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

Abstract. The past years have witnessed a considerable increase in research efforts put into neural network-assisted profiled side-channel analysis (SCA). Studies have also identified challenges, e.g., closing the gap between metrics for machine learning (ML) classification and side-channel attack evaluation. In fact, in the context of NN-assisted SCA, the NN's output distribution forms the basis for successful key recovery. In this respect, related work has covered various aspects of integrating neural networks (NNs) into SCA, including applying a diverse set of NN models, model selection and training, hyperparameter tuning, etc. Nevertheless, one well-known fact has been overlooked in the SCA-related literature, namely NNs' tendency to become "over-confident," i.e., suffering from an overly high probability of correctness when predicting the correct class (secret key in the sense of SCA). Temperature* scaling is among the powerful and effective techniques that have been devised as a remedy for this. Regarding the principles of deep learning, it is known that temperature scaling does not affect the NN's accuracy; however, its impact on metrics for secret key recovery, mainly guessing entropy, is worth investigating. This paper reintroduces temperature scaling into SCA and demonstrates that key recovery can become more effective through that. Interestingly, temperature scaling can be easily integrated into SCA, and no re-tuning of the network is needed. In doing so, temperature can be seen as a metric to assess the NN's performance before launching the attack. In this regard, the impact of hyperparameter tuning, network variance, and capacity have been studied. This leads to recommendations on how network miscalibration and overconfidence can be prevented.

Keywords: Profiled Side-channel Analysis · Neural Network · Tempertature 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 [MOP08]. When executing the algorithm, such vulnerabilities are caused by the secret-key leakages, observable in the form of power consumption, timing, and electromagnetic emissions [Koc96, KJJ99, GMO01, QS01]. Profiling SCA, which is more common thanks to their effectiveness [BPS$^+$18], refers to a stronger attacker with access to an open copy of the targeted device. This copy is utilized by the attacker to construct a device profile, facilitating attacks on another device of the same type [CRR02, HGDM$^+$11, LBM15, MHM13]. Profiling attacks, involving a profiling phase followed by an attack phase, are thus known as two-stage attacks.

E-mail: snouraniboosjin@wpi.edu (Seyedmohammad Nouraniboosjin), fganji@wpi.edu (Fatemeh Ganji)

*The term "temperature" should not be confused with the degree of heat present in a substance or object. Throughout this paper, temperature is a scalar parameter defined in the field of deep learning.

In recent years, the focus has been on profiling attacks leveraging machine learning (ML) [LPB$^+$15, HZ12, HGDM$^+$11], particularly deep learning [PPM$^+$21]. These attacks are notably capable of breaking implementations equipped with countermeasures against SCA [LBM15, GHO15, CDP17, PHJ$^+$19, KPH$^+$19, WPP22a]. However, numerous questions have remained unresolved. As a prime example, related work has highlighted that performance evaluation in NN-assisted SCA is still an open problem [IUH22]. On the one hand, when testing/validating an NN, its performance is evaluated by employing classification metrics for ML, such as accuracy, loss, and recall. SCA, on the other hand, has often been assessed by considering the guessing entropy (GE) and success rate (SR) [SMY09], which have been observed to be in conflict with classification metrics [PHJ$^+$19]. To narrow this gap, studies have come up with new metrics [ZZN$^+$20, IUH22] or followed a more systematic approach to understand the root cause of this discrepancy. In the latter category, [PCP20a] has visually shown how output class probabilities are ranked for both successful and unsuccessful attacks under the condition where accuracy is not enough to make a decision about the attack's success. In fact, the accuracy can be low or close to a random guessing value, whereas expected classes are among the first ones. In this case, in accordance with the definition of GE, the summation of the probabilities of those expected classes (based on the label's guessing for the correct key) explains the success of attacks cf. [PPM$^+$21].

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 [ZBHV20, WAGP20, AGF23, WPP22a, RWPP21]. 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 an 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 [GBC16, KWPP21], the log-softmax is usually preferred cf. [Cag18], whereas [AGF23] has suggested numerically stable softmax [GBC16].

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) [GBC16]. 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) [MDP20]. This has also been empirically verified in [KWPP21].

Temperature calibration. Setting all the configuration details aside, it is known that the NNs tend to be "too confident" when predicting the classes [GPSW17, MDR$^+$21]. 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 simplest and common calibration factor is called the *temperature*, T, which "softens" the softmax in a way that $T \rightarrow 1$ indicates the estimated output probability is close to its true probability cf. [GPSW17].

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 investigate the influence of temperature calibration on the attack performance of the models in terms of GE. For this purpose, we consider three temperature calibration methods, namely Platt's multi-class calibration [GPSW17], non-optimized vector scaling (stable softmax), as well as its combination with Platt's multi-class calibration, see Section 3. We stress that our paper aims to reintroduce the concept of confidence and temperature calibration into SCA. By so doing, we have selected several publicly available NN models trained on various benchmark datasets, whose results are presented in recent studies. Those models are either already trained, or the codes for training them have been published by the authors. The ultimate goal of our work is to understand how miscalibrated these models are and if it is possible to calibrate their output distributions to improve the attack effectiveness evaluated through GE. Therefore, we do not claim that these calibrated models would generate competitive results, but we demonstrate that their results could have been improved if their out distributions had been calibrated. In this respect, our observation is that calibrated NNs can break the target with fewer traces.

- In line with the contribution above, we investigate the impact of hyperparameter tuning on the performance of the models in terms of temperature. If the search space is large or the search mechanism is not designed well, the model would not be able to break the target, even if proper objective functions are chosen for this phase. On the other hand, for the same given search space and mechanism, the tuned hyperparameters obtained for two objective functions may produce similar results. When it comes to the attack performance in terms of GE, these have been studied in the literature without giving an insight into how one can judge how well the model is configured and trained *before launching the attack*. We demonstrate that temperature can be thought of as a metric for this purpose. Thereby, we show that if proper hyperparameters are not selected, the model's temperature would be high.

- Another critical factor in calibrating the NNs is the NN's variance, i.e., the model's sensitivity to the specifics of the training data. As training NNs aims to reduce the variance, regularization techniques or simply using more training data are recommended. To examine this for NNs trained on side-channel traces, we evaluate the impact of the number of traces used in training/validation on the calibration process. Our results demonstrate that more calibrated models with lower temperatures can be achieved if more traces are used for training.

- Last but not least, with regard to the lessons learned and concepts like temperature scaling borrowed from ML, this work gives recommendations related to why accuracy may not be the right metric to assess the effectiveness of SCA, in agreement with what has been reported in the literature before. Moreover, we highlight the importance of keeping the NNs' capacity reasonably low. These recommendations help NNs used in SCA with miscalibration, resulting in 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.

