Too Hot To Be True: Temperature Calibration for Higher Confidence in NN-assisted Side-channel Analysis

Seyedmohammad Nouraniboosjin and Fatemeh Ganji

Worcester Polytechnic Institute, Worcester, USA
snouraniboosjin@wpi.edu,fganji@wpi.edu

Abstract. The past years have witnessed a considerable increase in research efforts on neural network-assisted profiled side-channel analysis (SCA). At the same time, studies have identified challenges, including closing the gap between machine learning (ML) classification metrics and side-channel attack evaluation. In fact, in NN-assisted SCA, the NN's output distribution forms the basis for successful key recovery. In this respect, prior work has studied many aspects of integrating NNs into SCA, including model selection, training, and hyperparameter tuning. Nevertheless, a well-known fact has been largely overlooked in the SCA-related literature, namely NNs' tendency to become over-confident, that is, assigning overly high probability to the correct class (the secret key in the sense of SCA). Temperature[1] scaling is a powerful remedy for this behavior. From the perspective of deep learning, temperature scaling does not affect NN accuracy; however, its impact on secret-key recovery metrics, mainly guessing entropy, is worth investigating. This paper reintroduces temperature scaling into SCA and demonstrates that key recovery can become more effective. Importantly, temperature scaling can be integrated into SCA without re-tuning the network. In doing so, temperature can be treated as a metric to assess NN performance before launching the attack. In this regard, we study the impact of hyperparameter tuning, network variance, and capacity, and derive recommendations to prevent miscalibration and overconfidence.

Keywords: Profiled Side-channel Analysis, Neural Network, Temperature Calibration, Confidence, Metrics

1 Introduction

Side-channel analysis (SCA) refers to methods that exploit vulnerabilities in the implementation of cryptographic algorithms rather than the underlying protocols [32]. When executing the algorithm, such vulnerabilities are caused by the secret-key leakages, observable in the form of power consumption, timing,

[1] The term "temperature" refers to a scalar parameter in deep learning and should not be confused with physical heat.
The code is available at `https://github.com/vernamlab/CoolSCA`.

and electromagnetic emissions [27,26,14,42]. Profiling SCA, which is more common thanks to its effectiveness [5], refers to a stronger attacker with access to an open copy of the targeted device. This copy is utilized by the attacker to construct a device profile, facilitating attacks on another device of the same type [12,20,28,33]. Profiling attacks, involving a profiling phase followed by an attack phase, are thus known as two-stage attacks.

In recent years, the focus has been on profiling attacks leveraging machine learning (ML) [29,19,20], particularly deep learning [40]. These attacks are notably capable of breaking implementations equipped with countermeasures against SCA [28,15,11,39,25,48]. However, numerous questions have remained unresolved. As a prime example, related work has highlighted that performance evaluation in NN-assisted SCA is still an open problem [21]. On one hand, when testing/validating an NN, its performance is evaluated by employing classification metrics for ML, such as accuracy, loss, and recall. SCA, on the other hand, has often been assessed by considering the guessing entropy (GE) and success rate (SR) [46], which have been observed to be in conflict with classification metrics [39]. To narrow this gap, studies have come up with new metrics [53,21] or followed a more systematic approach to understand the root cause of this discrepancy. In the latter category, [38] has visually shown how output class probabilities are ranked for both successful and unsuccessful attacks under the condition where accuracy is not enough to make a decision about the attack's success. In fact, the accuracy can be low or close to a random guessing value, whereas expected classes are among the first ones. In this case, in accordance with the definition of GE, the summation of the probabilities of those expected classes (based on the label's guessing for the correct key) explains the success of attacks cf. [40].

What can be understood from this discussion is that, obviously, the attack's performance relies heavily on the output class probabilities. These probabilities depend on the network configuration, parameters (weights and biases), hyperparameters, loss functions, and the output function. First, determining NN configurations and hyperparameters to break the target is a challenging task tackled in the literature [50,47,1,48,44]. Besides the importance of hyperparameter tuning, choosing the output function needs to be better advised in the literature than others. Softmax output function, as the last layer of a NN, is the most common choice since it supports multi-class classification mimicking the profiled SCA. In order to prevent overflow (i.e., exploding gradient problem) as observed in ML- and SCA-related studies [17,24], the log-softmax is usually preferred cf. [10], whereas [1] has suggested numerically stable softmax [17].

The selection of the output layer is closely related to the choice of the loss function. Regarding principles of deep learning, NNs for classification that use a softmax function in the output layer learn faster and more robustly using the negative log-likelihood function (NLL) [17]. This aspect is well explored in SCA-related literature. It has been proven that minimizing the NLL function (similarly, cross-entropy) during training is asymptotically equivalent to maximizing the estimation of mutual information between the side-channel traces

and the leakage profiling model (i.e., the ML trained on the traces) [34]. This has also been empirically verified in [24].

Temperature calibration. Setting all the configuration details aside, it is known that the NNs tend to be "too confident" when predicting the classes [18,35]. Here, confidence means the probability of the correctness of the prediction (e.g., softmax probability for the predicted class). In deep learning-related literature, methods have been developed to *calibrate* the confidence, i.e., going closer to the true probability. One simple and common calibration factor is called the *temperature*, T, which "softens" the softmax in a way that $T \to 1$ indicates the estimated output probability is close to its true probability cf. [18].

It is well known that the NN's accuracy is invariant to the temperature, but what about GE? GE is the average position of the correct key in a key guessing vector built upon the output of softmax. In fact, temperature calibration improves confidence and scales the class prediction up/down; hence, it is expected to observe an improvement in GE. The extent of this and the conditions for such improvements are studied in this paper.

Contributions. This paper focuses on applying temperature calibration in NNs trained for profiling SCA.

- We analyze the effect of temperature calibration on attack performance in terms of guessing entropy (GE). We consider Platt's multi-class calibration [18], non-optimized vector scaling (stable softmax), and their combination (see Section 3). Using publicly available neural network models from benchmark SCA studies, we show that calibrating miscalibrated output distributions improves GE, allowing the target to be recovered with fewer traces.
- We study how hyperparameter tuning affects model performance through the lens of temperature. We show that poor search spaces or search strategies can prevent successful attacks, even with suitable objective functions, while different objectives may yield similar results under the same tuning setup. Unlike prior work that evaluates attack performance mainly via guessing entropy (GE), we demonstrate that temperature provides an indicator of how well a model is configured and trained *before launching the attack*, with improperly tuned models exhibiting high temperature.
- Another key factor in neural network calibration is variance, which reflects sensitivity to the training data. Since training aims to reduce variance, regularization or increasing the amount of training data is commonly recommended. For side-channel traces, we analyze how the number of training and validation traces affects calibration, and show that using more traces leads to better-calibrated models with lower temperatures.
- Finally, drawing on lessons from machine learning such as temperature scaling, we provide recommendations on model evaluation for SCA. In line with prior work, we argue that accuracy is not an appropriate metric to assess attack effectiveness. We also emphasize the importance of limiting neural network capacity, as these guidelines mitigate miscalibration and lead to more effective attacks.

1.1 Related work

This section gives a brief overview of the literature devoted to specific topics in NN-assisted SCA that are relevant to the scope of our paper, namely metrics in SCA and hyperparameter tuning.

Metrics in SCA. Compared to template attacks, which are information-theoretically optimal given a sufficiently large number of traces, NN-assisted profiled SCA using multi-layer perceptrons (MLPs), convolutional neural networks (CNNs), and stacked autoencoders can exhibit similar or superior performance [40]. Clearly, such comparisons require appropriate metrics. The differences between ML metrics and side-channel metrics have been investigated from several angles. Picek et al. report that class imbalance can contribute to inconsistencies between, for example, accuracy and attack performance [39]. This gap can be more pronounced for highly noisy traces and traces collected from protected implementations. To examine whether guessing entropy (GE) is an appropriate evaluation metric,[38] visually analyzes the output class probabilities. They observe that accuracy can be low, or close to random guessing, because the expected class is not always ranked first but often appears among the top candidates. Hence, summing these probabilities for each key-byte candidate yields a valid distinguisher, consistent with the definition of GE. Following this line of thought, the cross entropy ratio (CER) metric has been introduced and is closely related to GE and success rate (SR)[53]. CER is reported to improve attack performance, in particular under imbalanced training and test datasets. Note that here our focus is on common attack-performance metrics; therefore, other SCA-related metrics, such as perceived information (PI) and its extensions [43,9,21], are not discussed.

Hyperparameter tuning. A critical importance in the profiling process is attributed to selecting the most effective model configuration and its hyperparameters, so-called hyperparameter tuning. In this regard, Benadjila et al. explored the significance of hyperparameter tuning in their study and provided proposals to help researchers choose an appropriate set of hyperparameters [7]. Zaid et al. [50] introduced a visualization-based approach for selecting hyperparameters concerning the convolutional part (e.g., the number of filters) in CNNs. In doing so, an architecture with a minimized complexity can be figured out, cf. [50]. Building on Zaid's research, Wouters et al. demonstrated achieving comparable attack performance using smaller neural network designs [47]. Hyperparameter tuning has been made more automated in [48], where different objectives and search methods can be selected to find the best hyperparameter. More specifically, they examined the application of layer-level network morphism and Bayesian optimization integrated into Auto-Keras [23], their algorithm's core. The layer-level network morphism modifies a trained neural network to make a new architecture by employing different operations, e.g., inserting a layer or adding a skip-connection between layers, although at the cost of possible overfitting [48]. As another example, [44] has applied reinforcement learning to determine CNNs that are small (in terms of the number of trainable parameters), but exhibit good attack performance. Nevertheless, the configuration is

still (to some extent) guided by the expert through providing a random range of the hyperparameters' values. Recently, [1] has introduced InfoNEAT, a framework to not only select the configuration of an MLP-like model, but also tune its hyperparameters, including the number of epochs. Thanks to the irregular NN configuration automatically evolved by InfoNEAT, the networks are shift-invariant, i.e., training on a device and testing on a similar one protected by desynchronization.

