Learning when to stop: a mutual information approach to fight overfitting in profiled side-channel analysis

Guilherme Perin[1], Ileana Buhan[1] and Stjepan Picek[2]

[1] Riscure BV
[2] Delft University of Technology

Abstract. Today, deep neural networks represent a common option when conducting the profiled side-channel analysis. Such techniques commonly do not require pre-processing, and yet, they can break targets that are even protected with countermeasures. Unfortunately, it is usually far from trivial to find neural network hyper-parameters that would result in such top-performing attacks. The hyper-parameter leading the training process is the number of epochs during which the training happens. If the training is too short, the network does not reach its full capacity, while if the training is too long, the network overfits, and consequently, is not able to generalize to unseen examples. Finding the right moment to stop the training process is particularly difficult for side-channel analysis as there are no clear connections between machine learning and side-channel metrics that govern the training and attack phases, respectively.
In this paper, we tackle the problem of determining the correct epoch to stop the training in deep learning-based side-channel analysis. First, we explore how information is propagated through the hidden layers of a neural network, which allows us to monitor how training is evolving. Second, we demonstrate that the amount of information transferred to the output layer can be measured and used as a reference metric to determine the epoch at which the network offers optimal generalization. To validate the proposed methodology, we provide extensive experimental results that confirm the effectiveness of our metric of choice for avoiding overfitting in the profiled side-channel analysis.

Keywords: Side-channel Analysis · Neural Networks · Overfitting · Mutual Information · Information Bottleneck

1 Introduction

Profiled side-channel attacks (SCAs) can determine the worst-case security of protected cryptographic implementations. There, the attack model assumes an adversary with full control of a device identical to the target one. Various techniques like template attacks [CRR02], linear regression [SLP05] and machine learning [LMBM13, MZVT15] belong to profiled SCAs and can achieve strong performance. More recently, deep learning techniques proved to be even more potent as 1) they do not need to use the pre-processing phase to select the points of interest, and 2) they perform well even in the presence of noise and countermeasures. Although the application of deep learning for profiled SCA became very prominent, there are still open questions like the selection of hyper-parameters for side-channel attacks. For hyper-parameters like architectural details (e.g., number of layers, neurons), recent works provide certain directions to follow [ZBHV19]. At the same time, the ability to select the right moment to stop the training phase is left to experimentalism

(and sometimes, even luck) as there does not seem to be any direct connection between the machine learning metrics and the performance of a side-channel attack [PHJ+19].

While this lack of clear connection may seem relevant but not crucial, we note that commonly, one aims to stop the training phase once metrics like accuracy or recall start to degrade for a validation set. If there is no connection with the side-channel attack performance, we cannot know if the training phase stopped too late and machine learning model overfits. Then, as a consequence of too late stopping, the developed model will not generalize to unseen data, and will not perform well in the attack [1].

One way to explain generalization and training aspects of a deep neural network is through the lens of information theory. In [ST17], the authors propose a new methodology to interpret the training of multilayer perceptron (MLP) through a theory called the *Information Bottleneck* (IB) [TZ15]. There, they demonstrate that the training of an MLP provides two distinct phases - *fitting* and *compression*. These phases are determined by computing the mutual information between the intermediate representations (activations from hidden layers), and input (raw data) and output (labels). This way, an output from a hidden layer can be seen as a summary of statistics containing information about the input and output. The fitting phase is usually very fast, requiring only a few epochs, while the compression phase lasts longer. The compression phase is also the one responsible for the generalization of the neural network, i.e., its ability to perform on unseen data.

In this work, we consider the mutual information between output layer activations (*Softmax*) and the data labels (as given by the leakage model) as a metric to identify the epoch at which the neural network achieves its optimal generalization capacity. We empirically demonstrate that this metric provides better attack performance for profiled attacks against masked AES implementations when compared to usual metrics like accuracy or loss. More specifically, our results emphasize that training a neural network for too many epochs may affect generalization, and early stopping based on the mutual information metric is a reliable way to avoid this scenario. To the best of our knowledge, this is the first result providing a reliable attack performance metric different from conducting an actual attack (key ranking). While key ranking is a reliable validation metric to optimize the generalization of a deep neural network for side-channel attacks [CDP17, Per19], it brings significant computational overheads when using large validation sets. Mutual information, on the other hand, offers remarkable performance at a fraction of computational cost as it does not have to be computed for all key hypothesis. To facilitate reproducible research, we make the source code publicly available [Ano20].

The rest of this paper is organized as follows. Section 2 provides background information about the profiled side-channel analysis, deep learning, and information theory. Related works are presented in Section 3. The theory of information bottleneck and information plane for deep learning is provided in Section 4, where we describe the main contribution of this work. Section 5 provides empirical validation of the proposed analysis based on information theory to increase the performance of deep learning-based side-channel attacks. Finally, conclusions and possible future research directions are given in Section 6.

2 Background

This section provides information about profiled side-channel attacks and deep learning-based SCA. Afterward, the basic concepts of information theory are defined.

[1] It is also possible for a machine learning model to underfit if the training stopped too early. Still, this is usually of less concern as the resulting machine learning model would generalize to unseen data but just not use its full potential, i.e., the attack would not be as powerful as possible.

2.1 Profiled Attacks and Deep Learning-based SCA

Side-channel analysis (SCA) can be divided into non-profiled and profiled side-channel analysis. Standard non-profiled side-channel attacks like correlation power analysis [BCO04] or differential power analysis [KJJ99] explore the leakage from univariate statistics. These attacks are usually limited by the selection of a function defining the leakage model. They are also defined as the first-order side-channel attacks. If more than one variable needs to be explored from the non-profiled scenario, higher-order attacks can combine several samples, and the analysis usually requires more complex attacks.

The second approach, the so-called profiled attack, assumes that an adversary has an open cryptographic device from where a model can be learned by freely setting the key and the input data (plaintext or ciphertext). These attacks also explore the multivariate nature of the leakage and usually require features/points of interest selection for dimensionality reduction. The learned model is tested against side-channel traces collected from an identical target device. This second part is known as matching or attack phase and it returns the key hypothesis with the maximum likelihood estimation. Standard profiled attacks are template attacks [CRR02], linear regression [SLP05], and machine learning [LMBM13, MHM13].

Profiled attacks have different requirements for points of interest selection and leakage model, which is usually performed using the probability density function (*pdf*). The adoption of deep neural networks as a profiled attack paradigm can provide automated points of interest selection where recent results suggest such techniques are less sensitive to trace misalignment and countermeasures based on the first order-masking for software AES implementations [CDP17, KPH+19, PSB+18].

2.1.1 Deep Learning in the Context of SCA

A deep neural network can be seen as a system that accepts a random variable input X (a side-channel trace) and provides, at its output, a probability estimation for X, $\hat{Y}$, based on which the class label Y can be estimated with a decision rule. The size of the input vector X is the same as the number of samples/features in a side-channel trace. The size of the vector Y is directly derived from the leakage model selected according to the target cryptographic function (e.g., S_{box} output in the first encryption or decryption AES round). In the side-channel analysis, the trained neural network is then tested against a set of side-channel traces, in which the underlying key is unknown, and the key recovery methodology assumes that the correct key is the one that maximizes the cumulative probabilities for each key byte candidate:

$$P(g) = \sum_{i=1}^{N_T} log(p_i(c)^g). \tag{1}$$

The quantity $p_i(c)^g$ represents the neuron's activation value from the *Softmax* output layer corresponding to the neuron c that is a label associated to the current trace i based on the key byte guess g.

2.1.2 Training a Neural Network

The process of training a deep neural network is an optimization problem based on the minimization of a selected loss function based on the stochastic gradient descent algorithm. For side-channel analysis, we are interested in a model that is capable of compressing the input relevant information from X (i.e., a side-channel trace) while discarding the irrelevant information translated as noise or irrelevant samples. In the context of side-channel attacks, we assume that the trained neural network can *generalize* if the attack methodology based

on Eq. (1) provides larger $P(g)$ value for the correct key candidate $g \in G = \{0, 1, ..., g_{max}\}$ and for a bounded (limited) number of side-channel traces N_T.

2.1.3 Attack Performance

A usual approach to assess the attacker's performance is to use metrics that denote the number of measurements required to obtain the secret key k^*. Common examples of such metrics are guessing entropy (GE) and success rate (SR) [SMY09]. Guessing entropy represents the average number of key candidates an adversary needs to test to reveal the secret key after conducting a side-channel analysis. More specifically, given N_T amount of traces in the attacking phase, an attack outputs a key guessing vector $g = [g1, g2, ..., g_{|K|}]$ in decreasing order of probability. The guessing entropy is the average position of k^* in g over several experiments. On the other hand, the success rate is defined as the average empirical probability that g_1 equals the secret key k^* (note, one can consider other values except g_1, and in that case, the ordering of that value denotes the order of the success rate).