[†] The codes and models are available here: https://github.com/vernamlab/CoolSCA

Metrics in SCA. Compared to the template attacks, known to be optimal from an information-theoretic perspective given a large enough number of traces, NN-assisted profiled SCA employing multi-layer perceptron (MLPs), convolutional neural networks (CNNs), and stacked autoencoders could exhibit similar or superior performance [PPM$^+$21]. Clearly, to draw any comparison like this, a proper metric should be taken into account. The differences between the ML and side-channel metrics have been investigated in several studies from different angles. Picek et al. have reported that an imbalanced class problem could be a possible reason for the inconsistency between, e.g., accuracy and the SCA attack performance [PHJ$^+$19]. This difference can be more pronounced for highly noisy traces and ones collected from protected implementations. To answer whether GE is an appropriate metric for attack evaluation, [PCP20a] has visually investigated the output class probabilities. Their observation was that the accuracy is low or close to a random guess since expected classes are not always predicted as first, but rather among the first ones. Hence, the summation of these probabilities for each key byte candidate is a valid distinguisher in line with the definition of GE. Following this line of thought cross Entropy ratio (CER) metric has been introduced, closely related to GE and Success Rate (SR) [ZZN$^+$20]. They found CER helpful in improving the attack performance, especially when dealing with imbalanced training and test datasets. It is noteworthy that here our focus is on common *attack performance metric*; therefore, other SCA-related metrics, e.g., perceived information (PI) and its extension [RSVC$^+$11, BHM$^+$19, IUH22] are not discussed in this work.

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 [Ben13]. Zaid et al. [ZBHV20] 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. [ZBHV20]. Building on Zaid's research, Wouters et al. demonstrated achieving comparable attack performance using smaller neural network designs [WAGP20]. Hyperparameter tuning has been made more automated in [WPP22a], 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 [JSH19], 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 [WPP22a]. As another example, [RWPP21] 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, [AGF23] 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, GE is one of the most commonly applied metrics when evaluating the performance of NN-assisted SCA. To compute GE, the output of the NN in terms of output probabilities is used. Apparently, tuning the hyperparameters and training the network directly impact the distribution of these probabilities and, consequently, GE. In deep learning-related literature, another parameter has been devised to assess how confident an NN is when outputting the probabilities, namely confidence. The higher the confidence, the better the NN approximated the true probability of labels, i.e., secret key. Temperature scaling is a method developed to improve the confidence of an NN after training, which means enhancing the attack performance in the context of SCA. This paper addressed the need for such a mechanism, currently lacking in the SCA-related 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 [CRR02]. 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. [BPS+18].

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

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 $\{x_1^i, x_2^i, \cdots, x_k^i\}_{i=1}^{n}$, where each trace contains k features ($k \geq 2$). 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 [BPS+18]:

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

where k^* represents the key used to acquire the profiling 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* [MPP16], 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 [RWPP21, AGF23].

Leakage models. To disclose the secret key, SCA typically considers a divide-and-conquer approach, e.g., focusing on the recovery of sub-key bytes in the case of AES. In other words, k^* in Equation 1 can represent a sub-key, and the process can be repeated for each sub-key until the (full) key is recovered. In the literature, recovering one sub-key is assumed sufficient to argue about the implementation's vulnerability to the attack [PPM+21].

For each sub-key, to mimic the physical leakage of the device, a leakage model can be defined to map the hypothetical data value to the (approximation of) leakage. The leakage can be modeled in the form of an intermediate value of the cipher leading to 256 classes, where the *identity* (ID) model is assumed. On the other hand, by considering the *Hamming weight* of the hypothetical data, it is assumed that the leakage is proportional to the number of ones in the intermediate value.

2.3 Some Relevant Concepts in ML

Negative log-likelihood (NLL) loss. NNs employ the categorical cross-entropy loss function in many applications, including SCA. NLL computes the loss L fas

$$L = -\sum_{k=1}^{n} y_k \log \hat{y}_k,$$

where y_k and $\hat{y}_k$ are the k^{th} entries in $\mathbf{y}$ and $\hat{\mathbf{y}}$, respectively. For ML classification tasks, the terms "cross-entropy" and "negative log-likelihood" are used interchangeably [Mur12].

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

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

Softmax may suffer from overflow issues (i.e., exploding gradient problem) as observed in ML- and SCA-related studies [GBC16, KWPP21]. As a remedy, a numerically stable softmax (hereafter called stable softmax) has been proposed in [GBC16], 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 datasets serving as benchmarks for evaluating the performance of NN-based SCA, namely CHES-CTF and two versions of the ASCAD dataset [BPS+17c]. The latter targets an 8-bit AVR microcontroller with a masked AES-128 implementation, where electromagnetic emanation is the observed side-channel [BPS+20]. The first dataset comprises 50,000 traces for profiling or training and 10,000 for testing (attack). In this dataset, the focus is on attacking the third key byte, which is the first masked byte. For this dataset, a specific window of 700 features is utilized, and the same key is used for training and test sets. We refer to this dataset as ASCAD-f [BPS+17a].

The second dataset of the family of ASCAD datasets that we used has 200,000 traces with random keys for profiling and 100,000 with a fixed key for testing. In this dataset, the target byte is the same, and the selected window has 1400 features. We denote this dataset as ASCAD-r [BPS+17b].

We have also used the CHES-CTF dataset [Ris18]. Released in 2018 for the Conference on Cryptographic Hardware and Embedded Systems (CHES), this dataset is associated with the CHES capture-the-flag (CTF) event. This dataset consists of 45000 traces for profiling and 5000 traces for the attack phase. All the profiling traces use a fixed key, and the attack traces also use a different fixed key. Each of the traces has 2200 features, and they are captured running a masked AES-128 encryption 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[BDTD+16]. 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. [GPSW17]) 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 as 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 predicting 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 [GPSW17], 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[P+99].

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 corresponds 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 [P+99] is performed by training a logistic regression model on the validation set to learn a scaler $T > 0$, 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 [AGF23]. 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 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 [GPSW17]. We empirically examine if using stable softmax or coupling temperature calibration with non-optimized vector scaling would be useful in 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 [GPSW17]. 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 details our experimental results of the temperature calibration method. After introducing the experimental setup, the results for the impact of temperature calibration on the GE are presented in Section 4.2, whereas the effects of the number of validation/training traces, selecting the hyperparameters and its objective function are analyzed in Sections 4.3-4.4.

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 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 [GJS19]. 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 [WPP22a, PCP20a, WAGP20, ZBHV20]. Among the models used in our paper, some trained models have been available [COS20], whereas in other cases, we trained the model by using the available codes in [LW22, PCP20b]. 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. In the models

Table 1: MLP Models. Here FC(#neurons) denotes a fully-connected layer with the number of neurons is given in parentheses. SM(#classes) shows the number of classes at the softmax layer.