Summary. As reviewed above, guessing entropy (GE) is a common metric for NN-assisted SCA and is computed from the NN's output probabilities. Hyperparameter tuning and training therefore affect GE by shaping the output distribution. In deep learning, confidence is used to quantify how well these probabilities reflect true correctness, and temperature scaling is a post hoc method to improve this calibration after training. In the context of SCA, this provides a mechanism to mitigate overconfidence and can improve attack performance, addressing a gap that has been largely overlooked in the SCA literature.

2 Background

2.1 Notations

In this paper, sets are represented using calligraphic letters such as $\mathcal{X}$, while random variables are denoted by the corresponding upper-case letter X. Realizations of X are indicated by the corresponding lower-case letter x. Moreover, bold letters (e.g., $\mathbf{y}$) correspond to matrices and vectors. We use the standard notations for mathematical operators defined in the respective sections.

2.2 Profiled Side-channel Analysis

A profiled SCA is characterized by two phases: profiling and attack [12]. During the profiling phase, the adversary utilizes an open device that she can control to build a *profiling model* to extract the encryption key from similar devices. These two phases match the training and testing steps in the ML domain.

In order to build the profiling model, the adversary has access to guessable or public inputs: a chunk of plaintext P, as well as a part of the cryptographic algorithm's secret key S that the attacker aims to recover. Giving these inputs to the device, the adversary observes an estimation $\hat{\varphi}_s$ of the conditional probability distribution function for every possible $s \in \mathcal{S}$ as follows [5].

$$\varphi_s : (\mathbf{x}, s) \mapsto \Pr[\mathbf{X} = \mathbf{x} \mid (P, S) = (p, s)].$$

This means that the side-channel traces can be used to estimate $\hat{\varphi}_s$. In doing so, for a given profiling set of $\{p_i, s_i\}_{i=1}^{n}$, the adversary collects n traces $\{x_1^i, x_2^i, \cdots, x_d^i\}_{i=1}^{n}$, where each trace contains d features ($d \geq 2$). Here d denotes the dimensionality of each trace. The adversary constructs an ML model (i.e., a leakage model) using the profiling set, which can estimate the probability of inputs for every $s \in \mathcal{S}$ as: $\hat{\varphi}_{X,P} : (\mathbf{x}, p) \mapsto \Pr[(P, S) = (p, s)|\mathbf{X} = \mathbf{x}]$. To launch

the attack, the adversary aims to classify a set of N_{test} traces, the so-called test set corresponding to an unknown s, based on the profiling model. Similar to testing in an ML classification task, the adversary should derive the label for a trace: $\mathbf{y} = \hat{\varphi}_{X,P}(\mathbf{x}, p)$, for $\hat{s} \in \mathcal{S}$ so that $\hat{s} = \arg\max_{s \in \mathcal{S}} \mathbf{y}_k$, where $\mathbf{y}_k$ is the k^{th} entry in the vector $\mathbf{y}$.

Afterward, a *score* based on the maximum-likelihood of each hypothetical key can be obtained for N_{test} traces, as $\mathbf{d}_k = \prod_{i=1}^{N_{test}} \mathbf{y}_k^i$, where $\mathbf{y}_k^i$ is the k^{th} entry in the vector $\mathbf{y}^i$ corresponding to the i^{th} trace. With regard to this score, the key hypotheses are ranked in a decreasing order based on the rank function (Equation (1)), from which the adversary selects the key that is ranked first. The rank function is defined as [5]:

$$Rank(\hat{\varphi}, N_{test}) = |\{k \mid \mathbf{d}_k > \mathbf{d}_{k^*}\}|, \tag{1}$$

where k^* represents the key used to acquire the attack traces. The rank is calculated for a collection of N_{test} traces from the test dataset, where N_{test} is increased gradually until the rank is minimized (the lower the rank, the higher the score). It is common to compute the rank over different chunks of datasets. The *average rank*, also called the *GE* [31], is calculated as the mean of rank over different chunks. This paper also reports T_{GE0} denoting the least number of attack traces required to break the target [44,1].

Leakage models. SCA commonly follows a divide-and-conquer strategy, for example recovering AES sub-key bytes. Accordingly, k^* in Equation 1 can denote a sub-key, and the procedure is repeated across sub-keys to recover the full key; in the literature, recovering one sub-key is often considered sufficient to assess vulnerability [40]. For each sub-key, a leakage model maps hypothetical intermediate values to expected leakage. Under the identity (ID) model, the intermediate value yields 256 classes. Under the Hamming-weight (HW) model, leakage is assumed proportional to the number of ones in the intermediate value.

2.3 Some Relevant Concepts in ML

Negative log-likelihood (NLL) loss. In multi-class classification, neural networks are typically trained using the categorical cross-entropy loss, also known as the negative log-likelihood (NLL) [36]. For a single input trace $\mathbf{x}$, let $\mathbf{y} \in \{0,1\}^C$ denote the one-hot encoded true label vector over C classes, and let $\hat{\mathbf{y}} \in [0,1]^C$ denote the predicted probability distribution after the softmax layer. The NLL loss for one sample is defined as $L(\mathbf{x}) = -\sum_{m=1}^{C} y_m \log \hat{y}_m$. Since $\mathbf{y}$ is one-hot encoded, i.e., $y_{k^*} = 1$ for the correct class k^* and $y_m = 0$ otherwise, the loss simplifies to $L(\mathbf{x}) = -\log \hat{y}_{k^*}$. The total loss over a dataset is obtained by averaging (or summing) this quantity over all samples.

Numerically stable softmax. In practice, NLL is often coupled with the softmax output layer. Softmax function $\sigma_{SM} : \mathbb{R}^C \mapsto [0,1]^C$ is formulated as

$$\sigma_{SM}(\mathbf{z}_i)^{(c)} = \frac{\exp\left(z_i^{(c)}\right)}{\sum_{j=1}^{C} \exp\left(z_i^{(j)}\right)}.$$

Softmax may suffer from overflow issues (i.e., exploding gradient problem) as observed in ML- and SCA-related studies [17,24]. As a remedy, a numerically stable softmax (hereafter called stable softmax) has been proposed in [17], where $\mathbf{w}_i^{(m)} = \mathbf{z}_i^{(m)} - \max_m \mathbf{z}_i^{(m)}$ substitutes for $\mathbf{z}_i^{(m)}$ in the softmax formula.

Validation. Before testing the trained model, it is essential to ensure that it is effectively generalized to new, unseen data. Therefore, cross-validation methods are frequently used. Cross-validation is a statistical method for evaluating the effectiveness of ML models. It involves training the model on a portion of the dataset and using a different, disjoint part of the *training* dataset to evaluate its performance. In ML and SCA, the hold-out validation technique is usually applied. The hold-out validation technique (hereafter called validation) involves partitioning the dataset into training, validation, and test datasets. The model is trained using the training dataset, and its performance is evaluated using the validation dataset, holding out of the training dataset. The most effective model is then applied to the test dataset.

2.4 Datasets

This paper focuses on benchmark datasets commonly used to evaluate NN-based SCA, namely CHES-CTF and two versions of ASCAD [4]. ASCAD targets an 8-bit AVR microcontroller running a masked AES-128 implementation, where electromagnetic emanation is the observed side-channel [6]. ASCAD-f contains 50,000 profiling traces and 10,000 attack traces, targets the third key byte (the first masked byte), uses a 700-feature window, and uses the same key for profiling and attack traces [2]. ASCAD-r contains 200,000 profiling traces with random keys and 100,000 attack traces with a fixed key, targets the same byte, and uses a 1400-feature window [3].

We also use the CHES-CTF dataset [45], released for the CHES 2018 capture-the-flag event. It includes 45,000 profiling traces and 5,000 attack traces, where both phases use fixed keys (with different fixed keys across profiling and attack). Each trace has 2200 features and is captured from a masked AES-128 implementation on a 32-bit STM microcontroller.

3 Temperature Scaling for SCA

In systems where decisions are critical, it is not enough for classification networks to be precise; they also need to indicate potential inaccuracies. For example, consider an autonomous vehicle that utilizes an NN to identify pedestrians and obstacles [8]. If the network is confident about obstructions, the vehicle should depend more on additional sensor data to decide whether to brake. Similarly, in the context of SCA, other factors should be considered to evaluate the performance of an attack, indicating that the *confidence* of the model is high enough. Specifically, a network must give us a confidence estimate demonstrating how close its prediction distribution is to the *true* distribution of (sub-)keys.

Definition 1 *(cf. [18]) Let $X \in \mathcal{X}$ and $S \in \mathcal{S} = \{1, \ldots, 256\}$ be random variables following a ground truth joint distribution $\pi(X, S) = \pi(S|X)\pi(X)$. Consider a NN represented by $h(\cdot)$, where $h(X) = (\hat{S}, \hat{C})$. Here, $\hat{S}$ is the class prediction, and $\hat{C}$ is its associated* ***confidence****, which is the probability of correctness.*

Comparing this definition with the description of profiled SCA in Section 2.2, $\hat{\varphi}_{X,P}(\cdot, \cdot)$ is equivalent to $h(\cdot)$. Based on the Definition 1, methods have been devised to calibrate the predicted probability estimates $\hat{C}$ to better approximate the true correctness likelihood. For this, calibration techniques require a hold-out validation set. Using that, such methods take post-processing steps to produce calibrated probabilities. Guo et al. analyzed different calibration methods for binary and multi-class setups [18], and they concluded that temperature scaling has the best performance for deep learning applications. Their method, so-called temperature scaling, is an extension of Platt scaling [41].