2.1.4 On the Generalization Ability of Neural Networks

The generalization improvement of deep neural networks for side-channel attacks is considered with 1) data augmentation in [CDP17, PHJ+19], 2) noise addition as a regularization technique in [KPH+19], and 3) the use of ensembles in [Per19]. In this work, we aim to improve the generalization of a trained model to increase the attack performance by selecting the epoch when the training phase should stop. To that end, we consider the mutual information to define the epoch at which the neural network achieves its maximum generalization capacity during the training.

2.1.5 Datasets

In our experiments, we consider three publicly available datasets. All of them refer to the software AES implementations protected with the first-order Boolean masking. The first one is the widely used ASCAD database in the side-channel community for deep learning research [PSB+18]. The traces were measured from an implementation consisting of a software AES implementation running on an 8-bit microcontroller. The AES is protected with the first-order masking, where the two first key bytes (index 1 and 2) are not protected with masking (e.g., masks are set to zeros) and the key bytes 3 to 16 are masked. A trimmed trace set corresponding to the processing of key byte 3 (S_{box} operation in the first encryption round) is provided by the database. We consider the trace set containing 100 000 AES-128 encryption traces where the plaintext and key are randomly defined for each separate encryption. This trace set is used as a training and validation set. A second fixed-key trace set, consisting of 1 000 measurements is used as a test set. In this dataset, each trace contains 1 400 features.

The second trace set is the DPA Contest V4 (DPAv4) database [TEL14]. DPAv4 database provides trace sets collected from an AES-256 RSM (rotate shift masking) implementation. The training set consists of 34 000 traces with a fixed key. The test and validation sets contain 2 000 traces each. For convenience, we attack only the first key byte of an AES-256 implementation. Each trace consists of 2 000 features.

The third database refers to the CHES Capture-the-flag (CTF) AES-128 trace set, released in 2018 for the Conference on Cryptographic Hardware and Embedded Systems (CHES). This trace set also refers to AES-128 encryption, where 500 000 traces are available with random keys and random plaintext. An additional trace set of 5 000 fixed-key trace set was released to be considered as a test set. In our experiments, we consider 43 000 traces for the training set. The validation and test set consist of 2 500 traces each. The

keys for the training and validation set are different from the key configured for the test set. Each trace consists of 2 200 features.

2.2 Information Theory

In information theory, the entropy of a random variable X is defined as the average information obtained by observing X and it can be quantitatively defined as:

$$H(X) = -\sum_{x \in X} p(x) log_2 p(x), \tag{2}$$

where $p(x)$ represents the probability of variable X taking value x.

The *conditional entropy* of X given Y is defined as follows:

$$H(X|Y) = -\sum_{x \in X} p(x) \sum_{y \in Y} p(x|y) log_2 p(x|y). \tag{3}$$

This can be considered as the entropy of X when knowing Y. Finally, the *mutual information* defines the dependence between variable X and Y, and it can be defined using entropy and conditional entropy values as follows:

$$I(X;Y) = H(X) - H(X|Y). \tag{4}$$

Two properties of mutual information are important in this context. The first is the Data Processing Inequality (DPI), which states that for any three variables X, Y, Z, which form a Markov chain, $X \rightarrow Y \rightarrow Z$, the mutual information between the variables can only decrease and $I(X;Y) \geq I(X;Z)$. The second property is the invariance of mutual information to invertible transformations, which states that $I(X;Y) = I(f(X);g(Y))$ for any invertible functions f and g.

3 Related Works

In recent years, researchers proposed theoretical and empirical investigations about the deep neural network training assessment and its link to the profiled side-channel analysis. Next, we review the published works that address the problem of optimizing the training of a neural network based on consistent metrics. Additionally, we review related works that address information theory for deep learning in the context of side-channel analysis.

From the first paper considering convolutional neural networks for SCA [MPP16], deep learning gained a significant recognition in the SCA community as the paradigm to follow for profiled SCA. Still, despite very good results even when considering targets protected with countermeasures [CDP17, KPH+19], there are issues like interpretability and explainability of neural networks, or how to select the hyper-parameters (number/types of layers, number of neurons, activation functions, number of epochs, etc.) for the neural network.

The number of epochs for training is relevant for any application domain as one aims to have a model that generalizes as much as possible, but usually, machine learning metrics are a good indicator when to stop. Unfortunately, in SCA, such metrics are much less reliable [PHJ+19]. Currently, in profiled SCA, related works use early stopping where one stops training at the point when performance on a validation dataset starts to degrade [CDP17, PSK+18]. Alternatively, related works use a predetermined number of epochs for training [PSB+18, HGG19, KPH+19, ZBHV19]. In both cases, as the performance on the validation set is commonly observed through machine learning metrics, it is difficult to know if the training is stopped at the right moment.

To the best of our knowledge, the first paper that considers the explainability of neural networks in the context of SCA investigates the Singular Vector Canonical Correlation Analysis (SVCCA) tool to interpret what neural networks learn while training [vdVPB19]. There, the authors manage to show that even a small change in the dataset can radically change the internal representation (as observed through data from activation functions) of neural networks. An empirical analysis of several machine learning metrics for profiled side-channel attacks has been proposed in [PHJ+19]. The authors demonstrate the inconsistency of machine learning metrics such as accuracy, precision, F1 score, and recall in comparison to guessing entropy and success rate. In [Per19], the author indicates that validating a trained model based on the key ranking leads to higher success rates. The work proposed in [MDP20] states that maximizing the perceived information in a profiled side-channel attack is equivalent to minimizing the loss functions based on negative log-likelihood or cross-entropy. Finally, a connection between the mean square error loss function and the categorical cross-entropy is demonstrated in [vdVP19].

4 Information Theory of Deep Neural Networks

Shwartz-Ziv and Tishby [ST17] showed that information theory could be used to visualize the training phase of a deep network to compare the performance of different network architectures. When training a network, each layer is getting its information from the layer before and transforming it by using matrix multiplication of nonlinear functions.

Their insight was to treat each layer (the hidden activation functions) in the deep network as a random variable fully described by the information captured about the input data and the labels. Modeling each layer in the deep network as a random variable gives an alternative view of a deep network as a Markov chain. Each variable represents the nonlinear activation function, which successively transforms the input data into the label space. Using the mutual information between the layers, the input data, and the labels, we can visualize the transformation of the input data into the label space.

Definition 1. Information Path. Given en ensemble $(X, Y, (T_i)_n)$ where X represents the input data, Y represents a set of labels and T_i is a hidden layer in a n-layered network, described as $X, Y \rightarrow T_1 \rightarrow ... \rightarrow T_n$, the information path is defined as the set of points $\{[I(X;T_i), I(T_i;Y)]|i \in \{1,n\}\}$.

The *information path* is a record of the information each hidden layer preserves about the input data X and the output variables Y. It is typically computed for each epoch during the training phase. The information is plotted in a two-dimensional coordinate system referred to as the *information plane*. The coordinates of the information plane quantify the bits of information layer T_i has about the input data X, as $I(T_i;X)$ and the bits of information layer T_i has about the labels, as $I(T_i;Y)$.

We can view variable T_i as a compressed representation of the input X, and $I(T_i;X)$ calculated based on the value $p(x)p(t_i|x)$, which measures how compact the representation of X is. The maximum value for $I(T_i;X)$ is $H(X)$, which is the Shannon entropy that corresponds to the case where T_i copies X and there is no compression. The min value for $I(T_i;X)$ is 0 and corresponds to the case where T has one value.

Lemma 1. *Information Path Uniqueness. For each ensemble $(X, Y, (T_i)_n)$, where X represents the input data, Y represents a set of labels, and T_i is a layer in a n-layered network described as $X, Y \rightarrow T_1 \rightarrow ... \rightarrow T_n$ there exists a unique information path which satisfies the following two inequalities:*

$$H(X) \geq I(X;T_1) \geq ... \geq I(X;T_n) \geq I(X;\hat{Y}), \tag{5}$$

where $\hat{Y}$ are the labels predicted by the network, and

$$I(X;Y) \geq I(T_1;Y) \geq \ldots \geq I(T_n;Y) >\geq I(\hat{Y};Y). \tag{6}$$

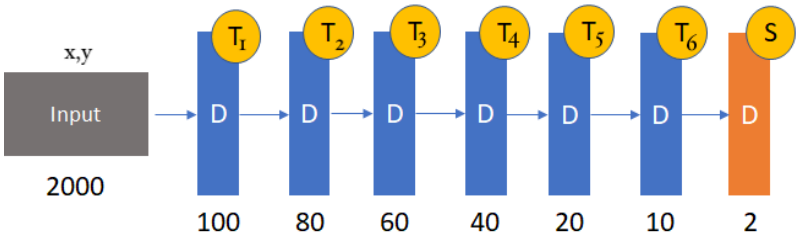


Figure 1: MLP with 6 hidden layers. The letter "D" denotes the dense (fully-connected) layer while the labels $T_1 - T_6$ correspond to the hidden layers 1 to 6 with the *tanh* activation function. "S" denotes the output layer with the *Softmax* activation function.