Data set	Leakage model	Architecture
ASCADf	ID [WPP22a]	FC(300),FC(300),FC(100), FC(100),FC(100),FC(100), SM(256)
	HW [WPP22a]	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 [WPP22a]	FC(400),FC(400),FC(100),FC(100),FC(100),FC(100), Softmax(256)
	HW [WPP22a]	FC(200),FC(200),FC(100),FC(100),FC(100),FC(100), FC(100),FC(100), SM(9)
CHES-CTF	HW [WPP22a]	FC(400),FC(400),FC(100),FC(100),FC(100), SM(9)

Table 2: CNN Models used in our experiments. Here C(filters, kernel size, strides), P(size, stride), and M(size, stride) show the hyperparameters for a convolutional layer, average pooling, and max pooling. FLAT denotes a flatten layer, whereas FC(#neurons) denotes a fully connected layer with the number of neurons given in parentheses. SM(#classes) shows the number of classes at the softmax layer.

Data set	Leakage model	Architecture
ASCADf	ID [WAGP20]	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 [PCP20a]	C(16,18,1), FLATT, FC(600), FC(600), SM(9)
ASCADr	ID [WPP22a]	C(120,3,1), P(32,2), C(8,1,1), P(32,2), FLAT, FC(30), FC(5), FC(5), SM(256)
	HW [WPP22a]	C(4,3,1), P(30,2), FLAT, FC(30), FC(20), FC(20), SM(9)
CHES-CTF	HW [WPP22a]	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)

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 traces. Validation sets are randomly taken from profiling traces. 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, softmaxfunction 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 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 [MDR+21].

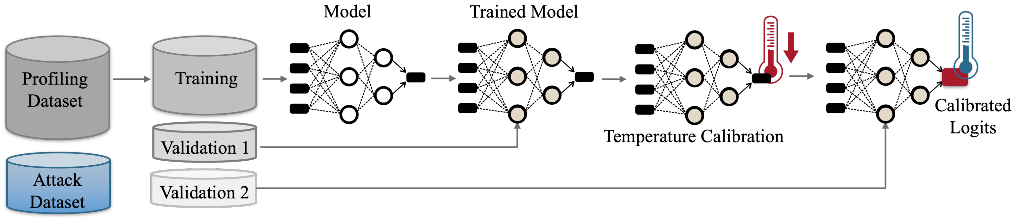


Figure 1: Experimental setup. The validation sets are taken randomly from the profiling set to perform temperature calibration by using the Validation 1 dataset, as explained in Section 3. During the attack phase, the logits are calibrated using the temperature learned in the evaluation step, and then the probabilities generated by softmax/stable softmax are fed to the rank function as usual. During this process, no parameter or hyperparameter of the NN is changed. The dataset "Validation 2" is solely used to measure the temperature of the calibrated model to assess the performance of the temperature scaling.

Table 3: T_{GE0} and temperature of the trained models before and after temperature calibration. T indicates the temperature with $T \approx 1$ showing that the model is well calibrated. It is worth mentioning that these results are obtained by calibrating the model on a portion of the profiling dataset that contains (approximately) one-fourth of the number of profiling traces in each dataset (see Section 4.3 for results with 2000 validation traces that better match the results in respective studies). 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.

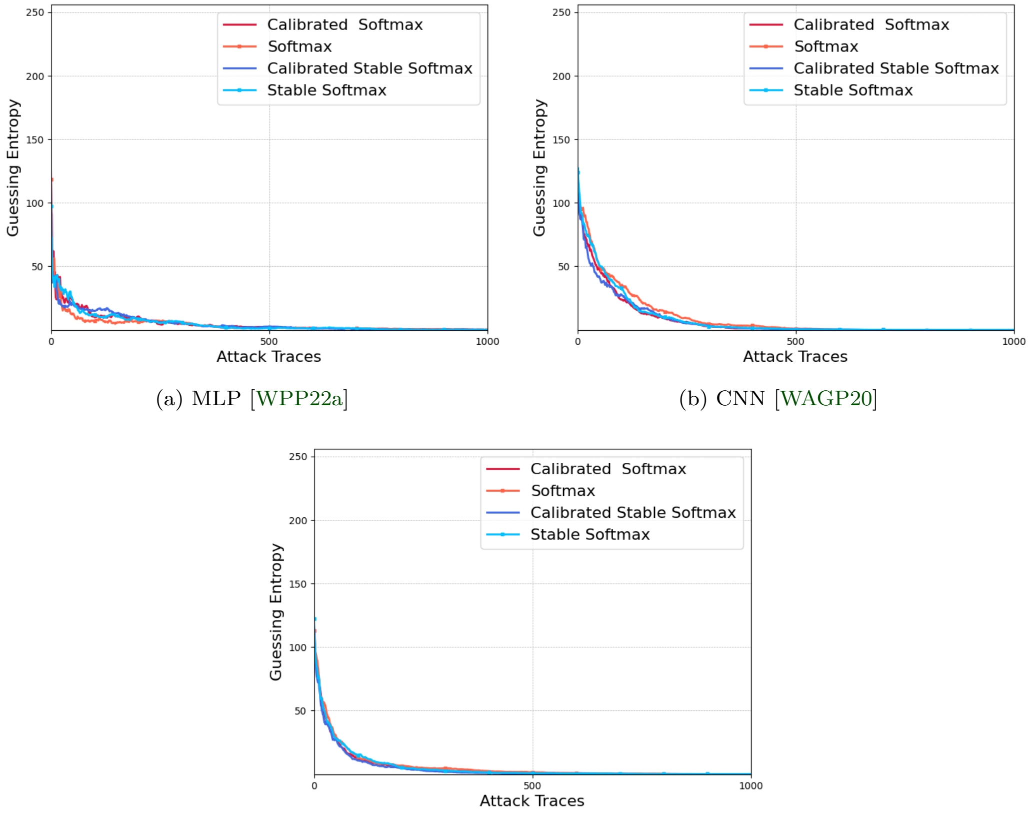
Dataset	Model	Leakage model	Uncalibrated		Calibrated	
			T	T_{GE0}	T	T_{GE0}
ASCADf	MLP	ID [WPP22a]	4.110	733	1.042	662
		HW [WPP22a]	6.470	3368	0.950	2736
	CNN	ID [WAGP20]	3.677	508	0.910	442
		ID [ZBHV20]	2.534	587	0.927	511
		HW [PCP20a]	1.601	1922	1.015	1895
ASCADr	MLP	ID [WPP22a]	3.05	1575	1.026	1025
		HW [WPP22a]	7.976	3443	0.977	2686
	CNN	ID [WPP22a]	1.165	324	1.014	281
		HW [WPP22a]	1.670	1851	0.983	1615
CHES-CTF	MLP	HW [WPP22a]	15.194	4403	1.028	3381
	CNN	HW [WPP22a]	24.707	4706	0.956	4550

4.2.1 ASCAD with the fixed key

We begin with ASCAD with the fixed key (ASCAD-f) dataset that is relatively easier to break [WPP22a].