Temperature scaling. For a multi-class problem, similar to SCA with 256 classes, the NN outputs a class prediction $\hat{s}_i$ and confidence score $\hat{c}_i$ for a given input $\mathbf{x}^i$. In this respect, for each class m $(1 \leq m \leq 256)$ the network logits $\mathbf{z}_i$ are typically given to a softmax function:

$$\sigma_{SM}(\mathbf{z}_i)^{(m)} = \frac{\exp\left(z_i^{(m)}\right)}{\sum_{j=1}^{256} \exp\left(z_i^{(j)}\right)}. \tag{2}$$

The confidence corresponding to the input $\mathbf{x}^i$ is calculated based on Equation (2) as $c_i = \max_m \sigma_{SM}(\mathbf{z}_i)^{(m)}$. Temperature calibration aims to output a calibrated probability $\hat{q}_i$. In that sense, Platt scaling [41] is performed by training a logistic regression model on the validation set to learn a scalar $T > 0$, using the logits as features:

$$\hat{q}_i = \max_m \sigma_{SM}(\mathbf{z}_i / T)^{(m)}. \tag{3}$$

By doing so, T is optimized concerning NLL on the validation set. Note that in this process, model parameters and hyperparameters are **not** changed.

Integration into SCA. After introducing the temperature calibration concept, we describe how it can be integrated into the profiling-attack pipeline. The only requirement for such an integration is a hold-out validation set, usually considered for other purposes (e.g., hyperparameter tuning) in any ML task. As explained above, the validation set is taken randomly from the profiling set to learn the parameter T. After obtaining T, during the attack phase, the logits are calibrated using the temperature learned in the evaluation step, i.e., T. Afterward, the probabilities generated by softmax are fed to the rank function as usual. During this process, no parameter or hyperparameter of the NN is changed.

Softmax vs. stable softmax. It might be thought that to generate output probabilities from the calibrated logits, it is also possible to use stable softmax as suggested in [1]. It is possible to feed the temperature-calibrated logits into stable softmax. We argue that this could not be beneficial as the logits are calibrated using T, optimized by applying logistic regression on a softmax function.

This suggests that the probability distribution of logits would better follow a softmax distribution than the stable softmax with an altered softmax distribution (see Section 2.3). In fact, applying stable softmax itself can be seen as a non-optimized vector scaling; therefore, it may or may not calibrate the logits effectively. Consequently, it is not surprising that in some experiments, stable softmax outperforms temperature-calibrated softmax or temperature-calibrated stable softmax. Generally speaking, Guo et al. have reported that (even) optimized vector scaling cannot surpass temperature calibration [18]. We empirically examine if using stable softmax or coupling temperature calibration with non-optimized vector scaling would be useful in Section 4.

Benefits and limitations of calibration methods in comparison to stable softmax. Stable softmax is primarily a numerically stable normalization that can be interpreted as a fixed, non-optimized transformation of logits; as such, it may reduce pathological saturation and occasionally improves attack performance, but it provides no guarantee of calibration because it is not fitted to a validation set. In contrast, temperature scaling and related calibration methods optimize a parameter, e.g., temperature T, to match predicted confidences to empirical correctness, which directly targets overconfidence and yields a controllable notion of calibration quality via the learned T. The limitation is that calibration requires a hold-out validation set. Under distribution shift, temperature scaling might not be effective (see Section 5). Our experiments therefore treat stable softmax as a low-cost baseline, and we compare it to temperature scaling and to their combination to quantify when a fitted calibration step yields gains beyond non-optimized logit transformations (Section 4).

Interpretation of temperature scaling. By calibrating the $\hat{q}_i$, $\sigma_{SM}(\mathbf{z}_i/T)^{(m)}$ is essentially calibrated. As a result, the output entropy is increased if $T > 1$, which is referred to as "softening" the softmax [18]. When $T = 1$, $\hat{q}_i = \hat{c}_i$, no scaling is applied. This implies that as $T \to 1$, the output probabilities are fairly well approximated. The probability $\hat{q}_i \to 1/256$ indicates that the model is "confused," corresponding to GE close to random. It is a known fact that since the scalar parameter T does not change the maximum of the softmax function, the class prediction $\hat{s}_i$ remains unchanged. Nevertheless, as $\sigma_{SM}(\mathbf{z}_i/T)^{(m)}$ is calibrated, $\mathbf{y}^i$, and consequently, the ranks are scaled (see Equation (1)).

4 Results

This section presents the experimental setup and results of temperature calibration in Sections 4.2-4.3. We also analyze the effects of hyperparameter selection and the tuning objective in Section 4.4.

4.1 Experimental Setup

All the experiments presented in this section are run on a high-computing cluster with a total of 10 CPUs allocated per task and a total memory of 50 GB with

Table 1. MLP models. FC(#neurons) denotes a fully connected layer with #neurons units. SM(#classes) denotes a softmax layer with #classes outputs.

Data set	Leakage model	Architecture
ASCADf	ID [48]	FC(300),FC(300),FC(100), FC(100),FC(100),FC(100), SM(256)
	HW [48]	FC(200),FC(200),FC(100),FC(100),FC(100),FC(100), FC(100),FC(100),FC(100),FC(100),FC(100), SM(9)
ASCADr	ID [48]	FC(400),FC(400),FC(100),FC(100),FC(100),FC(100), Softmax(256)
	HW [48]	FC(200),FC(200),FC(100),FC(100),FC(100),FC(100), FC(100),FC(100), SM(9)
CHES-CTF	HW [48]	FC(400),FC(400),FC(100),FC(100),FC(100), SM(9)

Table 2. CNN models used in our experiments. C(filters, kernel size, strides), P(size, stride), and M(size, stride) denote convolution, average pooling, and max pooling layers. FLAT denotes a flatten layer. FC(#neurons) denotes a fully connected layer with #neurons units. SM(#classes) denotes a softmax layer with #classes outputs.

Data set	Leakage model	Architecture
ASCADf	ID [47]	P(2,2), C(64,50), P(50,50), C(128,3,1), P(2,2), FLAT, FC(20), FC(20), FC(20), SM(256)
	HW [38]	C(16,18,1), FLATT, FC(600), FC(600), SM(9)
ASCADr	ID [48]	C(120,3,1), P(32,2), C(8,1,1), P(32,2), FLAT, FC(30), FC(5), FC(5), SM(256)
	HW [48]	C(4,3,1), P(30,2), FLAT, FC(30), FC(20), FC(20), SM(9)
CHES-CTF	HW [48]	C(216,10,1), P(2,2), C(200,12,1), M(2,2), C(8,2,1), P(2,2), FLAT, FC(300), FC(100), SM(9)

Scalable Gold 6248 and AMD Epyc 7543 processors. We focused on three primary datasets for our analysis as described in Section 2.4. These datasets were used with both ID and HW leakage models, except for the CHES-CTF dataset, whose leakage model is identified as HW [16]. For each dataset, we employed two NN models: a multi-layer perceptron (MLP) and a convolutional neural network (CNN), each adapted to the specifics of the leakage model in terms of the number of output nodes. We stress that we do **not** propose any new MLP or CNN architecture, but study the existing models to answer this question: how confident are those models when extracting the secret key? For this purpose, we have considered the models used in [48,38,47,50]. Among the models used in our paper, some trained models have been available [13], whereas in other cases, we trained the model by using the available codes in [30,37]. In those cases, we carefully compared our results with what has been reported in their respective papers to match them as closely as possible.

Architecture of the models. The architectures of the MLP models that we used are presented in Table 1, and the CNN models are presented in Table 2. We stress that the hyperparameter tuning procedure is out of scope of this work and the Table 1-2 are made following the methodologies in [48,47,38]. These models are selected as typical NN architectures from the SCA literature to ensure reproducibility. In the models considered in our study, NLL has been used as the loss function.

Training and testing sub-dataset preparation. In this study, the datasets employed, as detailed in Section 2.4, consist of two parts: profiling and attack

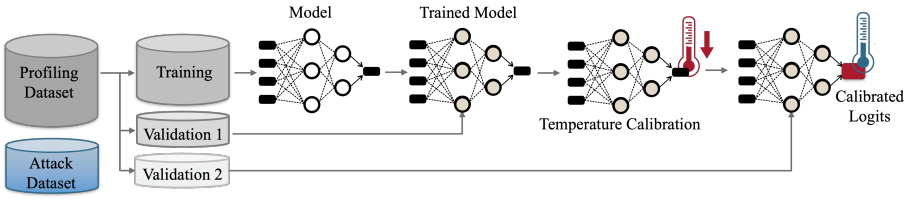


Fig. 1. Experimental setup. Validation1 is randomly sampled from the profiling set and used for temperature calibration (Section3). During the attack phase, logits are calibrated using the learned temperature, and softmax or stable-softmax probabilities are then passed to the rank function as usual. No network parameters or hyperparameters are changed. Validation 2 is used only to measure the post-calibration temperature and assess calibration quality.

traces. Validation sets are randomly taken from profiling traces. The NLL and accuracy were monitored during training not to have an overfitting issue (the role of validation and training trace counts is discussed in Appendix A.) The Validation 1 set is then used for temperature calibration, i.e., learning the scalar parameter T, as detailed in Section 3. Subsequently, in the attack phase, we apply the calibrated model to the attack traces. During this phase, we use the temperature learned in the evaluation step to calibrate the logits. The "Validation2" set is used to calculate the calibrated model's temperature to assess the calibration's performance. This process is illustrated in Figure 1. Next, softmax function generates probabilities from logits. These probabilities are then fed to the rank functions. As can be understood from this workflow, neither the parameters nor the hyperparameters of NNs were changed throughout the calibration process.