Proof. The proof for this lemma follows immediately by applying the DPI principle. □

4.1 The Information Bottleneck Principle

Shwartz-Ziv and Tishby [ST17] observe that stochastic gradient descent (SGD) optimization defines two distinct phases during training. The first phase refers to the *fitting* phase, where both $I(X;T_i)$ and $I(T_i;Y)$ increase fast as the training progresses. During the fitting phase, the deep network layers increase the amount of information about the input data and the labels. The second phase is the *compression* phase, where the network starts to compress or forget information about the input data and slowly increase its generalization capacity by retaining more information about the labels.

The behavior of the network during the compression phase has been linked to the form of the activation functions [CHO19]. This happens due to a random diffusion-like behavior of the SGD algorithm if double-sided saturating[2] nonlinear activation function such as *tanh* is employed.

More precisely, Shwartz-Ziv and Tishby provide results for *tanh* and show how the information about the labels increases in the compression phase [ST17]. On the other hand, for activation functions like $ReLU$, eLU, and $SeLU$), it seems that information about the labels continuously decrease while the deep neural network still shows compression and generalization. As demonstrated in [SBD+18], the non-saturating activation functions like $ReLU$ provide a different behavior in the compression phase and they show that generalization and compression have no causal connection. The authors of [CHO19] propose a solution based on *entropy-based binning* for the mutual information calculation and suggest that the mutual information between the output layer activations and the labels provide a better indication of generalization in the compression phase.

Figure 2 gives a snapshop of the information path of the deep network architecture described in Figure 1 at epochs: 1, 20, 100, and 200 during the training process. Each figure contains the coordinates of the information captured by 6 hidden layers $[I(X;T_i), I(T_i;Y)]$, where i represents the i-th hidden layer, and $[I(X;S), I(T_i;S)]$ represents the output layer. For this particular example, we see that information changes only for the last two hidden layers T_5 and T_6, and the output layer S. The plot contains mutual information results for twenty training experiments (consequently, each layer is represented by twenty dots). These results clearly demonstrate that at the beginning of the training phase, the mutual information quantities $[I(X;T_i), I(T_i;Y)]$ are at a minimum level. As the training progress (see the snapshots at epochs 20 and 100), the mutual information values increase until the $[I(X;T_i), I(T_i;Y)]$ reaches its maximum for all layers, including the output layer. If we continue the training process, the compression phase starts to happen as $I(X;T_i$ starts

[2] A saturating activation function squeezes the input data, i.e., the output is bounded to a certain range.

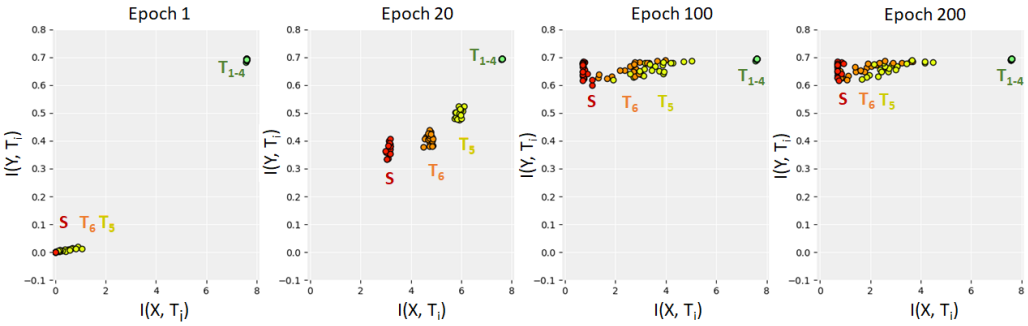


Figure 2: The information flow captured at epoch 1, 20, 100, and 200 for the network architecture depicted in Figure 1 when using the DPAv4 dataset (training set).

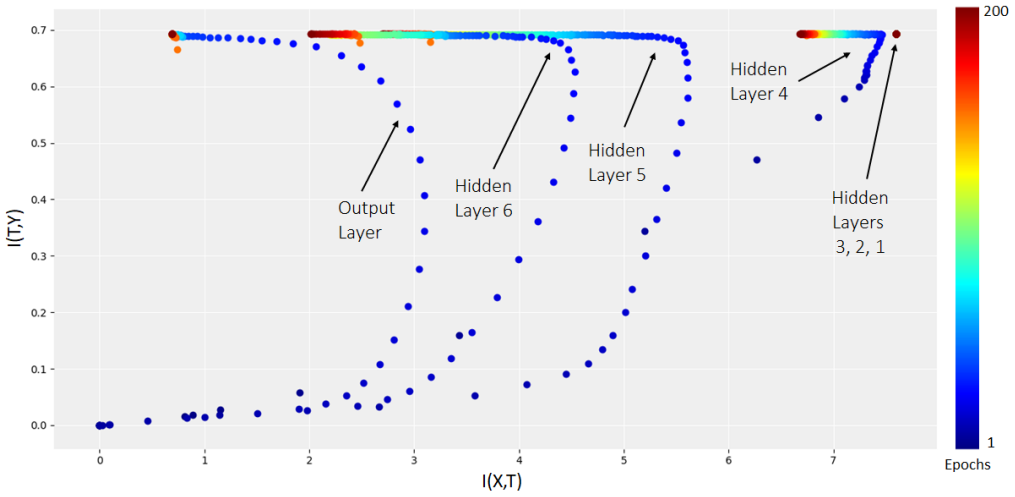


Figure 3: Information plane from DPAv4 dataset (training set).

to decrease and $I(T_i; Y)$ stays at a maximum level. In Figure 2, this information path is better observed for hidden layers T_5 and T_6 and the output layer S.

In Figure 3, we show the evolution of the information path for all training epochs for the DPAv4 dataset, for the same architecture illustrated in Figure 1. As we can observe, the training evolution provides two distinct phases, as proposed in the original publication [ST17]. In the first phase, the layers (mostly visible for hidden layers 4-6 and output layer) are fitting the training data and the information of an inner state T_i or layer increases for the input X and output Y[3].

4.2 The Information Path for Side-channel Analysis Data

In this section, we analyze whether information path (and consequently, information plane) can indicate the generalization properties of neural networks when conducting profiled side-channel analysis.

The generalization as information derived from the information path must be obtained from the validation set. The main reason for using the validation set instead of the training

[3]Note, information plane figures show different layers, but it is not possible to recognize a certain layer by just "observing" the graph, i.e., there is no pre-specified behavior for a certain layer. We store and plot data for each layer separately.

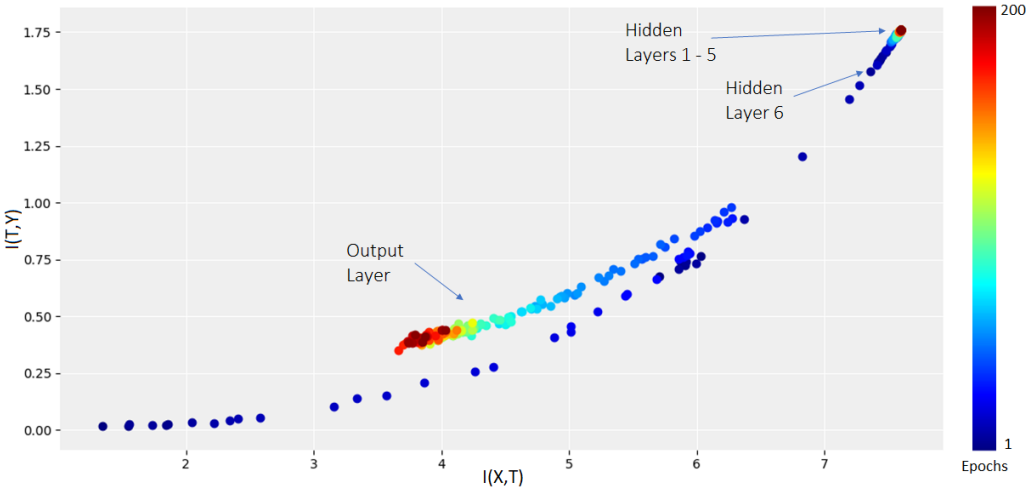


Figure 4: Information plane from DPAv4 dataset (validation set).

set is related to the generalization interval (in terms of epochs) that we must identify as the training progresses. Generalization interval is defined in Definition 2. Additionally, the metrics obtained from the training set are inconsistent for quantifying the generalization of a trained neural network, as already demonstrated in [Per19].

Definition 2. Generalization Interval. Given an ensemble $(X_{train}, Y_{train}, (T_i)_n)$ where X_{train} represents the input training data, Y_{train} represents a set of training labels, and T_i is a hidden layer in a n-layered network, the generalization interval defines the interval of training epochs at which we can obtain successful key byte recovery with Eq. (1) by classifying the dataset X_{test} with the trained neural network.

