Deep Stacking Ensemble Learning Applied to Profiling Side-Channel Attacks

Dorian Llavata, Eleonora Cagli, Rémi Eyraud, Vincent Grosso and Lilian Bossuet

October 9, 2023

Deep Stacking Ensemble Learning applied to Profiling Side-Channel Attacks

Dorian Llavata[1,2], Eleonora Cagli[1], Rémi Eyraud[2], Vincent Grosso[2]
, and Lilian Bossuet[2]

[1] Univ. Grenoble Alpes, F-38000, Grenoble, France, CEA, LETI, MINATEC
Campus, F-38054 Grenoble, France.
{dorian.llavata,eleonora.cagli}@cea.fr

[2] Univ. Jean Monnet Saint-Etienne, CNRS, Institut d Optique Graduate School,
Lab. Hubert Curien UMR 5516, F-42023, SAINT-ETIENNE, France.
{remi.eyraud,vincent.grosso,lilian.bossuet}@univ-st-etienne.fr

Abstract. Deep Learning is nowadays widely used by security evaluators to conduct side-channel attacks, especially in profiling attacks that allow a supervised learning phase. However, designing an efficient neural network model in a side-channel attack context can be a difficult task that may require a laborious hyperparameterization process. Hyperparameter selection is known to be a challenging problem in Deep Learning, while being a crucial factor for neural networks performances. Recent works investigate the so-called Deep Ensemble Learning in the side-channel context. It consists in using multiple neural networks in a single predictive task and aggregating the several predictions in an opportune way. The intuition behind is to use the power of numbers to improve the attack performance. In this work, we propose to use Stacking as an aggregation method, in which a meta-model is trained to learn the best way to combine the output class probabilities of the ensemble networks. Our proposal is supported by several experimental results, that allow to conclude that the use of Stacking can relieve the security evaluator from performing a fine hyperparameterization.

Keywords: Side-Channel Attacks · AES · Neural Networks · Ensemble Learning · Stacking.

1 Introduction

Embedded cryptography on constrained electronic devices like smart cards can be vulnerable to Side-Channel Attacks (SCA). These attacks exploit physical leaks collected on a device during the execution of cryptographic operations, such as energy consumption [16] or electromagnetic emission [10]. The analysis of these physical leaks can allow an attacker or an evaluator to retrieve sensitive data and compromise the devices security. Depending on the level of access and control of the target device, SCA can be categorized as profiling (*e.g.* template attacks [7]), or non-profiling attacks (*e.g.* DPA [16]/CPA [5]). The profiling scenario works on the principle that the evaluator has full control over a clone device identical to the target device. In this configuration, the evaluator splits the process into two phases. First, a profiling or characterization phase in which he uses the clone device to determine when the sensitive variable is leaking and to design an accurate model of the physical leakage of the clone device. Second, an attack phase in which the characterized leakage is used to attack the real target device. Since a profiling SCA may be viewed as a classical supervised learning problem, various machine learning methods have been investigated, such as Support Vector Machines [14] and Random Forests [17]. In recent years, profiling SCA based on deep learning have proved to be very efficient [3, 6, 18]. However, deep learning algorithms have much more tunable hyperparameters than other techniques. Their correct configuration is essential to obtain a good attack performance and it is very difficult to precisely know which hyperparameters influence the attack performance. In particular, too complex neural network architectures may be prone to the overfitting phenomenon, that is, when a model learns the training data by heart and is no more able to generalize on unseen data.

1.1 Related works

A few papers discuss about methodologies to build neural networks for SCA. Zaid *et al.* [27] proposed a methodology to generate robust convolutional neural network architectures. The authors used visualization tools to try to understand the impact of each convolutional hyperparameter. Robissout *et al.* [23] explore several regularization techniques (Batch Normalization, Weight decay and Dropout) to improve the attack performance of a neural network. Some work has investigated methods to automate the search of hyperparameters, Wu *et al.* [26] proposed to use Bayesian optimization to find optimal hyperparameters for neural network architectures. In the same way, Rijsdijk *et al.* [22] propose the use of Reinforcement Learning techniques. More recently, the SCA community has begun to experiment with neuroevolution and genetic algorithms for neural network design [1]. On the other hand, to improve the generalization and to limit the efforts of hyperparameterization, several works propose to use Ensemble Learning. Destouet *et al.* [8] use an ensemble of models for approximate a leakage model by targeting different sensitive values. Gao *et al.* [11] explore different ensemble methods based on decision tree to improve the attack success, including RusBoost, Bagging, and Adaboost methods. Recent work has begun to investigate the use of ensemble learning with deep neural network as weak model in SCA context. Perin *et al.* [20] provided experimental results on implementations of symmetric algorithms which show that combining predictions from multiple neural networks of different architectures with a bagging method allows to gain in attack performance. An extension of this work has been proposed by Zaid *et al.* [28], on asymmetric algorithms, with the proposal of a new loss function named *Ensembling loss* which aims to maximizing diversity between weak models during the training process.