Furthermore, to examine whether stable softmax (non-optimized vector scaling, see Section 3) could outperform temperature calibration, we also implement the stable softmax function as the activation function for the models. The results for this setting are marked as "stable softmax" throughout this section. We also consider the combination of temperature scaling and stable softmax, whose related results are marked as "calibrated stable softmax." The GE curves of our implementations will be presented in the subsequent subsections.

4.2 Temperature Calibration: Impact on Confidence

First, we aim to discuss the impact of the temperature calibration on the confidence of the SCA models. The model outputs a probability distribution at its softmax layer. The subkey with the highest probability is the results of the classification task. The probability assigned to this subkey is called confidence. For a calibrated model, the confidence should be close to accuracy. To assess the calibration of our models, we pick two models from table 3, one MLP model trained on the ASCAD-f fixed dataset with the Id leakage model, and one CNN model with ID leakage trained on ASCAD-r dataset. By looking at the first column of the confidence histograms presented in the Fig. 2, we observe that

Table 3. T_{GE0} and the model temperature T before and after calibration, where $T \approx 1$ indicates good calibration. Results use a validation subset of the profiling set of approximately one-fourth of the profiling traces (see Appendix A for results with 2000 validation traces consistent with the respective studies). Since our focus is the impact of calibration, the key quantity is the relative reduction in T_{GE0} rather than reproducing or improving prior results. The last column reports the standard deviation (Std.) of the calibrated T, computed over 30 repetitions using 1000 randomly sampled validation traces.

Dataset	Model	Leakage model	Uncalibrated		Calibrated		Std. of T
			T	T_{GE0}	T	T_{GE0}	
ASCADf	MLP	ID [48]	4.110	733	1.042	662	0.043
		HW [48]	6.470	3368	0.950	2736	0.09
	CNN	ID [47]	3.677	508	0.910	442	0.24
		HW [38]	1.601	1922	1.015	1895	0.04
ASCADr	MLP	ID [48]	3.05	1575	1.026	1025	0.18
		HW [48]	7.976	3443	0.977	2686	0.73
	CNN	ID [48]	1.165	324	1.014	281	0.075
		HW [48]	1.670	1851	0.983	1615	0.097
CHES-CTF	MLP	HW [48]	15.194	4403	1.028	3381	1.70
	CNN	HW [48]	24.707	4706	0.956	4550	1.04

there is a gap between the accuracy of the model and the average value of the model's confidence. In both cases the confidence of the model is more than its accuracy, showing the over-confidence of the model. After calibration, we see that the confidence of the model gets closer to the accuracy in both cases.

4.3 Temperature Calibration: Impact on GE

This section covers the results obtained by applying temperature calibration of the models introduced in Section 4.1. For the results presented in this subsection, the number of traces in the Validation 1 and 2 datasets is half that of the training traces. This is in accordance with the recommendation in the relevant literature [35].

ASCAD-f under ID leakage model. We begin with ASCAD with the fixed key (ASCAD-f) dataset that is relatively easier to break [48]. In Fig. 3, we show the results for ASCAD-f dataset with ID leakage model. The rank curves drawn for both MLP and CNN models indicate that the models are well-trained. When comparing the results for uncalibrated models with what has been presented in [48,47], the trends of the curves are similar in terms of achieving a lower rank and the number of traces to break the target. Note that as we are interested in investigating the impact of temperature calibration, the relative reduction in T_{GE0} is important rather than reproducing/improving the results in the respective papers. The attack results for the calibrated model have improved for both MLP and CNN models, reaching $GE = 0$ with 662 for the MLP model and 442 for the CNN model. The MLP model [48] had a temperature of 4.11. For the CNN models that we considered in this subsection, the temperature of the model [47] was 3.677.

ASCAD-f under ID leakage model with stable softmax vs. calibrated stable softmax. In the MLP model, compared to the results for temperature

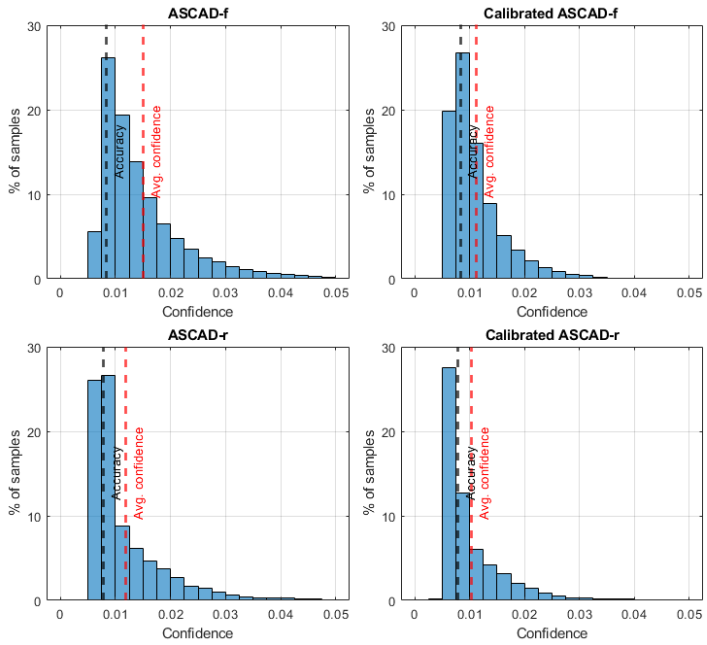
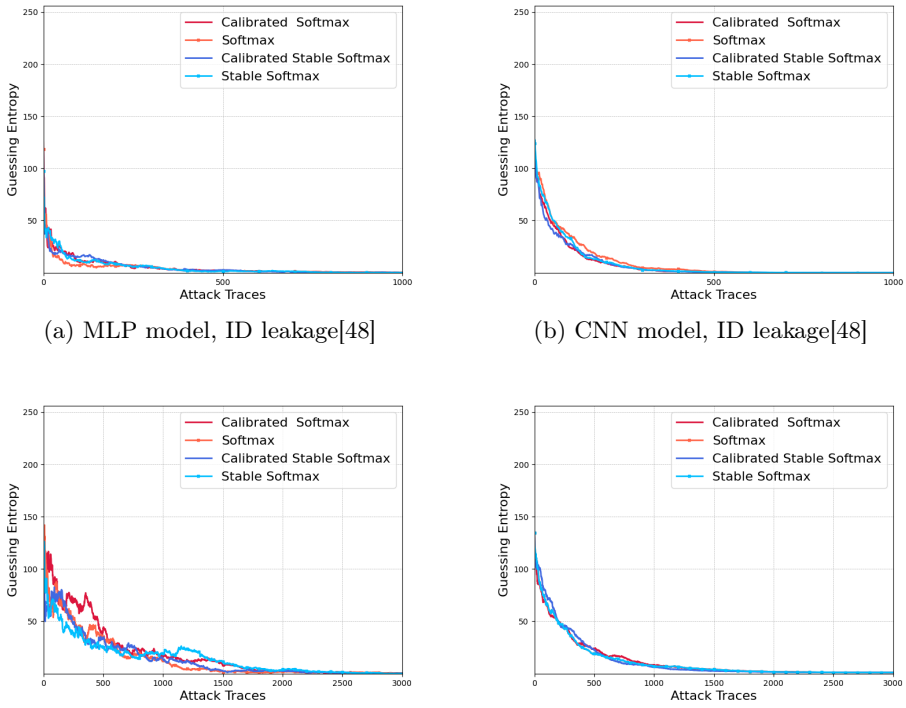


Fig. 2. The impact of the calibration on the confidence histograms of the two sample models trained on ASCAD-f and ASCAD-r datasets.

scaling presented in Table 4.3, the stable softmax demonstrated a good performance in reaching $GE = 0$, both before calibration with 458 traces and after calibration with 595 traces. In the case of the CNN model [47], the stable softmax achieved $GE = 0$ by utilizing 462 traces before calibration, and this number decreased to 409 traces after calibration. Therefore, we cannot conclude that coupling temperature scaling and stable softmax (non-optimized vector scaling) is useful for this dataset as the calibrated MLP model does not show a significant improvement over the uncalibrated one. However, for CNNs, a promising trend is observed.

ASCAD-f under HW leakage model. Fig. 3 depicts GE for both MLP model [48], and the CNN model [38], where the HW leakage model is considered. For the MLP model, the uncalibrated temperature was 6.470, which notably decreased to 0.950 after calibration, see Table 3. The CNN model started with a lower initial temperature of 1.601, which calibration further reduced to 1.015. Notably, calibration enhanced the GE of the CNN model, with the calibrated model achieving better T_{GE0}, see Table 3.