The results in Figure 4 show that it is possible to observe a different "movement" of the points $[I(X;T_i), I(T_i;Y)]$ in the information path when using the validation set. The fitting phase is clearly seen, as $I(X;T_i)$ and $I(T_i;Y)$ increase with the processing of first epochs (for the validation set, this movement is observable for output layer and hidden layer 6).

The compression phase is different from the information plane observed in Figure 3. For the validation set, the points $[I(X;T_i), I(T_i;Y]$ reach the maximum value for each hidden layer and, later, both quantities decrease with the processing of more epochs. This indicates the overfitting or over-training scenario for the given trained model. More specifically, this effect happens because the generalization in difficult side-channel analysis problems (i.e., masked or protected AES) is minimal if given in terms of deep learning metrics (accuracy, loss, recall). At the same time, we would like to capture the network model at an epoch when the best possible generalization occurs. From our observations, the epoch at which the generalization is optimal is given by the moment when $I(T_n;Y)$ reaches a maximum value, where T_n represents the activation from the output ($Softmax$) layer.

4.3 A Metric for Stopping the Training in SCA

A common technique to determine the optimal training point in neural networks is early stopping. This technique is based on a metric, usually training/validation loss or accuracy, that determines at which epoch the training must stop. The main idea is to avoid the continuation of the training process after the best possible value for a reference metric

is found. The assumption is that the neural network achieved the best generalization and will start overfitting and deteriorate the generalization after this point, which is an undesired behavior. For side-channel analysis, machine learning metrics have demonstrated to be inconsistent as reference for the validation process [PHJ+19]. As such, one can see that implementing early stopping based on, e.g., loss function or accuracy could lead to inconsistent results. An alternative would be to compute the key rank for the validation set at the end of each epoch. Unfortunately, calculating key rank will lead to a significant time overhead for larger datasets.

As stated in [CHO19], only the information transferred to the output network layer is important for measuring the generalization. At the same time, [ST17] mentioned that the beginning of the compression phase usually coincides with the best generalization. Typically, this is a moment when the value $I(T_n; Y)$ in the output layer achieves its maximum value. Additionally, [SBD+18] provides empirical results demonstrating that the compression phase does not necessarily improve generalization.

Consequently, we propose to use the mutual information $I(T_n; Y)$ in the output or *Softmax* layer as a reference metric for the early stopping. The maximum value for $I(T_n; Y)$ for the output layer happens when the fitting phase is usually finished and compression is started (In Figure 4, this moment is represented by the epoch when $I(T_n; Y)$ reaches a maximum value). This means that the training phase usually does not need to proceed to the compression phase in order to achieve the best generalization. The calculation of $I(T_n; Y)$ gives minimal overheads during the training process mainly because we make the computation for a small fraction of the validation set. Our estimation provided that the time overhead to compute $I(T_n; Y)$ at the end of each epoch is less than 2%. We demonstrate that, for several datasets, the success rate is commonly higher at the epoch that indicates the higher information value $I(T_n; Y)$ in the output layer.

The assumption is that, given a deep neural network defined with a set of hyper-parameters θ, the internal representations T_i, given by the output of a layer i, $l \in 1, n$, (where T_1 and T_n are the input and output layers, respectively) should inform about the labels Y and input X [AG18]. We assume that, during the training, the intermediate representation T_i will be compressed such that it contains the minimum amount of information about X in order to estimate Y correctly. Additionally, the intermediate representation T_i should be robust such that small addition of noise should not affect this compressed internal representation. This robustness can be achieved with regularization. Note, achieving the robustness for the input changes is not necessarily an easy task for machine learning as indicted in [PHAR18].

5 Experimental Validation

In this section, we empirically show that the mutual information between the output layer T_n and the desired output Y, $I(T_n; Y)$ provides a consistent metric to determine an optimal amount of epochs for the training phase. In all the experiments, the leakage model is the Hamming weight of one S_{box} output byte in the first AES encryption round. We use the Hamming weight leakage model based on the preliminary leakage assessment made on the considered datasets.

5.1 Number of Bins for $I(T_n; Y)$

One way to quantify the relationship between two variables X and Y through the mutual information is to convert these observations into histograms. There, we need to choose the number of bins for the histograms, which can be empirically determined. Thus, we start by estimating the most promising value for the number of bins for our datasets. This

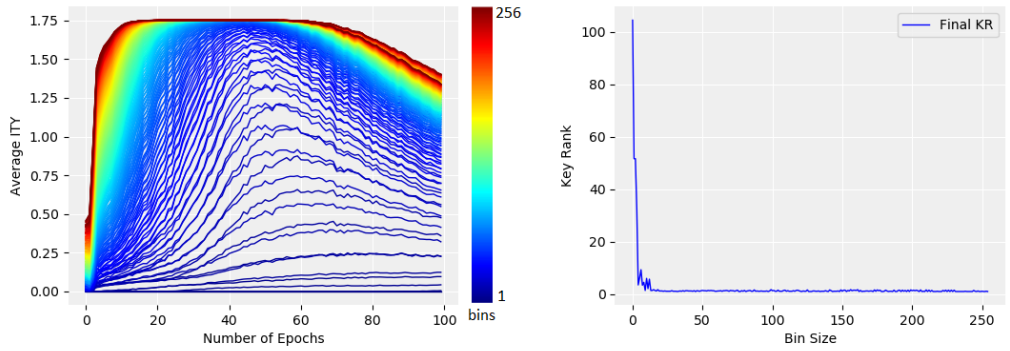


Figure 5: (to be viewed in colors) The influence of number of bins in the calculation of $I(T_n;Y)$ for the ASCAD dataset. Average $I(T_n;Y)$ w.r.t. number of epochs for different bin sizes (left). The final key ranking having the maximum value of $I(T_n;Y)$ as a reference metric for different bin sizes (1 to 256) (right).

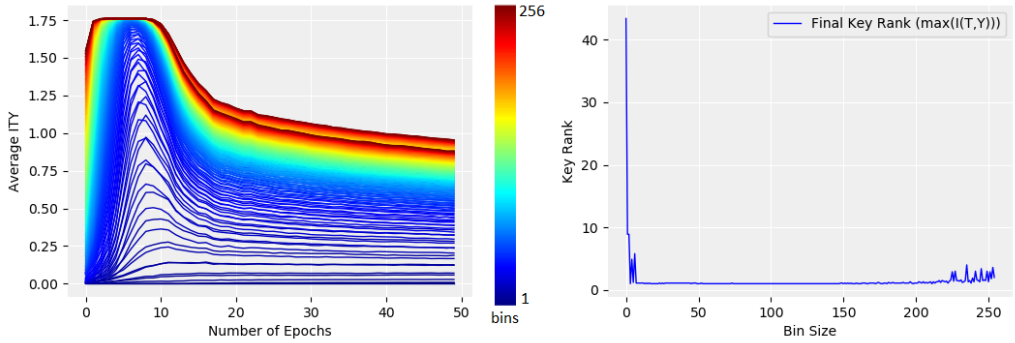


Figure 6: (to be viewed in colors) The influence of number of bins in the calculation of $I(T_n;Y)$ for the DPAv4 dataset. Average $I(T_n;Y)$ w.r.t. number of epochs for different bin sizes (left). The final key ranking having the maximum value of $I(T_n;Y)$ as a reference metric for different bin sizes (1 to 256) (right).

estimation is only computed for the quantity $I(T_n;Y)$, where T_n represents the activations from the output (*Softmax*) layer.

For the considered datasets, the target is always AES, which means that the output layer always contains 9 (the Hamming weight leakage model) units. As a consequence, the labels Y will have values inside these ranges. The histogram computed from T_i will range from 0 to 1 due to the *Softmax* activation function, which outputs the values between 0 and 1.

To verify the effect of the number of bins in the key ranking results, we test bin sizes from 1 to 256. The results are shown in Figures 5, 6, and 7 (the neural network architectures for each dataset are explained in Section 5.3). As we can observe in the figures on the right, any bin size larger than 15 leads to the final key rank lower than 4, in which key rank equal to 1 indicates the successful key recovery. The key rank is always computed for a separate test set and is obtained by selecting the model at the epoch giving the highest $I(T_n;Y)$ for each tested bin size. The plots on the left side of Figures 5, 6, and 7 show the value of $I(T_n;Y)$ w.r.t. the number of epochs for all tested bin sizes. As we can see, if the bin size is too small, the mutual information $I(T_n;Y)$ barely changes.

