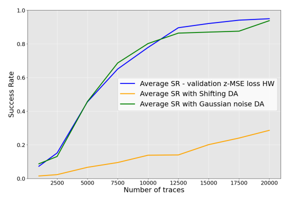
Figure 9: Key rank for AES_HD for the HD and ID leakage models.

5.2 Data Augmentation with MORE-based SCA

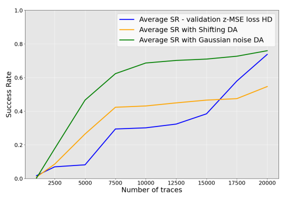
Insufficient data in a non-profiling side-channel analysis setting can lead to an unsuccessful attack. A possible reason is the overfitting that can easily occur, resulting in an unreliable distinguishing phase between key candidates. This section presents results for the MORE methodology with data augmentation (DA). We employ data augmentation on the best model chosen based on the highest objective function and the Z-score MSE for a validation set (MORE approach). We utilized random shifting as the first data augmentation technique, which involves applying a random time shift to each trace. The shift amount can be positive or negative and can be drawn from a uniform or normal distribution. In this work, we opted for a shift of five samples and a uniform distribution. The second data augmentation approach we employed is Gaussian noise, which applies a small perturbation to each sample of the trace. The perturbation value is drawn from a standard normal distribution (mean 0 and standard deviation 1). For both approaches, we augment the original traces by adding 10 000 more traces or double the number of traces if the original set is smaller than 10 000. This setting is selected based on the preliminary experiments.

Figure 10a displays SR for ASCADf, AES_HD, and AES_RD with and without data augmentation for 500 randomly chosen neural networks from the hyperparameter space defined in Table 1 (optimizer set to Adam, and initialization method to *he_uniform*) with MORE methodology. We observe that Gaussian noise DA slightly helps the success rate, especially when the number of traces before augmentation is less than 10 000. However, when shifting DA is employed, the success rate for all initial numbers of traces is rather low. On the other hand, in the case of AES_HD, we see that both approaches show a significant SR increase, especially for Gaussian noise, as can be seen in Figure 10b. For AES_RD, results in Figure 10c show that both techniques can help when the number of input traces is smaller than 10 000. We notice that SR can significantly decrease when using MORE with data augmentation. We believe this is because data augmentation reduces overfitting, which can quickly happen in the MORE context. Although we try to minimize this effect for MORE by using a validation set rather than the training set, as in the case of MOR, MORE is still susceptible to overfitting.

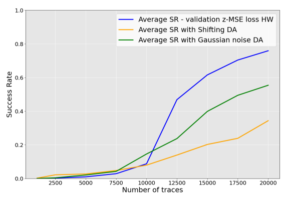
Although there is no substantial improvement in the success rate of MORE with the application of data augmentation, it is worth noting that the average guessing entropy can significantly decrease when DA is employed for the best MORE model. We trained the best MORE model 100 times with 5 000 traces before augmentation and compared the average GE with the average GE of the same model when using data augmentation. Figure 11a illustrates that Gaussian noise data augmentation yields better results compared to shifting, leading to a slight reduction in the overall guessing entropy across 40 training epochs. It is worth noting that this DA technique also yields the best results for AES_HD where after just five training epochs, the guessing entropy drops below ten, as indicated in Figure 11b.



(a) `ASCADf`, SR varying the number of traces before augmentation



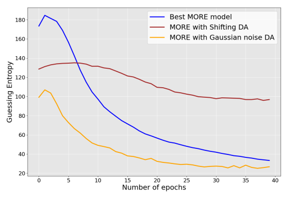
(b) `AES_HD`, SR varying the number of traces before augmentation



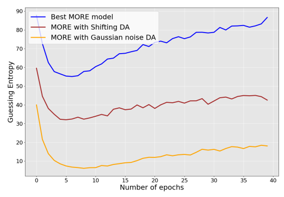
(c) `AES_RD`, SR varying the number of traces before augmentation

Figure 10: Success rate for MORE on ASCADf, AES_HD, and AES_RD for the HW/HD leakage model.

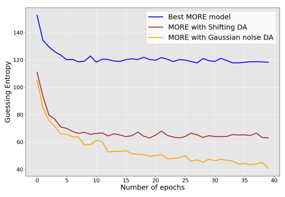
In the case of AES_RD, shifting DA appears to be even more effective in reducing GE, as seen in Figure 11c, but again, both DA techniques provide improvement. We note that DA methods on ASCADr obtained lower SR than without using DA for both HW and ID leakage models. As explained here, the experiments comparing GE of the best model with and without DA show that GE within the first 40 epochs is the lowest without using DA methods. We omit the figures with results due to space.



(a) `ASCADf`



(b) `AES_HD`



(c) `AES_RD`

Figure 11: Guessing entropy for MORE on ASCADf, AES_HD, and AES_RD, varying the number of epochs for the HW/HD leakage model with data augmentation.

6 Conclusions and Future Work

This work provides a novel outlook on the multi-output regression (MOR) SCA. We proposed MORE, an approach based on MOR that significantly improves the attack performance by having a 1) more powerful loss function, 2) stronger distinguisher, and 3) a better, more stable overfitting mitigation procedure. Moreover, we further improved MORE with ensembles and data augmentation, where both approaches show significant improvements over "vanilla" MORE when a smaller amount of data is available. Overall, the combination of the MORE methodology, ensemble methods, and data augmentation presents a promising approach for enhancing non-profiling DLSCA performance and improving the reliability of distinguishing between key candidates.

Future research could explore other data augmentation techniques and ensemble strategies to optimize the performance of MORE-based models further. Additionally, incorporating advanced regularization techniques may help address overfitting and improve the generalization capabilities of the models. Finally, this work considered synchronized datasets while investigating the desynchronization countermeasure could be an interesting research direction. We believe fine-tuning the MORE models could be an efficient option for such settings.