ASCAD-f under HW leakage model with stable softmax vs. calibrated stable softmax. In this context, the stable softmax showed superior performance compared to the softmax in the CNN model, with a T_{GE0} of 1845 before calibration and 1757 after calibration. Similarly, for the MLP model, calibration improved the T_{GE0} of the stable softmax from 2539 to 2199, and the T_{GE0} of soft-



(a) MLP model, ID leakage[48]

(b) CNN model, ID leakage[48]

(c) MLP model, HW leakage[48]

(d) CNN model, HW leakage[48]

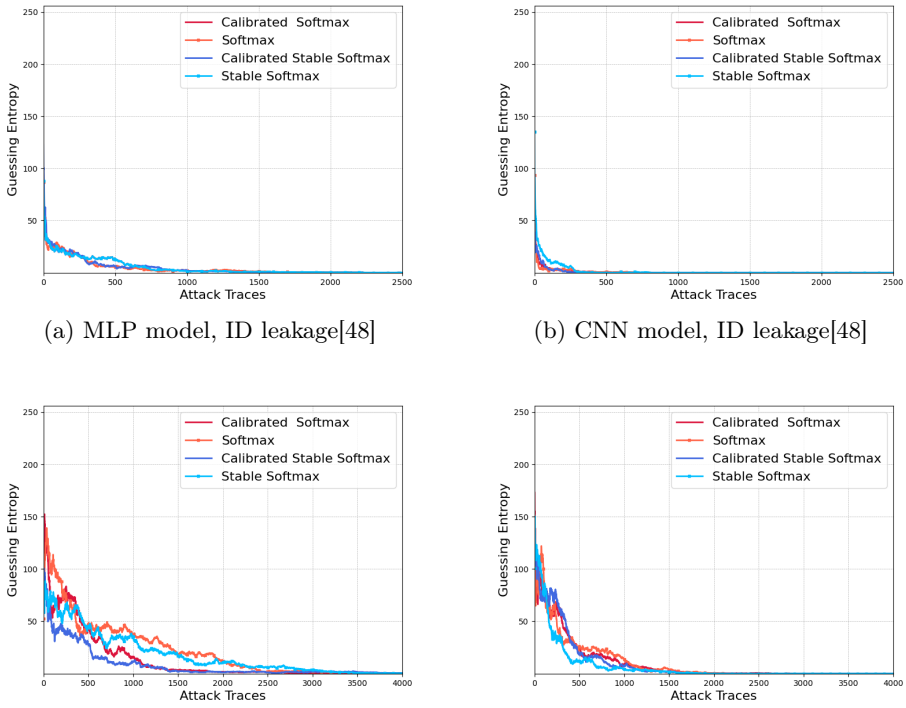
Fig. 3. GE for ASCAD-f dataset [2].

max from 3368 to 2736. Hence, for this setting, the combination of temperature scaling and stable softmax was useful.

ASCAD-r under ID leakage model. Next, we move to ASCAD-r dataset that random keys are used for the profiling traces. GE results of the models for the ASCAD dataset with the random key (ASCAD-r) and ID leakage model are presented in Fig. 4. Here, the CNN model [48] was well trained, and the temperature was 1.165. The calibrated model performed better in terms of GE, for the softmax T_{GE0} reduced to 281 from 324, See Table 3. In MLP model [48], we saw a temperature of 3.05. For this model, the number of the traces needed for $GE = 0$ was 1575 that reduced to 1025 after calibration3.

ASCAD-r under ID leakage model with stable softmax vs. calibrated stable softmax. For CNNs, when applying the stable softmax, $T_{GE0} = 296$ that reduces to $T_{GE0} = 223$ after calibration. For MLPs, The $T_{GE0} = 875$ for the stable softmax, and unlike the trend that we saw for CNNs after calibration, this number increased to 1290. Hence, again, no conclusion can be made about the usefulness of the combination of temperature calibration and stable softmax.

ASCAD-r under HW leakage model. Fig. 4 shows the rank curves of the HW leakage model for the ASCAD-r dataset. Like other models in this section, here again, the results of the CNN [48] model were better, resulting in



(a) MLP model, ID leakage[48]

(b) CNN model, ID leakage[48]

(c) MLP model, HW leakage[48]

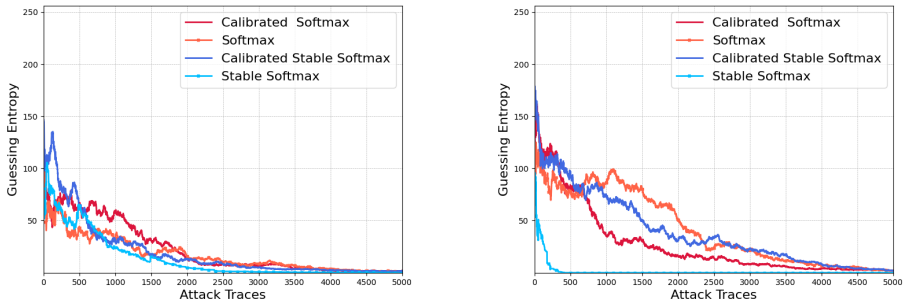
(d) CNN model, HW leakage[48]

Fig. 4. GE for ASCAD-r dataset [2].

a lower temperature and better calibration. The temperature before calibration was 1.670 for the CNN model, whereas for the MLP [48], this number was 7.976, according to the results presented in table 3. For the MLP, After calibration the rank curves improved so that T_{GE0} was 2686 for the calibrated model, while before calibration, the model needed 3443 traces to reach $GE = 0$, see Table 3.

ASCAD-r under HW leakage model with stable softmax vs. calibrated stable softmax. For the MLP, the results obtained for stable softmax also improved by calibration, reducing the T_{GE0} from 3447 to 2510. The same holds for the CNN models: the calibrated model required 1615 traces to reach the $GE = 0$ compared to 1851 traces for the uncalibrated model. For the CNN, T_{GE0} was 1597 before calibration and 1575 after calibration for the stable softmax, which were lower than the softmax in both cases. Overall, we could observe that temperature calibration combined with the stable softmax could enhance the attack performance.

CHES-CTF under HW leakage. As this dataset is presented only for HW leakage, next, we discuss the results of the CHES-CTF dataset under that scenario. For this dataset, the temperature of both the MLP and CNN models [48] were high. MLP model's temperature was 15.194; following calibration, the model used fewer traces for $GE = 0$. The T_{GE0} was 4403 for the uncalibrated



(a) MLP model, ID leakage[48] (b) CNN model, ID leakage[48]

Fig. 5. GE for CHES-CTF dataset.

model and 3381 after calibration. The same results were observed for the CNN model with a temperature of 24.707, and the model used 150 fewer traces to reach $GE = 0$ after calibration, see Fig. 5.

CHES-CTF under HW leakage model with stable softmax vs. calibrated stable softmax. The models incorporating the stable softmax performed significantly better than softmax, as shown in Fig. 5. T_{GE0} for the MLP model was 2374 and 328 for the CNN. As shown in Fig. 5, calibration of models with stable softmax output layer was not beneficial.

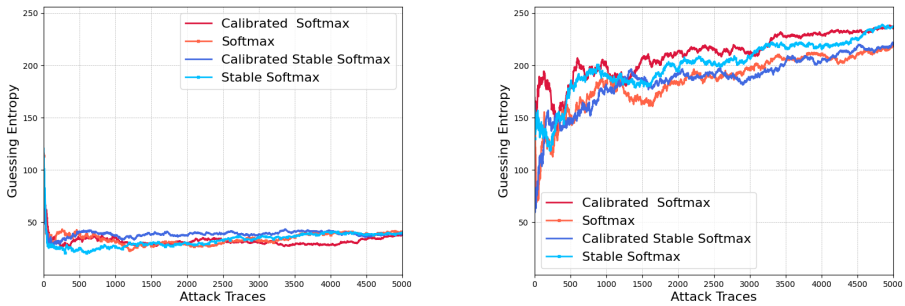
Summary. Based on the results shown in Table 3 for all the trained models, after calibration, the temperature decreases to a number close to 1, which indicates that the output probabilities are fairly well approximated. Our results shown in Figures 3-5 demonstrated that temperature scaling and non-optimized vector scaling (stable softmax) are both effective when launching an attack under various scenarios (different datasets, leakage models, NN models). Nevertheless, we did not conclude that *combining* temperature scaling and stable softmax (marked as calibrated stable softmax) would always be helpful.

4.4 Temperature Calibration: Impact of Hyperparameters

In this section, we aim to present the results for the models that have undergone hyperparameter tuning, although it has performed improperly. When the search space is large, too few hyperparameters are chosen to be tuned simultaneously, an unfit range of values is chosen, or wrong objective function (tuning metrics) are selected, hyperparameter tuning may fail. If the hyperparameters are not optimized, temperature calibration does not help much. As an example, Fig. 6 shows the results for the models whose hyperparameters were attempted to tune. In these cases, the range of hyperparameters was unfit *intentionally*, leading to a failure in tuning. Table 4 lists the hyperparameters of these models (see Tables 1-2 for the differences between proper and improper hyperparameters). As we expected, using improper hyperparameters increases the temperature of the models. For the CNN and the MLP models [48], the temperature was 14.84

Table 4. Models with improper hyperparameters (notations are similar to Tables 1-2).

Model	Architecture
MLP [48]	FC(200), FC(200), FC(100), FC(100), FC(100), FC(100), FC(100), SM(256)
CNN [48]	C(152,7,1), M(2,2), C(24,8,1), M(2,2), C(8,2,1), P(2,2), FLAT, FC(500), FC(100), FC(100), SM(256)



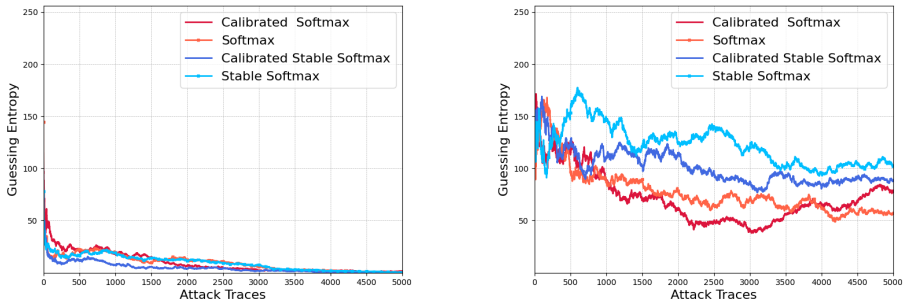
(a) MLP with key rank [48] (b) CNN with accuracy [48]

Fig. 6. GE of the models with improper hyperparameters for ASCAD-r and ID leakage model.

and 13.58, respectively. The GE curves either converge to a high value or exhibit random behavior. What can be concluded is that the temperature can serve as a metric to verify if the hyperparameters are tuned well before launching the attack.