From these calculations, we conclude that the bin size for the mutual information

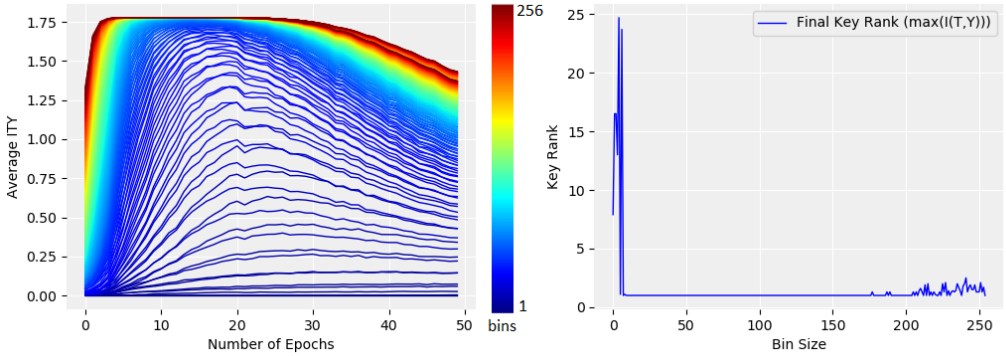


Figure 7: (to be viewed in colors) The influence of number of bins in the calculation of $I(T_i, Y)$ for the CHES CTF 2018 dataset. Average $I(T_i, Y)$ w.r.t. number of epochs for different bin sizes (left). The final key ranking having the maximum value of $I(T_i, Y)$ as a reference metric for different bin sizes (1 to 256) (right).

estimation directly impacts the side-channel analysis results in terms of the test key rank. In our experiments, a bin size of 100 leads to the best results for the Hamming weight leakage model (9 classes). Still, different datasets (or even other leakage models) might require another number of bins in order to provide optimal results.

5.2 Mutual Information Bounds for $I(T_n; Y)$

The analysis of the mutual information in the output layer provides an indication of the neural network's generalization ability. As specified in Section 4.3, the maximum value for $I(T_n; Y)$, when T_n and Y are distributions obtained from the validation set, indicates a training epoch at which the neural network provides its best generalization capacity. Next, we are interested in empirically demonstrating the minimum and maximum bounds for the $I(T_n; Y)$ in the output layer. In case a neural network can provide the maximum classification accuracy (100%), the mutual information $I(T_n; Y)$ stays at a maximum value, which we define as being the upper bound. If we test the network against random labels, the classification accuracy is around random guessing and the quantity $I(T_n, Y)$ stays at the minimum value, which we define as being the lower bound.

Definition 3. $I(T_n; Y)$ upper bound. Given an ensemble $(X, Y, (T_i)_n)$ where X represents the input data, Y represents a set of labels, and T_i is a hidden layer in a n-layered network, the upper bound for $I(T_n, Y)$ is reached when the classification accuracy for the dataset X with labels Y achieves 100%, where T_n represents the *Softmax* output probabilities for the output layer.

Definition 4. $I(T_n, Y)$ lower bound. Given an ensemble $(X, Y, (T_n)_k)$ where X represents the input data, Y represents a set of labels, and T_i is a hidden layer in a n-layered network, the lower bound for $I(T_n; Y)$ is given by a dataset X with labels Y defined at random, where T_n represents the *Softmax* output probabilities for the output layer.

Remark 1. Note, for the Hamming weight leakage model of a byte (e.g., S_{box} output byte in the first AES encryption round), the lower bound can be seen as classifying all traces as the Hamming weight 4. For other leakage models (e.g., bit or identity), the lower bound of $I(T_n; Y)$ will be defined by situations when validation data provides accuracy similar to random guessing.

In Figure 8, we show the lower and upper bounds. The tests were conducted with an unprotected software AES implementation. Note that the network achieves the upper

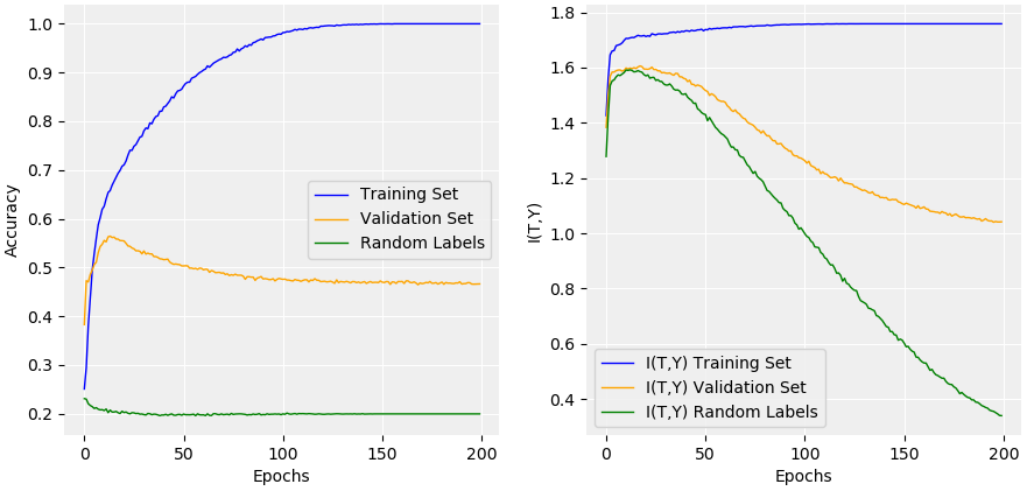


Figure 8: (to be viewed in colors) Training, validation, and random labels accuracy (left). Training, validation, and random labels $I(T_n;Y)$ (right).

bound for $I(T_n;Y)$ when the classification accuracy reaches 100%, which is the case when T_n and Y are obtained from the training set and the trained network is overfitting on this training set. Figure 8 also provides the mutual information value $I(T_n;Y)$ computed from the validation set. The validation accuracy is approximately 46% after the processing of 200 epochs. For side-channel analysis, with this test or validation accuracy, it is possible to obtain successful key recovery with Eq. (1) by processing a few dozens of traces for any considered side-channel trace set.

5.2.1 $I(T_n;Y)$ for the Regularized Neural Networks

Regularization techniques can help prevent a deep neural network from overfitting during the training process. To check the impact of the regularization on the neural network, we use the information plane as it provides a visual indication for the relationship of $I(X;T_n)$ and $I(T_n;Y)$.

The maximum value of $I(T_n;Y)$ during training indicates an epoch at which the neural network should be inside the generalization interval with respect to the training process, as defined in Definition 2. When the network does not implement any regularization technique in its hyper-parameters configuration, the trained model has a higher chance of overfitting the training data. As a consequence, the mutual information $I(T_n;Y)$ reaches a maximum value (where the distributions T_n and Y are obtained from the validation set) and after that, $I(T_n;Y)$ for validation decreases continuously while $I(T_n,Y)$ for the training stays at a maximum value (see Figure 8). On the other hand, a regularized network prevents overfitting and, consequently, the $I(T_n;Y)$ obtained for the validation set takes more time (i.e., epochs) to reach its maximum value. Thus, the generalization interval, given in terms of epochs lasts longer in the training process.

Figure 9 shows the accuracy and $I(T_n;Y)$ for training, validation, and random labels sets obtained from a regularized convolutional neural network with dropout. As we can see, after processing 100 epochs, the training accuracy is still under 100%, which means that the network still did not fit the training data perfectly, and that is the desired outcome for a regularized neural network. The validation accuracy reaches approximately 56%, which is a significantly higher value compared to 46% without regularization. The mutual information $I(T_n;Y)$ for the validation set (see Figure 9 on the right) reaches its maximum value and stays longer at this level. For this particular case, this indicates that the same

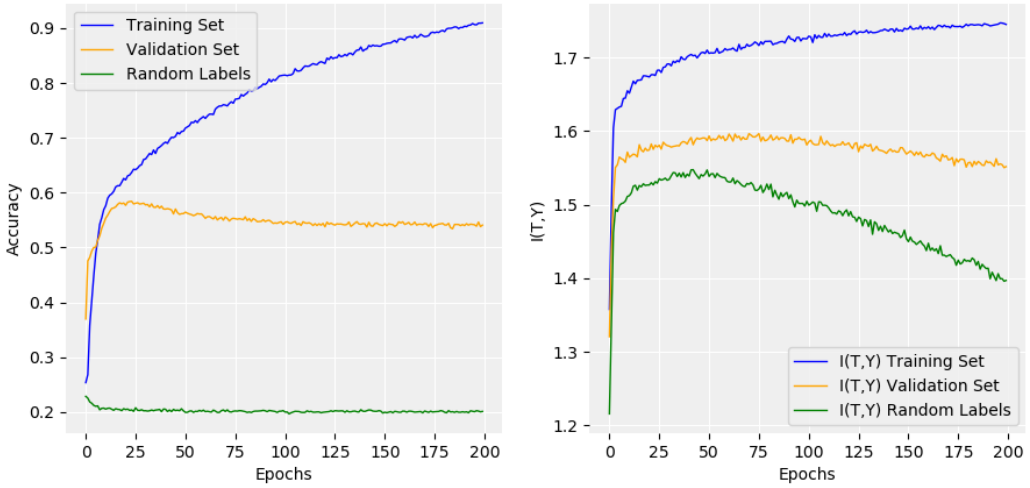
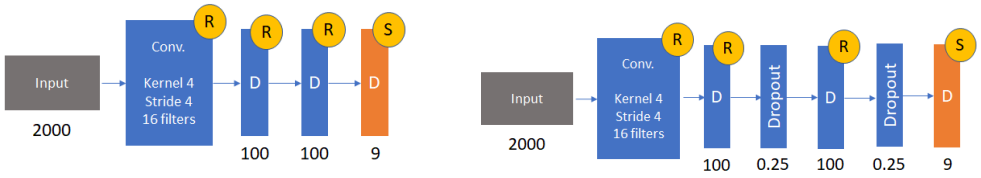


Figure 9: (to be viewed in colors) Results from a regularized neural network with dropout. Training, validation, and random labels accuracy (left). Training, validation, and random labels $I(T_n; Y)$ (right).