ID leakage model. In Figure 2, 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 [WPP22a, WAGP20, ZBHV20], the trend 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, 442 for Wouters' model, and 511 for Zaid's model. The MLP model [WPP22a] had a temperature of 4.11. For the CNN models that we considered in this subsection, the temperature of the Wouters' model [WAGP20] was 3.677, and the temperature of the Zaid's model [ZBHV20] was 2.534. This comparison is indeed interesting as Wouters' CNN has a smaller capacity than Zaid's model. If the



(a) MLP [WPP22a]

(b) CNN [WAGP20]

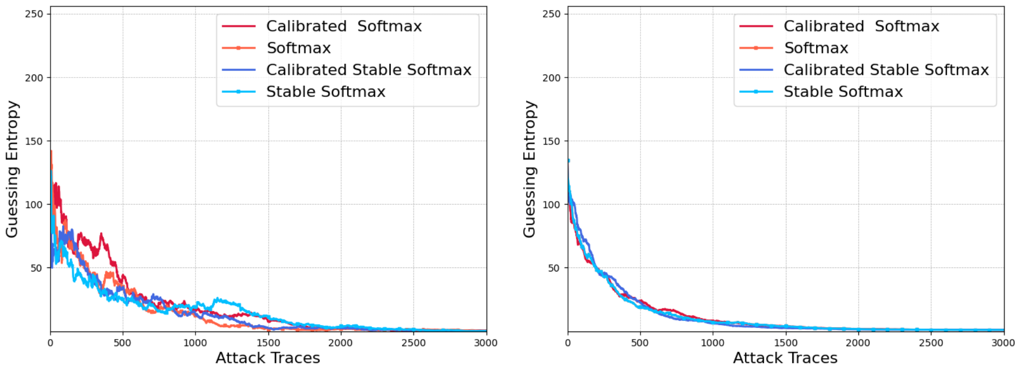
(c) CNN [ZBHV20]

Figure 2: GE of ASCAD-f with the ID leakage model.

models were overparameterized, it was expected that the temperature of Zaid's CNN would have been higher, which is not the case. Hence, one can draw the conclusion that both models exhibit a reasonable size.

ID leakage model with stable softmax vs. calibrated stable softmax. In the MLP model, compared to the results for temperature scaling presented in Table 4.2, 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 Wouters' model [WAGP20], the stable softmax achieved $GE = 0$ by utilizing 462 traces before calibration, and this number decreased to 409 traces after calibration. For Zaid's model, the number of traces needed for $GE = 0$ were 477 and 386, before and after calibration in stable softmax. 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.

HW leakage model. Figure 3 depicts GE for both MLP model [WPP22a], and the CNN model [PCP20a], 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.



(a) MLP [WPP22a]

(b) CNN [PCP20a]

Figure 3: GE of ASCAD-f with the HW leakage model.

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

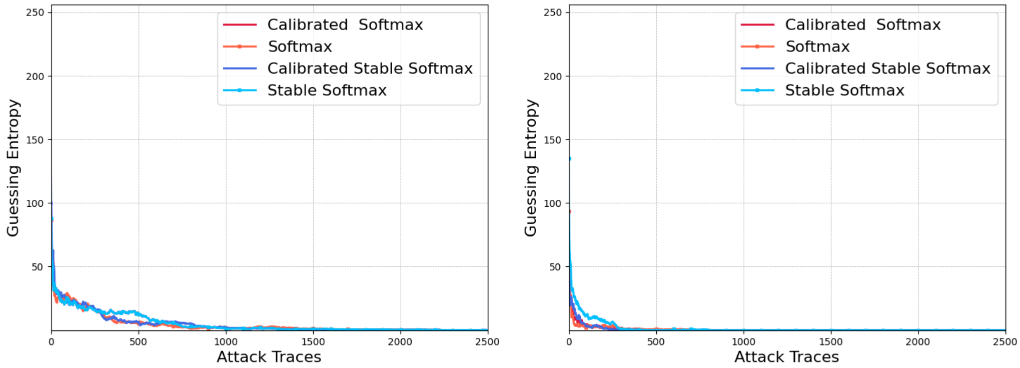
4.2.2 ASCAD with the random key

ID leakage model. GE results of the models for the ASCAD dataset with the random key (ASCAD-r) and ID leakage model are presented in Figure 4. Here, the CNN model [WPP22a] 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 [WPP22a], 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.

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.

HW leakage model. Figure 5 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 [WPP22a] model were better, resulting in a lower temperature and better calibration. The temperature before calibration was 1.670 for the CNN model, whereas for the MLP [WPP22a], this number was 7.976 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.

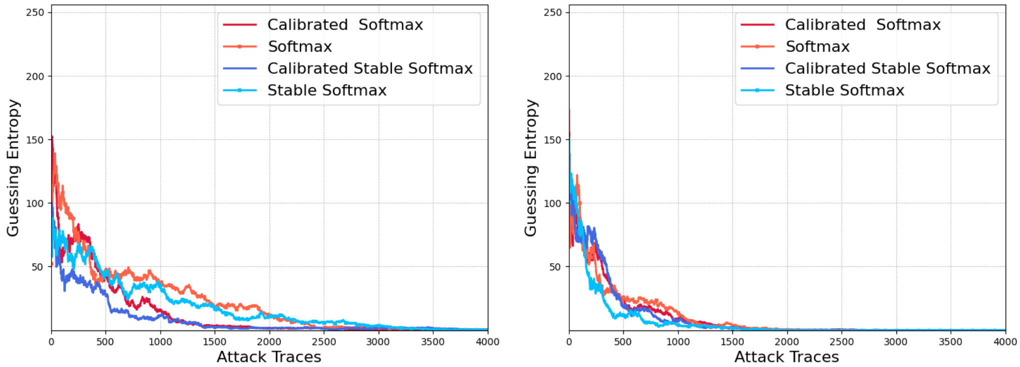
HW leakage model with stable softmax vs. calibrated stable softmax. For the MLP, the results obtained for stable softmax also improved by calibration, reducing the T_{GE0} from 3447 to 2510. The same holds for the CNN models: the calibrated model required 1615 traces to reach the $GE = 0$ compared to 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.



(a) MLP [WPP22a]

(b) CNN [WPP22a]

Figure 4: GE of ASCAD-r with the ID leakage model.