Impact of improper objective function. In order to tune the hyperparameters, it is necessary to define an objective function. There are three common objective functions that Wu et al. [48] utilized in their research: accuracy, key rank, and L_m [49]. The first two objective functions are commonly used in ML tasks, whereas L_m represents the correlation between the leakage distribution variation observed for different key candidates and the key guessing vector. Hence, L_m gives insight into the profiling model's generalizability [49]. Wu et al. have demonstrated that choosing these objective functions has impacted the attack performances for different models and datasets [48]. In our experiments, we aimed to understand the effect of these objective functions on the temperature of the models. In doing so, we focused on two experiments performed on the ASCAD-r to train MLP and CNN models with ID leakage models. Fig. 7 illustrates the results for a model trained with L_m objective, while the best results for the MLP model were obtained by incorporating the key rank objective function. Comparing this with the results in Fig. 4, it is evident that the attack performance is degraded. The temperature of this model was 6.981, twice as large as the temperature of the model with the key rank objective.

In the case of CNNs, the best results were achieved by considering the accuracy as the objective function. We trained another model using the L_m objective function to see the differences in the temperatures and GE. Fig. 7 shows the GE



(a) MLP with key rank [48] (b) CNN with accuracy [48]

Fig. 7. GE of the models with wrong objective functions for ASCAD-r and ID leakage model.

for this model, which is not as good as the results in Fig. 4. The temperature of this model was 20.41 which is much higher than the model trained with the proper objective.

Summary. Factors that affect the quality of hyperparameter tuning directly influence the temperature of the model. As an example, an improper objective function leads to a high model temperature. Other factors, like the range of values, can similarly impact the tuned model and, consequently, the model's temperature. In such cases, the temperature of the model is high. This could be beneficial in assessing the performance of the hyperparameter tuning as well as attack performance, even before mounting the attack.

5 Discussion

Why SCA should not push for higher accuracy levels. Besides improving the performance of the attack in terms of GE, temperature scaling can give insight into issues concerning the evaluation metrics of SCA. As mentioned before in Section 2.3, multi-class ML tasks are often tackled by incorporating the softmax output layer along with the NLL loss function. NLL has also found application in NN-assisted SCA since it has been proven that NLL is inversely related to "perceived information" (PI) [43,34,9]. The PI refers to generalizing the mutual information between the side-channel traces and the leakage profiling model (i.e., the ML trained on the traces). This aligns with ML's view: NLL is minimized if and only if the NN recovers the ground truth conditional distribution $\pi(S|X)$ [18]. Therefore, the PI can quantify how well the ML model is trained. Specifically, minimizing the NLL loss function (similarly, cross-entropy) during NN training is asymptotically equivalent to maximizing the perceived information and improving the trained NN performance cf. [34]. Consider a scenario where NLL reaches a minimum value. If the training is not stopped, the model still can improve its accuracy (see Fig. 8), although the model is overfitting to NLL, i.e., NLL starts increasing. This effect

can also be formulated as overfitting to NLL without overfitting the classification accuracy. In fact, overfitting to NLL helps improve classification accuracy, where the network reaches better classification accuracy *"at the expense of well-modeled probabilities,"* cf. [18]. In the context of SCA, if probabilities are not well modeled, the attack performance in terms of GE degrades [38]. This clarifies what has been empirically verified in SCA-related literature [38], i.e., accuracy might not reflect the performance of the attacks as GE does. To mitigate overfitting to NLL, training can be stopped before entering the NLL-overfitting regime by monitoring NLL or using information-based stopping criteria that indirectly minimize PI [1]. Alternatively, temperature scaling can be applied to calibrate the network's output probabilities and confidence in a simple and effective manner.

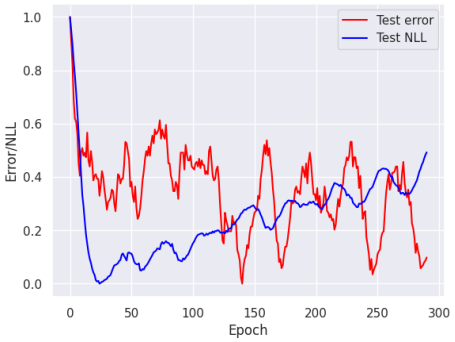


Fig. 8. Test error and NLL of the MLP model trained on ASCAD-f datasets for 300 epochs, values are scaled to fit in the same figure.

Why compact NNs for SCA? Another aspect of NN-assisted SCA that can benefit from this study is the configuration of NNs, particularly their size. Several works have highlighted the importance of reducing the NNs' trainable parameters [50,47,1], mainly to reduce computational complexity and memory footprint. On the other hand, studies, e.g., [48], have argued that there might be no reason why the need for smaller NN should be emphasized, as the size of NNs used for SCA is small compared to other ML tasks.

According to learning theory, large models with little or no regularization will not generalize well [51,52]. In this regard, although increasing the NN's capacity (depth and width) may reduce classification error, such increases negatively affect model confidence. The discussion on the disconnect between overfitting to NLL and accuracy, provided in Section 5, is also relevant to this aspect. When the model capacity is high, additional training epochs can be needed to converge and reach the desired performance. During those additional epochs, the model's classification error may be reduced; however, it is possible that NLL will not be further minimized, and the model will start overfitting to NLL. While this benefits classification accuracy, it is not helpful for SCA.

Impact of class imbalance and desynchronization on temperature scaling. Class imbalance and desynchronization can affect both the interpretation and the applicability of temperature scaling in NN-assisted SCA. With class imbalance, the network may be over-confident on majority classes and miscalibrated on minority classes, therefore, a single global temperature T can improve average calibration. However, it leaves class-conditional errors that still impact key ranking and guessing entropy. Desynchronization, here meaning shifts in feature positions within a trace, induces a distribution shift. If both training and

validation come from the same desynchronized distribution, temperature scaling remains valid and typically transfers to attack traces drawn from that same distribution. It can reduce overconfidence under the shift and may improve GE. On the other hand, a temperature learned on synchronized validation traces may not transfer to desynchronized attack traces. Therefore, T should be calibrated on validation data that matches the expected imbalance and desynchronization level in attack traces; the future work can extend this study by considering the transferability of temperature scaling in this case.

Link between temperature scaling and effective perceived information (PI)/cross-entropy (CE). [21,22] have applied "inverse temperature" β, where temperature $T = 1/\beta$. [21] shows that applying an inverse-temperature transform to the modeled posterior changes CE and PI without changing SR, which is exactly why "raw" CE or PI can be misleading as attack-performance predictors. To remove that ambiguity, they define effective CE/PI by optimizing CE (or PI) over β. [22] reiterates the same point and explicitly motivates ECE/EPI as "calibrating" CE/PI with respect to inverse temperature β. In our setting, temperature scaling learns a single scalar T on a hold-out validation set by minimizing NLL after scaling logits by $1/T$ (i.e., softmax on z/T). Defining $\beta = 1/T$, our procedure is a practical way to pick an inverse temperature using the validation dataset, rather than sweeping β abstractly as in [21,22]. That is, our learned T is a concrete instantiation of the temperature degree of freedom that the PI literature flags as the source of CE/PI ambiguity. Our results demonstrate that applying the learned T can reduce GE. While [21,22] describe a transformation family under which SR (and GE) remain unchanged, our calibration uses labeled validation data to select a specific scaling in a practical NN-assisted SCA pipeline. This optimization improves the ranking behavior (GE) of a miscalibrated model by refining the probability mass assigned to the correct key across multiple traces.

6 Conclusion

In the context of SCA, this paper addresses neural-network overconfidence and shows that post hoc calibration can improve guessing entropy (GE). We focus on temperature scaling, which integrates into NN-assisted SCA without retraining, and we show that the learned temperature provides a useful indicator of hyperparameter tuning quality. Using publicly available models across multiple benchmark datasets, we find that temperature scaling reduces the traces needed to reach $GE = 0$ and reflects the effects of training-set size and tuning choices. The main cost is a hold-out validation set for calibration, which can be limiting when profiling traces are scarce, although profiling on an open device copy often mitigates this constraint.

Acknowledgments

This work has been partially supported by NSF under award number 2138420.

Appendix A: Impact of the Number of Validation Traces

The main objective of this appendix is to observe the effect of the number of training/validation traces on the model's temperature. In this subsection, we used 2000 traces as the validation set and trained the models with the rest of the traces in the profiling traces. This means that the models had the luxury of using a significantly higher number of training traces in comparison to the cases studied in Section 4.3. This reduces the model's variance, and therefore, it is expected that the temperature of uncalibrated models will be reduced.