(a) CNN

(b) CNN with dropout

Figure 10: Convolutional neural network configurations (learning rate = 0.001, *Adam* optimizer, batch size = 400, randomly uniform initialized weights).

generalization level is kept until at least epoch 100. Consequently, the value of $I(T_n; Y)$ suggests why regularized networks provide better generalization and are more robust against overfitting. As the value of $I(T_n; Y)$ stays high for more training epochs, the neural network provides better generalization for those epochs.

At the same time, accuracy is not able to indicate the same phenomenon, as its value remains stable (albeit of different magnitude for validation set) for both regularized and non-regularized networks. The neural network configurations (with and without dropout) are illustrated in Figure 10. The "R" and "S" labels refer to *RelU* and *Softmax*, respectively. The number under the layer block indicates the number of neurons in dense layers ("D") and the dropout rate for dropout layers.

5.3 Results for Publicly Available Datasets

Next, we compare the profiled attack performance for the AES encryption based on different metrics. As mentioned in Section 4.3, a particular metric is used to select the machine learning model at epoch t that achieves the best performance. The selected model is subsequently used in the validation and test phases. We compare the following metrics: validation loss, validation accuracy, validation recall, key rank for the validation set, and $I(T_n; Y)$ with a histogram bin size of 100.

Note, for the three tested datasets (ASCAD, DPAv4, and CHES CTF), the results

are obtained by attacking one key byte in the first AES encryption round. The selected leakage model is the Hamming weight of an S_{box} output. We conduct a tuning phase for hyper-parameters, where we experiment with varying CNN and MLP architectures. Finally, we verified that the selected MLP provides satisfactory results for the ASCAD database and the selected CNNs provide better results for the DPAv4 and CHES CTF databases. We emphasize that we do not claim these architectures to be optimal, as finding optimal architectures was not the goal of this work. The final neural network configurations use the *adam* optimizer for the backpropagation algorithm and the learning rate is always set to 0.001. Initial weights are initialized at random and the training regime is based on mini-batches of 400 traces. The selected loss function is the categorical cross-entropy provided by *keras* library. All the experiments were performed on a computer equipped with a GPU Nvidia RTX 2060.

5.3.1 AES-128 Encryption - ASCAD (Random Keys)

The empirical validation on the ASCAD database (key byte 3) considers 99 000 traces for training, 500 traces for validation, and 500 traces for the test. Both validation and test sets have a fixed key. The neural network is trained for 100 epochs and validation accuracy, recall, and loss are used to determine the epoch with the best corresponding metric. Additionally, the validation set is used to determine the epoch with the best key ranking (based on the number of traces to achieve key rank 1) and the maximum $I(T_n; Y)$. After identifying the best epoch for each metric, the corresponding machine learning models are applied to the test set.

Figure 11 shows the guessing entropy and success rate for the test set obtained for each validation metric. These results were obtained from averaging of 100 trained models with the same hyper-parameters. The configured neural network is a multiple layer perceptron with four dense layers containing 400 neurons each. As results in Figure 11 indicate, the best success rate is achieved when the machine learning model is selected from the epoch when the metric is the maximum value of $I(T_n; Y)$. More precisely, around the processing of 400 traces, the success rate reaches 100% if the model is selected from the epoch determined by the maximum $I(T_n, Y)$ value.

Figure 12 shows the results for 100 experiments of the same neural network. The plot on the upper left side of Figure 12 shows the $I(T_n; Y)$ evolution for the processed epochs. On average, the maximum $I(T_n, Y)$ value is achieved between epochs 32 and 38, as indicated by the plot distribution on the upper right side of Figure 12. The same figure (right bottom) shows the distribution of the best epoch based on the validation key rank metric. This histogram contains the results of 100 experiments (unchanged hyper-parameters) and indicates that the best validation key rank may happen at different epochs. Finally, the plot on the bottom left shows the averaged test and validation key ranks for the number of epochs (training phase) for the 100 experiments. It is clear that for the ASCAD dataset and the considered MLP architecture, there is an interval (epochs between 15 and 40) in which the key rank is low and, consequently, generalization is satisfactory. Therefore, we need a consistent validation metric to detect this interval.

5.3.2 AES-256 Encryption - DPAv4

For the DPAv4 dataset, we consider 36 000 traces in the training set and 2 000 traces in the validation set. An additional 2 000 traces are used as a test set. Figure 13 shows the guessing entropy and success rate obtained from the selected metrics (accuracy, recall, loss, key rank, and maximum $I(T_n; Y)$) from the validation set. Again, selecting the model at an epoch with the maximum $I(T_n, Y)$ for the validation set provides success rates that are among the best success rates scores. The same stands for the guessing entropy. These results were obtained from the 100 training runs on a neural network configured with

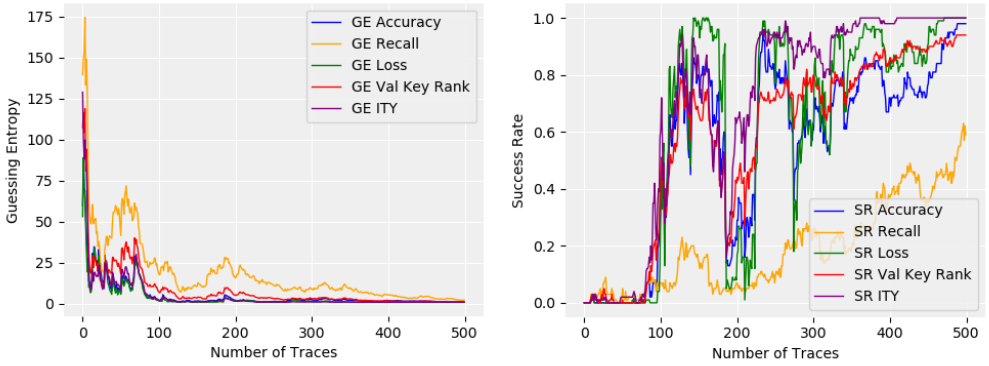


Figure 11: (to be viewed in colors) Guessing entropy (left) and success rate (right) of key byte 0 based on different early stopping metrics. MLP trained from 99 000 traces from AES-128 - ASCAD database.

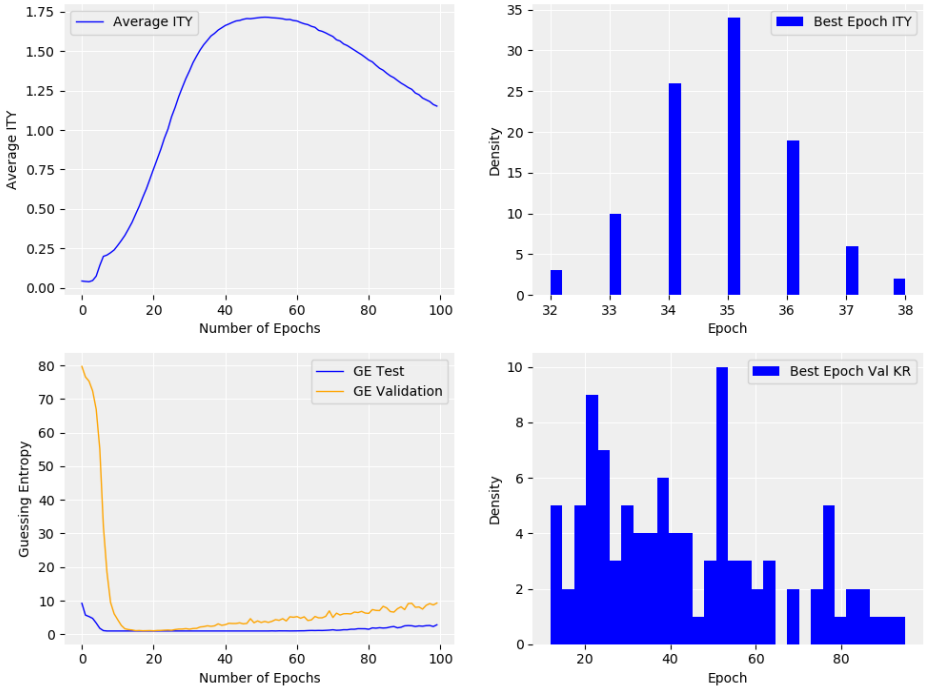


Figure 12: (to be viewed in colors) Results on ASCAD dataset trained with a MLP. Evolution of $I(T_n;Y)$ and Guessing Entropy w.r.t. number of training epochs (left). Distribution of best epoch w.r.t. maximum $I(T_n;Y)$ and validation key rank metrics (right).

unchanged hyper-parameters. We use a convolutional neural network with one convolution layer (16 filters, kernel size of 10) and two dense layers with 400 neurons each. The activation function is $ReLU$ for all layers (except the output layer where it is $Softmax$).