(a) MLP [WPP22a]

(b) CNN [WPP22a]

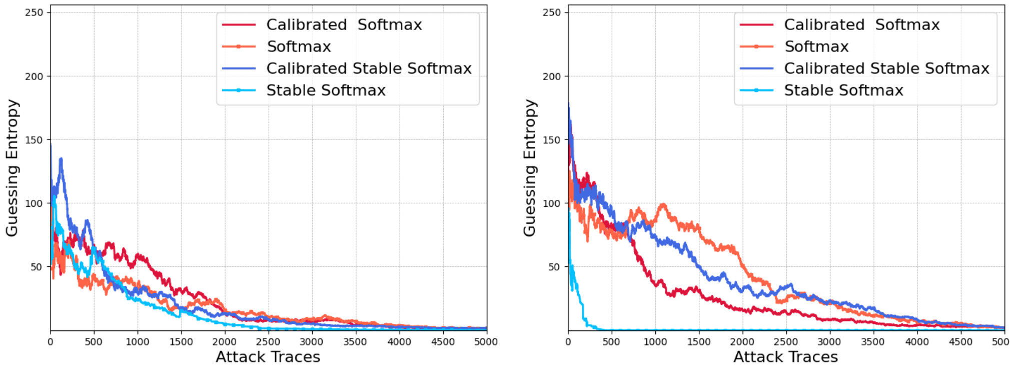
Figure 5: GE of ASCAD-r with the HW leakage model.

4.2.3 CHES-CTF

HW leakage model. Next, we discuss the results of the CHES-CTF dataset. For this dataset, the temperature of both the MLP and CNN models [WPP22a] 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 model and 3381 after calibration3. 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 Figure 3.

HW leakage model with stable softmax vs. calibrated stable softmax. The models incorporating the stable softmax performed significantly better than softmax, as shown in Figure 6. T_{GE0} for the MLP model was 2374 and 328 for the CNN. As shown in Figure 6, 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 2-6 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



(a) MLP [WPP22a]

(b) CNN [WPP22a]

Figure 6: GE of CHES-CTF with the HW leakage model.

Table 4: 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 [WPP22a]	2.953	449	295
		HW [WPP22a]	1.604	432	246
	CNN	ID [WAGP20]	1.513	226	190
		ID [ZBHV20]	1.435	162	120
		HW [PCP20a]	1.605	1922	1895
ASCADr	MLP	ID [WPP22a]	3.08	1492	1137
		HW [WPP22a]	7.216	1152	1164
	CNN	ID [WPP22a]	1.120	347	269
		HW [WPP22a]	1.794	1376	1038
CHES-CTF	MLP	HW [WPP22a]	15.263	2198	768
	CNN	HW [WPP22a]	25.17	5000	4913

scaling and stable softmax (marked as calibrated stable softmax) would always be helpful.

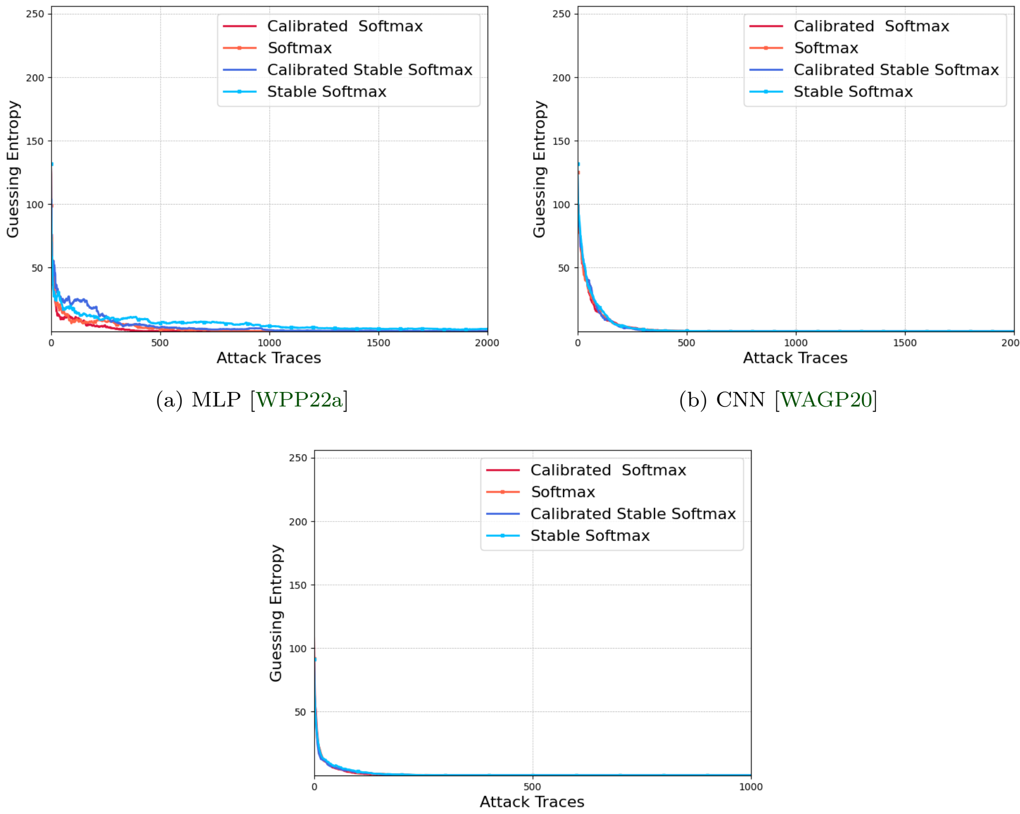
4.3 Temperature Calibration: Impact of the Number of Validation Traces

The main objective of this section 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 4.2. This reduces the model's variance, and therefore, it is expected that the temperature of uncalibrated models will be reduced.

4.3.1 ASCAD with the fixed key

This dataset contains 50,000 profiling traces. To achieve the results presented in Section 4.2, 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. Figure 7 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 Wouters' model[WAGP20] and 1.435 for Zaid's model [ZBHV20] (see Table 3 and Table 4). Compared to the results in Table 3, Wouters' model [WAGP20] reached lower ranks with fewer traces: $T_{GE0} = 190$ for the



(a) MLP [WPP22a]

(b) CNN [WAGP20]

(c) CNN [ZBHV20]

Figure 7: GE of ASCAD-f with the ID leakage model and 2000 traces in the validation set.

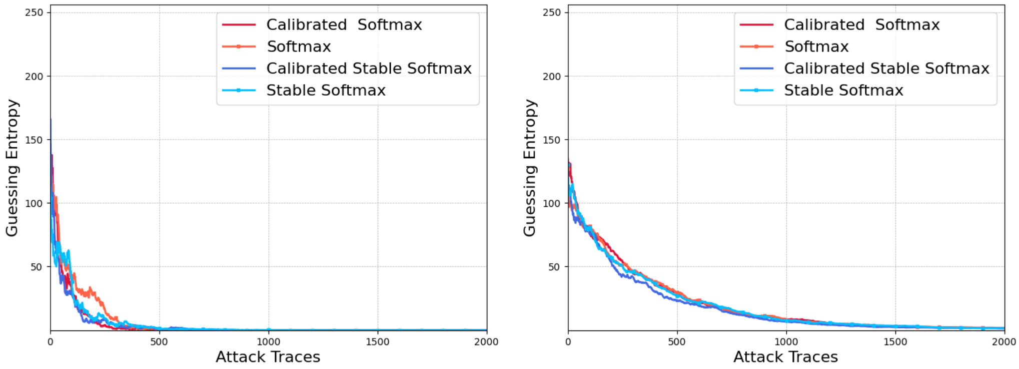
calibrated softmax, while it was 226 before calibration. Similarly, T_{GE0} for the model with stable softmax was also improved after calibration from 280 to 226. Considering Zaid's model, T_{GE0} was 162 for the model with softmax and 156 for the one with stable softmax, and after calibration, these numbers reduced to 120 and 149, respectively. The MLP model [WPP22a] 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 Figure 8 present the results obtained for models with the HW leakage model. For the CNN model [PCP20a], 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 [WPP22a](see table 4). 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 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.