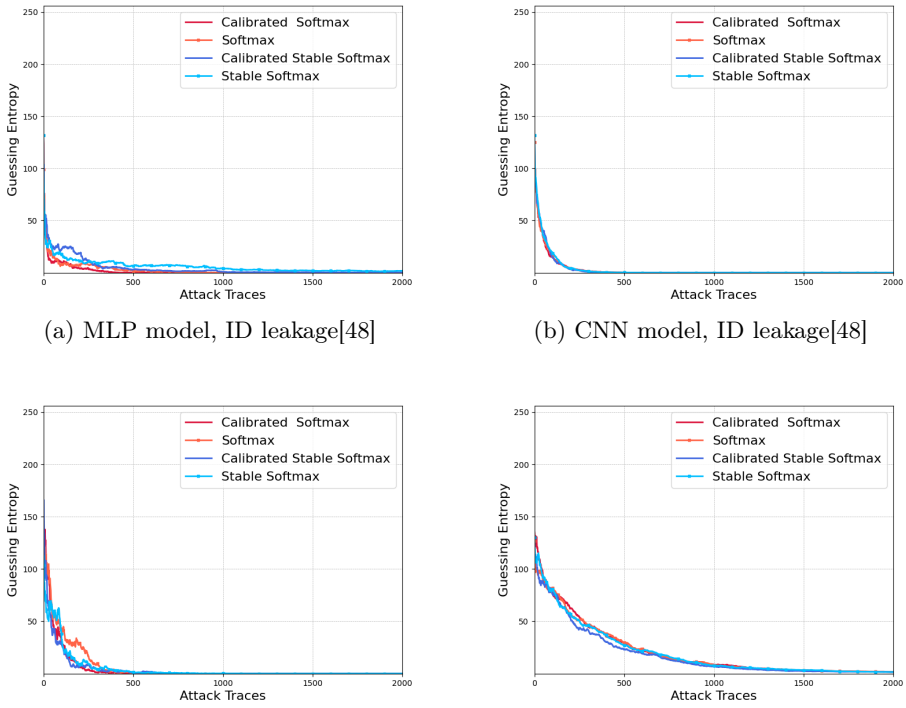
Table 5. T_{GE0} and temperature of the trained models before temperature calibration as well as T_{GE0} after calibration. T indicates the temperature with $T \approx 1$ showing that the model is well calibrated. 2000 traces taken from the profiling dataset are used for calibrating the trained model.

Dataset	Model	Leakage model	Uncalibrated		Calibrated
			T	T_{GE0}	T_{GE0}
ASCADf	MLP	ID [48]	2.953	449	295
		HW [48]	1.604	432	246
	CNN	ID [47]	1.513	226	190
		HW [38]	1.605	1922	1895
ASCADr	MLP	ID [48]	3.08	1492	1137
		HW [48]	7.216	1152	1164
	CNN	ID [48]	1.120	347	269
		HW [48]	1.794	1376	1038
CHES-CTF	MLP	HW [48]	15.263	2198	768
	CNN	HW [48]	25.17	5000	4913

ASCAD with the fixed key This dataset contains 50,000 profiling traces. To achieve the results presented in Section 4.3, a fourth of these traces were used for validation. Here, only 2,000 traces are taken from the profiling dataset for this purpose, allowing more traces for training. Consequently, the models exhibit improved performance.

ID leakage model. Fig. 9 shows the results for the ID leakage model. Using a larger number of traces for training CNNs resulted in obtaining models with a good performance. The temperature was reduced to 1.513 for the CNN model[47] (see Table 3 and Table 5). Compared to the results in Table 3, CNN model [47] reached lower ranks with fewer traces: $T_{GE0} = 190$ for the calibrated softmax, while it was 226 before calibration. Similarly, T_{GE0} for the model with stable softmax was also improved after calibration from 280 to 226. The MLP model [48] had an initial temperature of 2.953, and T_{GE0} reduced from 449 to 295 for softmax and from 3360 to 989 for stable softmax.

HW leakage model. The GE curves depicted in Fig. 9 present the results obtained for models with the HW leakage model. For the CNN model [38], an increase in the number of validation traces slightly changed the temperature. The temperature was 1.605 for the CNN model, whereas it was 1.604 for the MLP model [48](see table 5). Calibration improved the results in both models, reducing T_{GE0} from 432 to 346 in the MLP model and from 1922 to 1895 in the CNN model. We can also see improvements in the results when taking stable



(a) MLP model, ID leakage[48]

(b) CNN model, ID leakage[48]

(c) MLP model, HW leakage[48]

(d) CNN model, HW leakage[48]

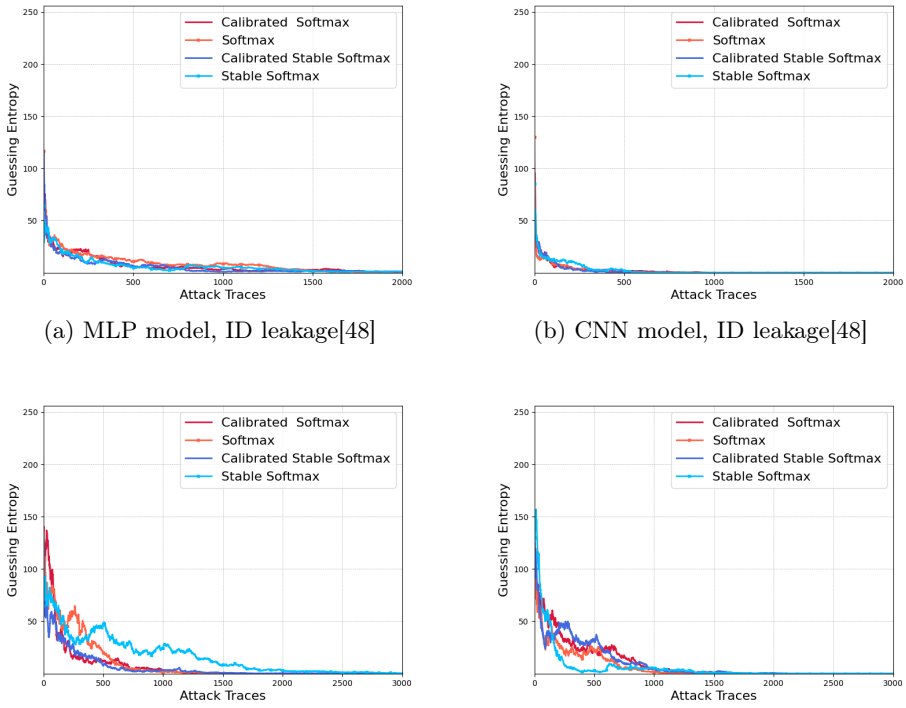
Fig. 9. GE for ASCAD-f dataset [2] with 2000 traces in the validation set.

softmax into account. Before calibration, T_{GE0} was 540 and 1845, which were reduced to 501 and 1757 for the MLP model and CNN model with stable softmax, respectively.

ASCAD with the random key This dataset is larger than the ASCAD-f dataset, therefore, it is expected that the change in the training/validation split does not influence the model variance, and consequently, temperature.

ID leakage model. For the ASCAD-r dataset, using the ID leakage model, the MLP model [48] temperature was 3.08, slightly higher than the previous 3.05, see Table 3, which is not statistically significant. Similarly, the CNN model [48] temperature did not change significantly, and it was 1.120. The MLP model with the calibrated stable softmax achieved $T_{GE0} = 815$, while the result for the CNN model with calibrated softmax was $T_{GE0} = 269$ (see Fig. 10).

HW leakage model. Next, as shown in Fig. 10, the calibrated model performed well in terms of reaching $GE = 0$ with fewer attack traces. The CNN model had a low temperature, but compared to the results in Section 4.3, we observed that the temperature was slightly increased to 1.794. The T_{GE0} was obtained for 1038 and 1376 traces before and after calibration. The temperature of the MLP model was 7.216. It took 1164 traces to achieve $GE = 0$ for this model after calibration and 1152 before it. CNNs with stable softmax and calibrated



(a) MLP model, ID leakage[48]

(b) CNN model, ID leakage[48]

(c) MLP model, HW leakage[48]

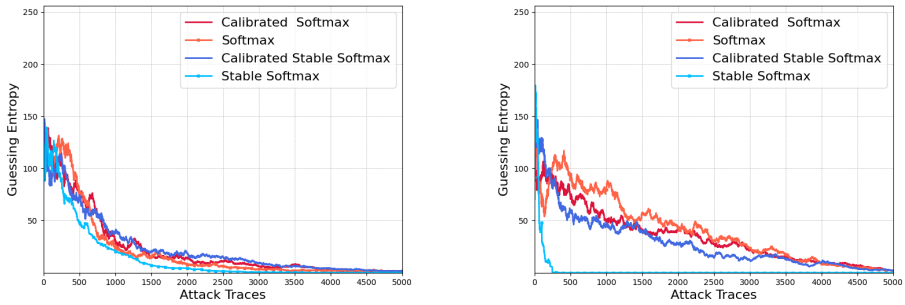
(d) CNN model, HW leakage[48]

Fig. 10. GE for ASCAD-r dataset [2] with 2000 traces in the validation set.

softmax could also achieve competitive results, while for MLP ones, calibrated softmax was advantageous.

CHES-CTF dataset Finally, we can see the GE curves of the models trained on the CHES-CTF dataset in Fig. 11. Here, the temperature of the CNN and MLP was 25.17 and 15.263, respectively. High temperature and poor performance indicate that the models suffer from overfitting. Interestingly enough, stable softmax outperformed other calibration methods and the softmax itself.

Summary. In NN-assisted SCA, where profiling datasets are relatively limited, it is not recommended to reduce training traces merely to allocate more validation traces. Both stable softmax (non-optimized vector scaling) and temperature scaling can improve attack performance. Moreover, increasing profiling traces can induce overfitting, as observed on CHES-CTF; therefore, monitoring NLL and accuracy during training is recommended. This highlights that simply increasing the number of profiling traces does not automatically improve attack effectiveness, as excessive training can deteriorate probability calibration even when classification accuracy appears stable.



(a) MLP model, ID leakage[48] (b) CNN model, ID leakage[48]

Fig. 11. GE of CHES-CTF with the HW leakage model and 2000 traces in the validation set.