As we can see from Figure 14, the network achieves its maximum $I(T_n;Y)$ value, between epochs 8 and 10. The lower right plot in Figure 14 indicates the epochs at which the network provides satisfactory generalization (as observed through guessing entropy). As we can see, the best key rank happens between epochs 5 and 25. The maximum $I(T_n;Y)$

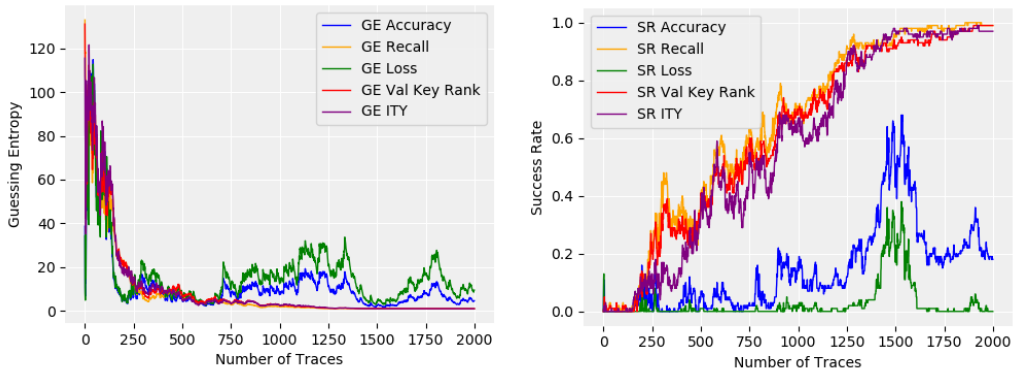


Figure 13: (to be viewed in colors) Guessing entropy (left) and success rate (right) of key byte 0 based on different early stopping metrics. CNN trained with 36 000 traces from AES-256 - DPAv4 database.

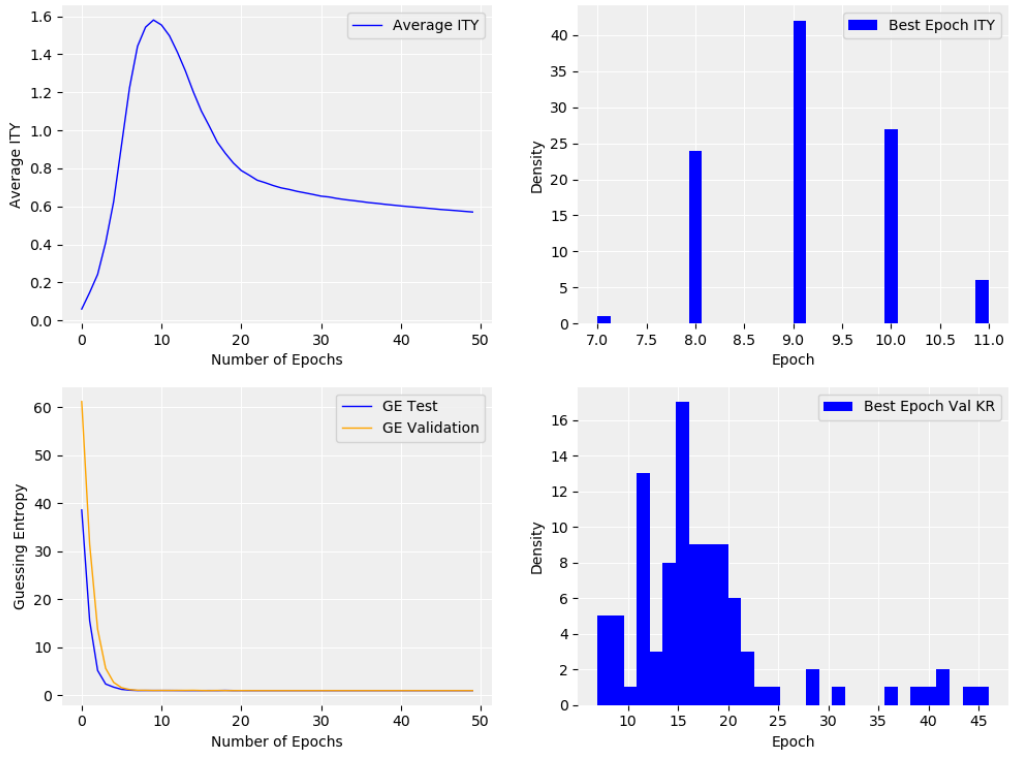


Figure 14: (to be viewed in colors) Results on DPAv4 dataset for a CNN. Evolution of $I(T_n, Y)$ and Guessing Entropy w.r.t. number of training epochs (left). Distribution of best epoch w.r.t. maximum $I(T_n, Y)$ and validation key rank metrics (right).

value is located inside this range, which means that the maximum $I(T_n; Y)$ is an adequate reference metric to define the optimal number of training epochs.

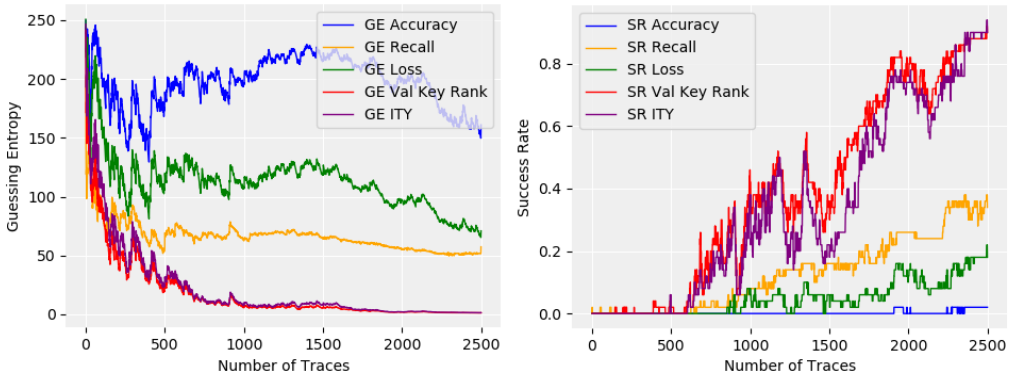


Figure 15: (to be viewed in colors) Guessing entropy (left) and success rate (right) of key byte 0 based on different early stopping metrics. CNN trained with 45 000 (random key) traces from AES-128 - CHES CTF 2018 database.

5.3.3 AES-128 Encryption - CHES CTF 2018

Figure 15 shows the guessing entropy and success rate for five considered validation metrics on the CHES CTF dataset. The configured neural network is a convolutional neural network with two convolution layers (16 and 32 filters, respectively, and a kernel size of 4 for both convolution layers) and two dense layers with 100 neurons each. The activation function is *ReLU* for all layers (except the output layer where it is *Softmax*). We can observe that using the training model at the epoch with the best validation key rank and maximum $I(T_n;Y)$ provides higher success rates. At the same time, retrieving the model at epochs indicated by the best validation accuracy, loss, or recall leads to poor success rate and guessing entropy results.

Figure 16 provides the mutual information value $I(T_n;Y)$ for training phase and every epoch (upper left). Additionally, this figure gives the guessing entropy value (lower left) at each training epoch for both validation and test sets. As the figure indicates, there is a range of epochs where the trained neural network provides sufficient generalization to recover the key (i.e., guessing entropy is 1). In this particular example, after epoch 30, the network starts to degrade its generalization capacity as it starts to overfit on the training set. On the other hand, the generalization capacity before epoch eight also provides, on average, poor generalization since the network is inside the fitting phase, where satisfactory generalization is not achieved yet (i.e., the network underfits). As a consequence, a metric that will select the trained machine learning model between (approximately) epochs 8 and 30 will result in a successful key recovery. In our case, the $I(T_n;Y)$ metric indicates that the network achieves its best generalization capacity between epochs 14 and 26, matching the interval when the guessing entropy is, on average, close to 1. The continuous reduction in the average of $I(T_n;Y)$ indicates that processing more epochs leads to overfitting the network with respect to the training data.