4.3.2 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.

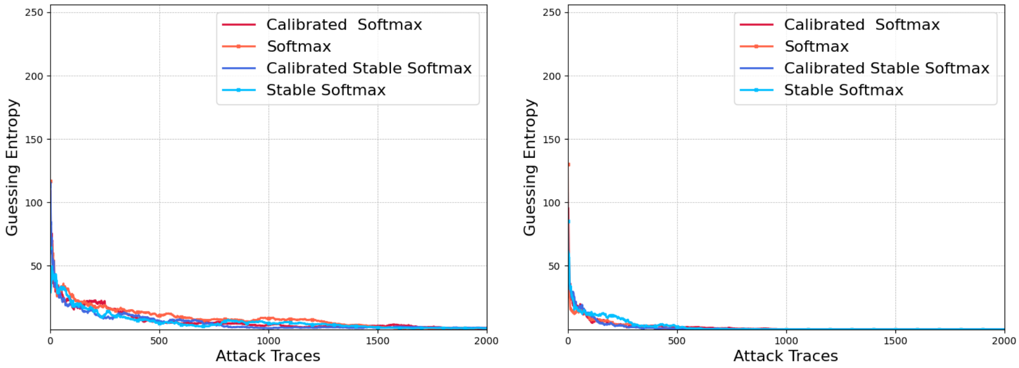
ID leakage model. For the ASCAD-r dataset, using the ID leakage model, the MLP model [WPP22a] temperature was 3.08, slightly higher than the previous 3.05, see Table 3,



(a) MLP [WPP22a]

(b) CNN [PCP20a]

Figure 8: GE of ASCAD-f with the HW leakage model and 2000 traces in the validation set.



(a) MLP [WPP22a]

(b) CNN [WPP22a]

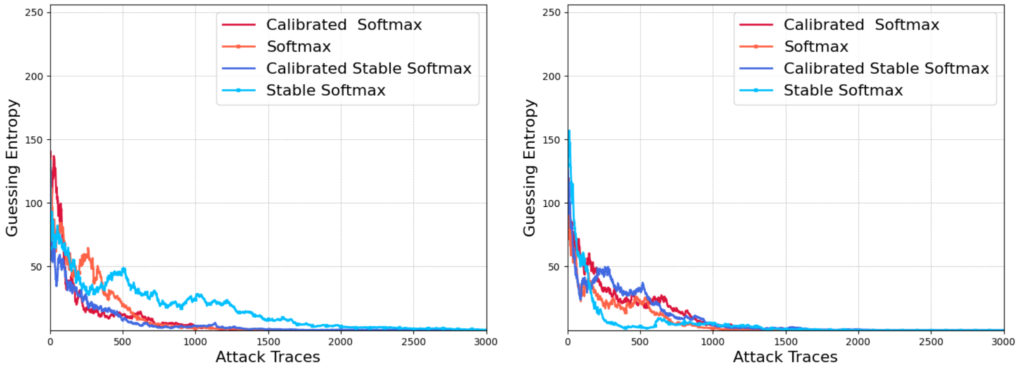
Figure 9: GE of ASCAD-r with the ID leakage model and 2000 traces in the validation set.

which is not statistically significant. Similarly, the CNN model [WPP22a] 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 Figure 9).

HW leakage model. Next, as shown in Figure 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 4.2, 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 softmax could also achieve competitive results, while for MLP ones, calibrated softmax was advantageous.

4.3.3 CHES-CTF dataset

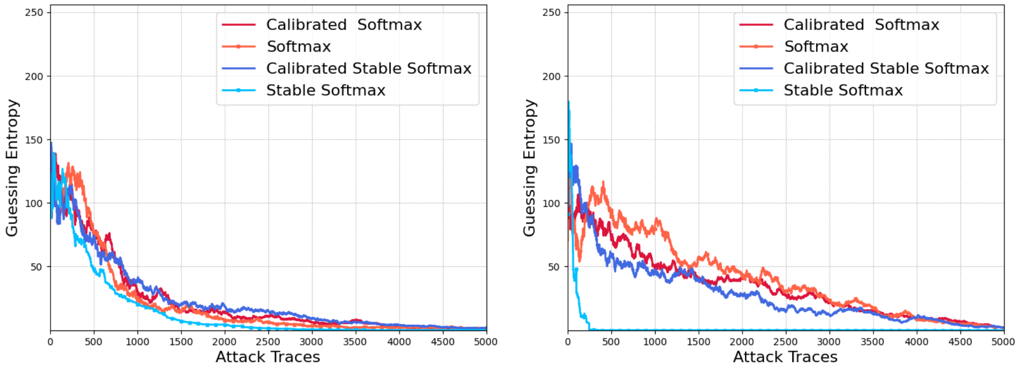
Finally, we can see the GE curves of the models trained on the CHES-CTF dataset in Figure 11. Here, the temperature of the CNN and MLP was 25.17 and 15.263, respectively. High temperature and poor performance indicate that the models suffer from overfitting. Interestingly enough, stable softmax outperformed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



(a) MLP [WPP22a]

(b) CNN [WPP22a]

Figure 10: GE of ASCAD-r with the HW leakage model and 2000 traces in the validation set.



(a) MLP [WPP22a]

(b) CNN [WPP22a]

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

softmax itself.