References

1. Acharya, R.Y., Ganji, F., Forte, D.: Information theory-based evolution of neural networks for side-channel analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 401–437 (2023)
2. Benadjila, R., Prouff, E., Strullu, R., Cagli, E., Dumas, C.: ASCAD: the ATMega8515 SCA traces databases (fixed key). `[Online]https://github.com/ANSSI-FR/ASCAD/tree/master/ATMEGA_AES_v1/ATM_AES_v1_fixed_key` [Accessed: Jan.8, 2024] (2017)
3. Benadjila, R., Prouff, E., Strullu, R., Cagli, E., Dumas, C.: ASCAD: the ATMega8515 SCA traces databases (variable key). `[Online]https://github.com/ANSSI-FR/ASCAD/tree/master/ATMEGA_AES_v1/ATM_AES_v1_variable_key` [Accessed: Jan.8, 2024] (2017)
4. Benadjila, R., Prouff, E., Strullu, R., Cagli, E., Dumas, C.: ASCADv1 Dataset: the atmega8515 sca campaigns. `[Online]https://github.com/ANSSI-FR/ASCAD/tree/master/ATMEGA_AES_v1` [Accessed: Jan.8, 2024] (2017)
5. Benadjila, R., Prouff, E., Strullu, R., Cagli, E., Dumas, C.: Study of deep learning techniques for side-channel analysis and introduction to ascad database (2018)
6. Benadjila, R., Prouff, E., Strullu, R., Cagli, E., Dumas, C.: Deep learning for side-channel analysis and introduction to ascad database. Journal of Cryptographic Engineering **10**(2), 163–188 (2020)
7. Bengio, Y.: Deep learning of representations: Looking forward. In: International conference on statistical language and speech processing. pp. 1–37. Springer (2013)
8. Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., et al.: End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316 (2016)
9. Bronchain, O., Hendrickx, J.M., Massart, C., Olshevsky, A., Standaert, F.X.: Leakage certification revisited: Bounding model errors in side-channel security evaluations. In: Annual Intrl. Cryptol. Conf. pp. 713–737. Springer (2019)
10. Cagli, E.: Feature extraction for side-channel attacks. Ph.D. thesis, Sorbonne université (2018)
11. Cagli, E., Dumas, C., Prouff, E.: Convolutional neural networks with data augmentation against jitter-based countermeasures. In: Intrl. Conf. on Cryptographic Hardware and Embedded Systems. pp. 45–68. Springer (2017)

12. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Intrl. Workshop on Cryptographic Hardware and Embedded Systems. pp. 13–28. Springer (2002)
13. COSIC, K.L.: TCHES20V3 CNN SCA Repository: cnn-based side-channel analysis. [Online]https://github.com/KULeuven-COSIC/TCHES20V3_CNN_SCA [Accessed: Jan. 8, 2024] (2020)
14. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: Cryptographic Hardware and Embedded Systems—CHES 2001. pp. 251–261. Springer (2001)
15. Gilmore, R., Hanley, N., O'Neill, M.: Neural network based attack on a masked implementation of aes. In: Intrl. Symposium on Hardware Oriented Security and Trust (HOST). pp. 106–111. IEEE (2015)
16. Gohr, A., Jacob, S., Schindler, W.: Ches 2018 side channel contest ctf-solution of the aes challenges. Cryptology ePrint Archive (2019)
17. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT press (2016)
18. Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. In: International conference on machine learning. pp. 1321–1330. PMLR (2017)
19. Heuser, A., Zohner, M.: Intelligent machine homicide. In: Intrl. Workshop on Constructive Side-Channel Analysis and Secure Design. pp. 249–264. Springer (2012)
20. Hospodar, G., Gierlichs, B., De Mulder, E., Verbauwhede, I., Vandewalle, J.: Machine learning in side-channel analysis: a first study. Journal of Cryptographic Engineering **1**(4), 293 (2011)
21. Ito, A., Ueno, R., Homma, N.: Perceived information revisited: New metrics to evaluate success rate of side-channel attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 228–254 (2022)
22. Ito, A., Ueno, R., Homma, N.: Perceived information revisited ii: Information-theoretical analysis of deep-learning based side-channel attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems **2025**(1), 450–474 (2025)
23. Jin, H., Song, Q., Hu, X.: Auto-keras: An efficient neural architecture search system. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. pp. 1946–1956 (2019)
24. Kerkhof, M., Wu, L., Perin, G., Picek, S.: No (good) loss no gain: Systematic evaluation of loss functions in deep learning-based side-channel analysis. IACR Cryptol. ePrint Arch., 2021/1091 (2021)
25. Kim, J., Picek, S., Heuser, A., Bhasin, S., Hanjalic, A.: Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. IACR Trans. on Cryptographic Hardware and Embedded Systems pp. 148–179 (2019)
26. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings 19. pp. 388–397. Springer (1999)
27. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Annual International Cryptology Conference. pp. 104–113. Springer (1996)
28. Lerman, L., Bontempi, G., Markowitch, O.: A machine learning approach against a masked aes. Journal of Cryptographic Engineering **5**(2), 123–139 (2015)
29. Lerman, L., Poussier, R., Bontempi, G., Markowitch, O., Standaert, F.X.: Template attacks vs. machine learning revisited (and the curse of dimensionality in side-channel analysis). In: Intrl. Workshop on Constructive Side-Channel Analysis and Secure Design. pp. 20–33. Springer (2015)

30. Lichao Wu, Guilherme Perin, S.P.: AutoSCA: automated hyperparameter tuning for deep learning-based side-channel analysis. [Online]https://github.com/AISyLab/AutoSCA [Accessed: Jan. 8, 2024] (2022)
31. Maghrebi, H., Portigliatti, T., Prouff, E.: Breaking cryptographic implementations using deep learning techniques. In: Intrl. Conf. on Security, Privacy, and Applied Cryptography Engineering. pp. 3–26. Springer (2016)
32. Mangard, S., Oswald, E., Popp, T.: Power analysis attacks: Revealing the secrets of smart cards, vol. 31. Springer Science & Business Media (2008)
33. Martinasek, Z., Hajny, J., Malina, L.: Optimization of power analysis using neural network. In: Intrl. Conf. on Smart Card Research and Advanced Applications. pp. 94–107. Springer (2013)
34. Masure, L., Dumas, C., Prouff, E.: A comprehensive study of deep learning for side-channel analysis. IACR Trans. on Cryptographic Hardware and Embedded Systems pp. 348–375 (2020)
35. Minderer, M., Djolonga, J., Romijnders, R., Hubis, F., Zhai, X., Houlsby, N., Tran, D., Lucic, M.: Revisiting the calibration of modern neural networks. Advances in Neural Information Processing Systems **34**, 15682–15694 (2021)
36. Murphy, K.P.: Machine Learning: A Probabilistic Perspective. MIT press (2012)
37. Perin, G., Chmielewski, Ł., Picek, S.: Repository code to support tches2020 paper "strength in numbers: Improving generalization with ensembles in machine learning-based profiled side-channel analysis". [Online]https://github.com/AISyLab/EnsembleSCA [Accessed: Jan. 4, 2024] (2020)
38. Perin, G., Chmielewski, Ł., Picek, S.: Strength in numbers: Improving generalization with ensembles in machine learning-based profiled side-channel analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 337–364 (2020)
39. Picek, S., Heuser, A., Jovic, A., Bhasin, S., Regazzoni, F.: The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. IACR Trans. on Cryptographic Hardware and Embedded Systems **2019**(1), 1–29 (2019)
40. Picek, S., Perin, G., Mariot, L., Wu, L., Batina, L.: SoK: Deep learning-based physical side-channel analysis. IACR Cryptol. ePrint Arch., 2021/1092 (2021)
41. Platt, J., et al.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. Advances in large margin classifiers **10**(3), 61–74 (1999)
42. Quisquater, J.J., Samyde, D.: Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In: Smart Card Programming and Security: International Conference on Research in Smart Cards, E-smart 2001 Cannes, France, September 19–21, 2001 Proceedings. pp. 200–210. Springer (2001)
43. Renauld, M., Standaert, F.X., Veyrat-Charvillon, N., Kamel, D., Flandre, D.: A formal study of power variability issues and side-channel attacks for nanoscale devices. In: Annual Intrl. Conf. on the Theory and Applications of Cryptographic Techniques. pp. 109–128. Springer (2011)
44. Rijsdijk, J., Wu, L., Perin, G., Picek, S.: Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis. Cryptol. ePrint Arch., Report 2021/071 (2021)
45. Riscure: CHES_CTF: trace database. [Online]http://aisylabdatasets.ewi.tudelft.nl [Accessed: Jan.8, 2024] (2018)
46. Standaert, F.X., Malkin, T.G., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Annual Intrl. Conf. on the Theory and Applications of Cryptographic Techniques. pp. 443–461. Springer (2009)

47. Wouters, L., Arribas, V., Gierlichs, B., Preneel, B.: Revisiting a methodology for efficient cnn architectures in profiling attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 147–168 (2020)
48. Wu, L., Perin, G., Picek, S.: I choose you: Automated hyperparameter tuning for deep learning-based side-channel analysis. IEEE Transactions on Emerging Topics in Computing (2022)
49. Wu, L., Perin, G., Picek, S.: On the evaluation of deep learning-based side-channel analysis. In: International Workshop on Constructive Side-Channel Analysis and Secure Design. pp. 49–71. Springer (2022)
50. Zaid, G., Bossuet, L., Habrard, A., Venelli, A.: Methodology for efficient cnn architectures in profiling attacks. IACR Trans. on Cryptographic Hardware and Embedded Systems **2020**(1), 1–36 (2020)
51. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires rethinking generalization. iclr 2017. arXiv preprint arXiv:1611.03530 (2017)
52. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning (still) requires rethinking generalization. Communications of the ACM **64**(3), 107–115 (2021)
53. Zhang, J., Zheng, M., Nan, J., Hu, H., Yu, N.: A novel evaluation metric for deep learning-based side channel analysis and its extended application to imbalanced data. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 73–96 (2020)