References

[APB+20] Amir Alipour, Athanasios Papadimitriou, Vincent Beroulle, Ehsan Aerabi, and David Hély. On the performance of non-profiled differential deep learning attacks against an aes encryption algorithm protected using a correlated noise generation based hiding countermeasure. In *Proceedings of the 23rd Conference on Design, Automation and Test in Europe*, DATE '20, page 614–617, San Jose, CA, USA, 2020. EDA Consortium.

[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.

[BDMS22] Olivier Bronchain, François Durvaux, Loïc Masure, and François-Xavier Standaert. Efficient profiled side-channel analysis of masked implementations, extended. *IEEE Trans. Inf. Forensics Secur.*, 17:574–584, 2022.

[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.

[Bre96] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[BVBL15] Hanen Borchani, Gherardo Varando, Concha Bielza, and Pedro Larranaga. A survey on multi-output regression. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(5):216–233, 2015.

[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.

[CK09] Jean-Sébastien Coron and Ilya Kizhvatov. An Efficient Method for Random Delay Generation in Embedded Software. In *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, pages 156–170, 2009.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template Attacks. In *CHES*, volume 2523 of *LNCS*, pages 13–28. Springer, August 2002. San Francisco Bay (Redwood City), USA.

[DHD23] Ngoc-Tuan Do, Van-Phuc Hoang, and Van Sang Doan. A novel non-profiled side channel attack based on multi-output regression neural network. *Journal of Cryptographic Engineering*, pages 1–13, 2023.

[DHDP22] Ngoc-Tuan Do, Van-Phuc Hoang, Van Sang Doan, and Cong-Kha Pham. On the performance of non-profiled side channel attacks based on deep learning techniques. *IET Information Security*, n/a(n/a), 2022.

[DLH+22] Ngoc-Tuan Do, Phu-Cuong Le, Van-Phuc Hoang, Van-Sang Doan, Hoai Giang Nguyen, and Cong-Kha Pham. Mo-dlsca: Deep learning based non-profiled side channel analysis using multi-output neural networks. In *2022 International Conference on Advanced Technologies for Communications (ATC)*, pages 245–250, 2022.

[HDD22] Van-Phuc Hoang, Ngoc-Tuan Do, and Van Sang Doan. Efficient non-profiled side channel attack using multi-output classification neural network. *IEEE Embedded Systems Letters*, pages 1–1, 2022.

[KFYF21] Kunihiro Kuroda, Yuta Fukuda, Kota Yoshida, and Takeshi Fujino. Practical aspects on non-profiled deep-learning side-channel attacks against aes software implementation with two types of masking countermeasures including rsm. In *Proceedings of the 5th Workshop on Attacks and Solutions in Hardware Security*, ASHES '21, page 29–40, New York, NY, USA, 2021. Association for Computing Machinery.

[KHK22] Donggeun Kwon, Seokhie Hong, and Heeseok Kim. Optimizing implementations of non-profiled deep learning-based side-channel attacks. *IEEE Access*, 10:5957–5967, 2022.

[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.

[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.

[PCBP20] Guilherme Perin, Łukasz Chmielewski, Lejla Batina, and Stjepan Picek. Keep it unsupervised: Horizontal attacks meet deep learning. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(1):343–372, Dec. 2020.

[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.

[PPM$^+$22] Stjepan Picek, Guilherme Perin, Luca Mariot, Lichao Wu, and Lejla Batina. Sok: Deep learning-based physical side-channel analysis. *ACM Comput. Surv.*, oct 2022. Just Accepted.

[PS15] SGOPAL Patro and Kishore Kumar Sahu. Normalization: A preprocessing stage. *arXiv preprint arXiv:1503.06462*, 2015.

[PWP22] Guilherme Perin, Lichao Wu, and Stjepan Picek. Exploring feature selection scenarios for deep learning-based side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(4):828–861, Aug. 2022.

[RAD20] Keyvan Ramezanpour, Paul Ampadu, and William Diehl. SCARL: side-channel analysis with reinforcement learning on the ascon authenticated cipher. *CoRR*, abs/2006.03995, 2020.

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

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

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

[WMZT20] Qi Wang, Yue Ma, Kun Zhao, and Yingjie Tian. A comprehensive survey of loss functions in machine learning. *Annals of Data Science*, pages 1–26, 2020.

[WP20] Lichao Wu and Stjepan Picek. Remove some noise: On pre-processing of side-channel measurements with autoencoders. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(4):389–415, 2020.

[WPP22] Lichao Wu, Guilherme Perin, and Stjepan Picek. I choose you: Automated hyperparameter tuning for deep learning-based side-channel analysis. *IEEE Transactions on Emerging Topics in Computing*, pages 1–12, 2022.

[WPP23] Lichao Wu, Guilherme Perin, and Stjepan Picek. Hiding in plain sight: Non-profiling deep learning-based side-channel analysis with plaintext/ciphertext. Cryptology ePrint Archive, Paper 2023/209, 2023. https://eprint.iacr.org/2023/209.

[WWK$^+$23] Lichao Wu, Léo Weissbart, Marina Krček, Huimin Li, Guilherme Perin, Lejla Batina, and Stjepan Picek. Label correlation in deep learning-based side-channel analysis. *IEEE Transactions on Information Forensics and Security*, 18:3849–3861, 2023.

[XST$^+$19] Donna Xu, Yaxin Shi, Ivor W Tsang, Yew-Soon Ong, Chen Gong, and Xiaobo Shen. Survey on multi-output learning. *IEEE transactions on neural networks and learning systems*, 31(7):2409–2429, 2019.