When attacking a protected target, like the public databases consisting of the first-order masked AES implementations, model generalization is very limited and validation or test metrics are close to random guessing. For side-channel analysis, a "good enough" generalization is given by a low guessing entropy or high success rate. As we can observe from the results given in Section 5, the trained model at each epoch provides different key rank results and the over-training easily leads to deterioration of the model's generalization. This problem can be addressed by using an appropriate metric to save the trained model at the epoch that provides the best success rate or guessing entropy. Our experimental analysis shows that having the maximum value of $I(T_n;Y)$ as a metric to select the model

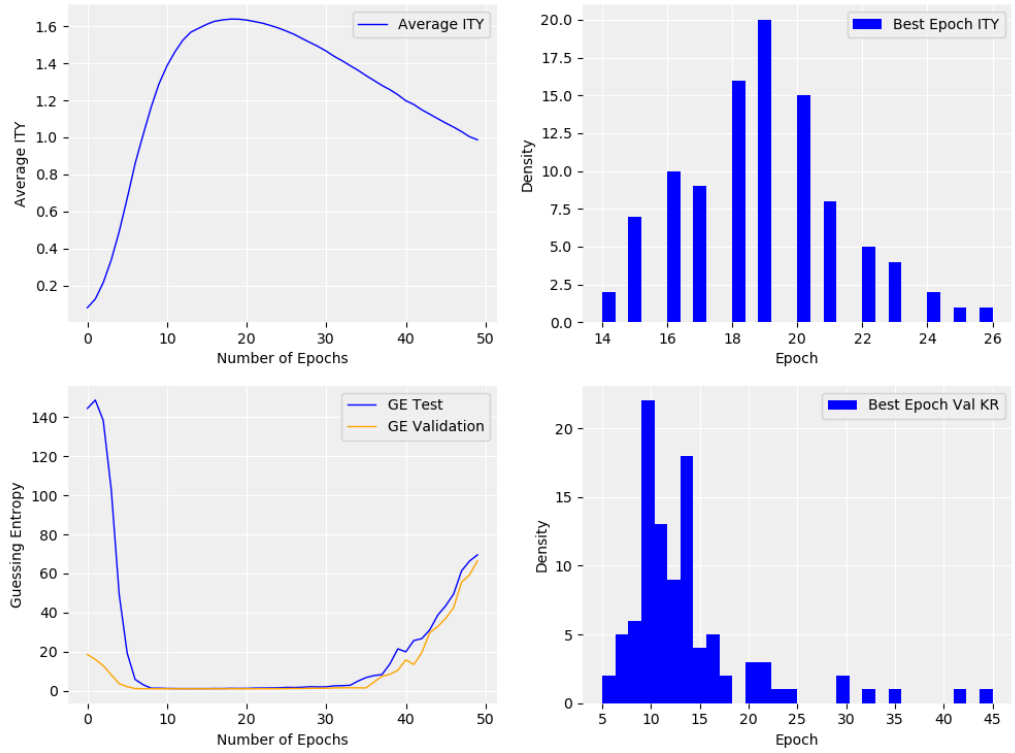


Figure 16: (to be viewed in colors) Results on CHES CTF dataset for a CNN. Evolution of $I(T_n;Y)$ and Guessing Entropy w.r.t. number of training epochs (left). Distribution of the best epoch w.r.t. maximum $I(T_n;Y)$ and validation key rank metrics (right).

at the best epoch provides better success rate and guessing entropy results when compared to machine learning metrics like loss, recall, or accuracy.

6 Conclusions and Future Work

The selection of correct performance metrics leads to a better interpretation of deep neural networks. In particular, the usage of deep learning for profiled side-channel attacks on the masked AES implementations is very sensitive to the number of processed epochs (i.e., how long is the training). In this paper, we demonstrate that using the mutual information between output layer activations (i.e., the output *Softmax* probabilities) and the true labels ($I(T_n;Y)$) of a validation set leads to better generalization for separate test sets. Too long training of a deep neural network can negatively affect its generalization due to overfitting. We compared $I(T_n;Y)$ metric against conventional machine learning metrics (accuracy, recall, and loss) and we verified that mutual information is a more reliable metric to detect an epoch at which the trained neural network is inside a reliable generalization interval. As such, we can conclude that mutual information should be considered as a metric of choice when conducting a deep learning-based side-channel analysis.

For future works, we aim to investigate the usage of mutual information metric as a reference for the selection of other hyper-parameters. Additionally, we would like to investigate the behavior of this metric in cases when the traces contain misalignment and consequently, the generalization is even more difficult. Such analysis is also important to improve the portability capabilities of trained deep neural networks for side-channel attacks.

References

[AG18] Rana Ali Amjad and Bernhard C. Geiger. How (not) to train your neural network using the information bottleneck principle. *CoRR*, abs/1802.09766, 2018.

[Ano20] Anonymous for TCHES submission. Source code for mutual information appraoch for stopping the training), 2020. https://gitlab.com/ches_submission_dl_mia/ches_submission_dl_mia.

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

[CDP17] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures - profiling attacks without pre-processing. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 45–68. Springer, 2017.

[CHO19] Ivan Chelombiev, Conor Houghton, and Cian O'Donnell. Adaptive estimators show information compression in deep neural networks. In *International Conference on Learning Representations*, 2019.

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

[FR14] Aurélien Francillon and Pankaj Rohatgi, editors. *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*, volume 8419 of *Lecture Notes in Computer Science*. Springer, 2014.

[HGG19] Benjamin Hettwer, Stefan Gehrer, and Tim Güneysu. Profiled power analysis attacks using convolutional neural networks with domain knowledge. In Carlos Cid and Michael J. Jacobson Jr., editors, *Selected Areas in Cryptography – SAC 2018*, pages 479–498, Cham, 2019. Springer International Publishing.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[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. Cryptogr. Hardw. Embed. Syst.*, 2019(3):148–179, 2019.

[LMBM13] Liran Lerman, Stephane Fernandes Medeiros, Gianluca Bontempi, and Olivier Markowitch. A machine learning approach against a masked AES. In Francillon and Rohatgi [FR14], pages 61–75.

[MDP20] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. A comprehensive study of deep learning for side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1):348–375, 2020.

[MHM13] Zdenek Martinasek, Jan Hajny, and Lukas Malina. Optimization of power analysis using neural network. In Francillon and Rohatgi [FR14], pages 94–107.

[MPP16] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. In Claude Carlet, M. Anwar Hasan, and Vishal Saraswat, editors, *Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings*, volume 10076 of *Lecture Notes in Computer Science*, pages 3–26. Springer, 2016.

[MZVT15] Zdenek Martinasek, Ondrej Zapletal, Kamil Vrba, and Krisztina Trasy. Power analysis attack based on the MLP in DPA contest v4. In *38th International Conference on Telecommunications and Signal Processing, TSP 2015, Prague, Czech Republic, July 9-11, 2015*, pages 154–158. IEEE, 2015.

[Per19] Guilherme Perin. Deep learning model generalization in side-channel analysis. *IACR Cryptology ePrint Archive*, 2019:978, 2019.

[PHAR18] Stjepan Picek, Annelie Heuser, Cesare Alippi, and Francesco Regazzoni. When theory meets practice: A framework for robust profiled side-channel analysis. Cryptology ePrint Archive, Report 2018/1123, 2018. https://eprint.iacr.org/2018/1123.

[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. Cryptogr. Hardw. Embed. Syst.*, 2019(1):209–237, 2019.

[PSB+18] Emmanuel Prouff, Remi Strullu, Ryad Benadjila, Eleonora Cagli, and Cécile Dumas. Study of deep learning techniques for side-channel analysis and introduction to ASCAD database. *IACR Cryptology ePrint Archive*, 2018:53, 2018.

[PSK+18] Stjepan Picek, Ioannis Petros Samiotis, Jaehun Kim, Annelie Heuser, Shivam Bhasin, and Axel Legay. On the performance of convolutional neural networks for side-channel analysis. In Anupam Chattopadhyay, Chester Rebeiro, and Yuval Yarom, editors, *Security, Privacy, and Applied Cryptography Engineering*, pages 157–176, Cham, 2018. Springer International Publishing.

[SBD+18] Andrew M. Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan D. Tracey, and David Daniel Cox. On the information bottleneck theory of deep learning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. A stochastic model for differential side channel cryptanalysis. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005,*

Proceedings, volume 3659 of *Lecture Notes in Computer Science*, pages 30–46. Springer, 2005.

[SMY09] François-Xavier Standaert, Tal G Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 443–461. Springer, 2009.

[ST17] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *CoRR*, abs/1703.00810, 2017.

[TEL14] TELECOM ParisTech SEN research group. DPA Contest (4^{th} edition), 2013–2014. http://www.DPAcontest.org/v4/.

[TZ15] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle, 2015.

[vdVP19] Daan van der Valk and Stjepan Picek. Bias-variance decomposition in machine learning-based side-channel analysis. Cryptology ePrint Archive, Report 2019/570, 2019. https://eprint.iacr.org/2019/570.

[vdVPB19] Daan van der Valk, Stjepan Picek, and Shivam Bhasin. Kilroy was here: The first step towards explainability of neural networks in profiled side-channel analysis. Cryptology ePrint Archive, Report 2019/1477, 2019. https://eprint.iacr.org/2019/1477.

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