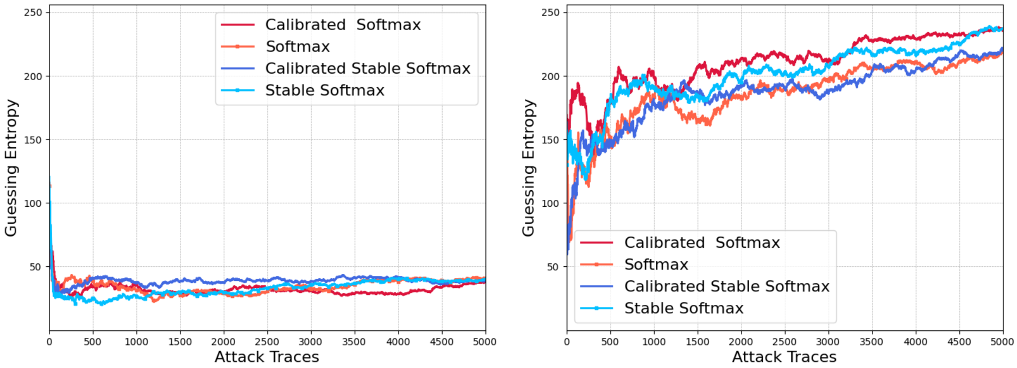
Summary. The take-home message from the experiments, whose results are presented in this section, is as follows. In the context of SCA, where the profiling datasets are not as large as what can be found in other deep learning applications, it is recommended not to limit the number of training traces to allocate more traces for validation. Both stable softmax (non-optimized vector scaling) and temperature scaling effectively improve the attack performance. Increasing the number of profiling traces may result in overfitting, as observed for the CHES-CTF dataset. Therefore, monitoring the NLL and accuracy during training is, therefore recommended.

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, Figure 12 shows the results for the models whose hyperparameters

Table 5: Models with improper hyperparameters (notations are similar to Tables 1-2).

Model	Architecture
MLP [WPP22a]	FC(200), FC(200), FC(100), FC(100), FC(100), FC(100), FC(100), SM(256)
CNN [WPP22a]	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 [WPP22a]

(b) CNN with accuracy [WPP22a]

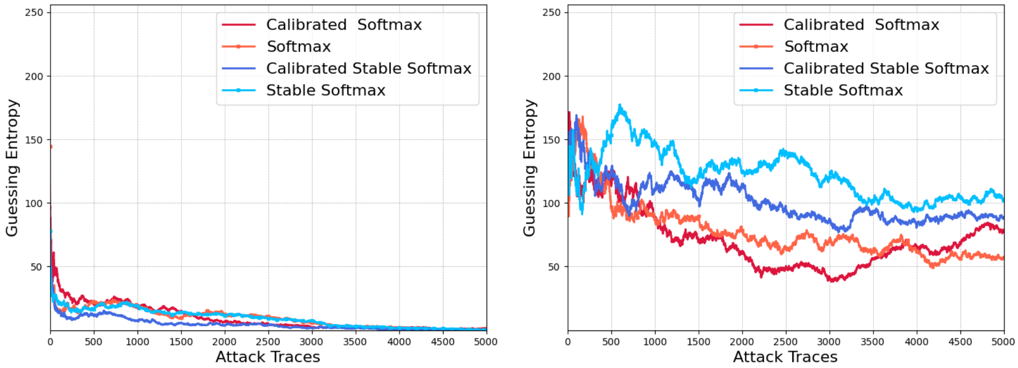
Figure 12: GE of the models with improper hyperparameters for ASCAD-r and ID leakage model.

were attempted to tune. In these cases, the range of hyperparameters was unfit, leading to a failure in tuning. Table 5 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 [WPP22a], the temperature was 14.84 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.

4.4.1 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. [WPP22a] utilized in their research: accuracy, key rank, and L_m [WPP22b]. 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 [WPP22b]. Wu et al. have demonstrated that choosing these objective functions has impacted the attack performances for different models and datasets [WPP22a]. 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. Figure 13a 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 Figure 9, 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. Figure 13b shows the GE for this model, which is



(a) MLP with L_m [WPP22a]

(b) CNN with L_m [WPP22a]

Figure 13: GE of the models with wrong objective functions for ASCAD-r and ID leakage model.

not as good as the results in Figure 9. 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 that.

5 Discussion

5.1 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) [RSVC+11, MDP20, BHM+19]. 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 $\pi(S|X)$ [GPSW17]. 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. [MDP20].

Consider a scenario where NLL reaches a minimum value. If the training is not stopped, the model still can improve its accuracy, 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. [GPSW17]. In the context of SCA, if probabilities are not well modeled, the attack performance in terms of GE degrades [PCP20a]. This clarifies what has been empirically verified in SCA-related literature [PCP20a], i.e., accuracy might not reflect the performance of the attacks as GE does.

To address overfitting to NLL, two major actions can be carried out.

- First, the training can be stopped before changing the training regime to NLL overfitting. In this sense, NLL itself calibrates the NN's output probability. This can be performed by observing the NLL as a stopping criterion. Another possible approach is to stop the model by encompassing criteria, which measure the information the NN extracts from the input, indirectly minimizing PI [AGF23].

- It is also possible to apply temperature scaling as proposed in this paper to calibrate the NN's output probability and its confidence. This can be the most straightforward to incorporate into a neural network training pipeline.

5.2 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 [ZBHV20, WAGP20, AGF23], mainly to reduce computational complexity and memory footprint. On the other hand, studies, e.g., [WPP22a], 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 [ZBH+17, ZBH+21]. 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 5.1, 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.

6 Conclusion and Future Work

In the context of SCA, the problem tackled in this paper is that NNs tend to be excessively confident in their predictions. We demonstrate that the methods developed in deep learning to address the issue with NNs' overconfidence can be leveraged to improve the attack performance in terms of GE. In this regard, our work focuses on temperature scaling, which can be easily integrated into SCA without reconfiguring or retuning the NN. Another problem identified in studies on NN-assisted SCA is that it is not straightforward to assess the performance of the hyperparameter tuning before launching an attack. The concept of temperature scaling is indeed useful in this matter.

To evaluate the effectiveness of our approach, we used publicly available NNs that were trained on various benchmark datasets. The temperature of these models was calculated to see how miscalibrated they are. The results of taking our approach indicate that the number of attack traces needed to break the target is reduced (reaching $GE = 0$). We also examined the impact of the number of training traces and hyperparameter tuning on the performance of the attack and how the models' temperature can reflect that.

The requirement for a hold-out validation dataset contributes to the cost of our approach. Consequently, in scenarios where datasets have a limited number of profiling traces, there are fewer validation traces available to perform temperature calibration. Nevertheless, with respect to SCA, collecting profiling traces from the open copy of the device might not be an issue.

We will shift our focus to a calibration method that relies on a divide-and-conquer strategy in the future. In this work, we used a single scalar temperature for all the classes, while using one temperature for each class can help calibrate the models more efficiently. Moreover, examining other types of model calibration could also be a viable option for future research.

7 Acknowledgments

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

References

[AGF23] Rabin Y Acharya, Fatemeh Ganji, and Domenic Forte. Information theory-based evolution of neural networks for side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 401–437, 2023.

[BDTD+16] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[Ben13] Yoshua Bengio. Deep learning of representations: Looking forward. In *International conference on statistical language and speech processing*, pages 1–37. Springer, 2013.

[BHM+19] Olivier Bronchain, Julien M Hendrickx, Clément Massart, Alex Olshevsky, and François-Xavier Standaert. Leakage certification revisited: Bounding model errors in side-channel security evaluations. In *Annual Intrl. Cryptol. Conf.*, pages 713–737. Springer, 2019.

[BPS+17a] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. 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.

[BPS+17b] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. 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.

[BPS+17c] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. ASCADv1 Dataset: the atmega8515 sca campaigns. [Online] https://github.com/ANSSI-FR/ASCAD/tree/master/ATMEGA_AES_v1 [Accessed: Jan.8, 2024], 2017.

[BPS+18] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. Study of deep learning techniques for side-channel analysis and introduction to ascad database. 2018.

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

[Cag18] Eleonora Cagli. *Feature extraction for side-channel attacks.* PhD thesis, Sorbonne université, 2018.

[CDP17] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures. In *Intrl. Conf. on Cryptographic Hardware and Embedded Systems*, pages 45–68. Springer, 2017.

[COS20] KU Leuven COSIC. TCHES20V3 CNN SCA Repository: cnn-based side-channel analysis. [Online]https://github.com/KULeuven-COSIC/TCHES20V3_CNN_SCA [Accessed: Jan. 8, 2024], 2020.

[CRR02] Suresh Chari, Josyula R Rao, and Pankaj Rohatgi. Template attacks. In *Intrl. Workshop on Cryptographic Hardware and Embedded Systems*, pages 13–28. Springer, 2002.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[GHO15] Richard Gilmore, Neil Hanley, and Maire O'Neill. Neural network based attack on a masked implementation of aes. In *Intrl. Symposium on Hardware Oriented Security and Trust (HOST)*, pages 106–111. IEEE, 2015.

[GJS19] Aron Gohr, Sven Jacob, and Werner Schindler. Ches 2018 side channel contest ctf-solution of the aes challenges. *Cryptology ePrint Archive*, 2019.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In *Cryptographic Hardware and Embedded Systems—CHES 2001*, pages 251–261. Springer, 2001.

[GPSW17] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.

[HGDM$^+$11] Gabriel Hospodar, Benedikt Gierlichs, Elke De Mulder, Ingrid Verbauwhede, and Joos Vandewalle. Machine learning in side-channel analysis: a first study. *Journal of Cryptographic Engineering*, 1(4):293, 2011.

[HZ12] Annelie Heuser and Michael Zohner. Intelligent machine homicide. In *Intrl. Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 249–264. Springer, 2012.

[IUH22] Akira Ito, Rei Ueno, and Naofumi Homma. Perceived information revisited: New metrics to evaluate success rate of side-channel attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 228–254, 2022.

[JSH19] Haifeng Jin, Qingquan Song, and Xia Hu. Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1946–1956, 2019.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings 19*, pages 388–397. Springer, 1999.

[Koc96] Paul C Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Annual International Cryptology Conference*, pages 104–113. Springer, 1996.

[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 Trans. on Cryptographic Hardware and Embedded Systems*, pages 148–179, 2019.

[KWPP21] Maikel Kerkhof, Lichao Wu, Guilherme Perin, and Stjepan Picek. No (good) loss no gain: Systematic evaluation of loss functions in deep learning-based side-channel analysis. *IACR Cryptol. ePrint Arch., 2021/1091*, 2021.

[LBM15] Liran Lerman, Gianluca Bontempi, and Olivier Markowitch. A machine learning approach against a masked aes. *Journal of Cryptographic Engineering*, 5(2):123–139, 2015.

[LPB$^+$15] Liran Lerman, Romain Poussier, Gianluca Bontempi, Olivier Markowitch, and François-Xavier Standaert. 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*, pages 20–33. Springer, 2015.

[LW22] Stjepan Picek Lichao Wu, Guilherme Perin. AutoSCA: automated hyperparameter tuning for deep learning-based side-channel analysis. [Online] https://github.com/AISyLab/AutoSCA [Accessed: Jan. 8, 2024], 2022.

[MDP20] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. A comprehensive study of deep learning for side-channel analysis. *IACR Trans. on Cryptographic Hardware and Embedded Systems*, pages 348–375, 2020.

[MDR$^+$21] Matthias Minderer, Josip Djolonga, Rob Romijnders, Frances Hubis, Xiaohua Zhai, Neil Houlsby, Dustin Tran, and Mario Lucic. Revisiting the calibration of modern neural networks. *Advances in Neural Information Processing Systems*, 34:15682–15694, 2021.

[MHM13] Zdenek Martinasek, Jan Hajny, and Lukas Malina. Optimization of power analysis using neural network. In *Intrl. Conf. on Smart Card Research and Advanced Applications*, pages 94–107. Springer, 2013.

[MOP08] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks: Revealing the secrets of smart cards*, volume 31. Springer Science & Business Media, 2008.

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

[Mur12] Kevin P Murphy. *Machine Learning: A Probabilistic Perspective.* MIT press, 2012.

[P$^+$99] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[PCP20a] Guilherme Perin, Łukasz 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*, pages 337–364, 2020.

[PCP20b] Guilherme Perin, Łukasz Chmielewski, and Stjepan Picek. 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.

[PHJ+19] Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni. The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. *IACR Trans. on Cryptographic Hardware and Embedded Systems*, 2019(1):1–29, 2019.

[PPM+21] Stjepan Picek, Guilherme Perin, Luca Mariot, Lichao Wu, and Lejla Batina. SoK: Deep learning-based physical side-channel analysis. *IACR Cryptol. ePrint Arch., 2021/1092*, 2021.

[QS01] Jean-Jacques Quisquater and David Samyde. 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*, pages 200–210. Springer, 2001.

[Ris18] Riscure. CHES_CTF: trace database. [Online]http://aisylabdatasets.ewi.tudelft.nl [Accessed: Jan.8, 2024], 2018.

[RSVC+11] Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre. 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*, pages 109–128. Springer, 2011.

[RWPP21] Jorai Rijsdijk, Lichao Wu, Guilherme Perin, and Stjepan Picek. Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis. *Cryptol. ePrint Arch., Report 2021/071*, 2021.

[SMY09] François-Xavier Standaert, Tal G Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In *Annual Intrl. Conf. on the Theory and Applications of Cryptographic Techniques*, pages 443–461. Springer, 2009.

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

[WPP22a] 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*, 2022.

[WPP22b] Lichao Wu, Guilherme Perin, and Stjepan Picek. On the evaluation of deep learning-based side-channel analysis. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 49–71. Springer, 2022.

[ZBH+17] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. iclr 2017. *arXiv preprint arXiv:1611.03530*, 2017.

[ZBH+21] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[ZBHV20] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Methodology for efficient cnn architectures in profiling attacks. *IACR Trans. on Cryptographic Hardware and Embedded Systems*, 2020(1):1–36, 2020.

[ZZN+20] Jiajia Zhang, Mengce Zheng, Jiehui Nan, Honggang Hu, and Nenghai Yu. A novel evaluation metric for deep learning-based side channel analysis and its extended application to imbalanced data. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 73–96, 2020.