1.2 Contributions and Paper organization

Contributions. This paper extends the preliminary results of deep ensemble learning applied in profiling SCA context by proposing to use stacking as an aggregation method [25]. We report an experimental exploration of the stacking method, whose goals are to assess the soundness of such a technique in profiling SCA context to highlight its adventages and inconvenients and to compare it with the bagging. The performance optimization is out of the scope of the experimental campaign.

Paper organization. The paper is organized as follows. Section 2 provides background about profiling SCA and ensemble learning. Section 3 presents our experimental results, together with a discussion about how stacking can significantly improve generalization and attack performance by comparing it with bagging. In Section 4 we discuss about results, highlight the cost and the effectiveness of such an approach to reduce the hyperparameterization effort and try to deduce a way to construct suitable architectures for a meta-model. Finally, in Section 5 we conclude and discuss possible future research directions.

2 Introduction to Profiling Side-Channel Attacks and Ensemble Learning

2.1 Notations

Let capital letters X denote random variables (random vectors if in bold $\boldsymbol{X}$), and the corresponding lowercase letters x denote their realizations. (resp. $\boldsymbol{x}$ for vectors). During their acquisition, each trace is associated with a target sensitive variable $Z = f(K, P)$, where P denotes some public variable, e.g. a plaintext, and K the part of secret key the evaluator aims to retrieve.

2.2 Profiling Side-Channel Attacks and Evaluation metrics

Profiling SCA. Side-channel attacks are typically performed using a divide-and-conquer strategy to independently attack chunk of the secret key called *subkeys*. For example, in the case of AES-128, instead of directly attacking the entire 16-byte key (which is computationally infeasible), the attack is divided into 16 parts and attempts to recover each 1-byte subkey separately. In the profiling phase, the evaluator aims to characterize the leakage from the clone device. The issue of such a characterization may be, for example in the case of classification-task-inspired deep learning attacks, a model $F(z, t)$ that provides an estimate of the posterior probability $Pr[Z = z|\boldsymbol{T} = \boldsymbol{t}]$, being Z the target sensitive variable and $\boldsymbol{T}$ the random vector representing side-channel traces. In this work, the model F is assumed to be either a Multilayer Perceptron (MLP) or a Convolutional Neural Network (CNN). Once the profiling phase is done, and the model is able to establish the relationship between the leakage and the corresponding value of the sensitive variable (which is linked to the target subkey), the evaluator applies the model on additional traces from the real target device. Finally to retrieve the subkey value, the evaluator will need to match the predicted sensitive values to an estimation of the subkey value. To do this, all possible values of a subkey are enumerated, and for each of them, all resulting sensitive variables are computed. Then, for all the subkey candidates, the evaluator exploits the set of traces by summing the logarithms of the output probabilities of their respective labels. This gives the following logarithmic probability vector $\boldsymbol{g} = (g_1, \ldots, g_C)$ used to determine the likelihood that each of the C candidates is the correct subkey :

$$g_k^Q = \sum_{i=1}^{Q} log(F[f(k, p_i), t_i]), \tag{1}$$

where Q is the number of attack traces and $f(\cdot)$ is the sensitive operation, The value $F[f(k, p_i), t_i]$ denotes the $f(k, p_i)$-th compotent of output of the neural network model, given the trace t_i as an input. It is interpreted as the probability assigned by the profiling model of obtaining the sensitive variable $f(k, p_i)$ corresponding to leakage trace t_i with a subkey hypothesis k and a plaintext p_i.

Evaluation metrics. Accuracy (defined as the successful classification rate) is the most common metric to evaluate a deep learning model. Nevertheless, while this metric is perfectly suitable for a general classification problem, it may not be suitable for SCA, as discussed by Cagli *et al.* [6]. Indeed, in SCA context it only corresponds to the success rate of a *simple attack*, *i.e.* a single-trace attack. When the evaluator can exploit several traces for varying plaintexts, the accuracy metric is not sufficient to evaluate the attack performance. In this case, it is worth considering SCA specific metric like the Empirical Guessing Entropy (GE) [24]. After the attack, the evaluator exploits the sorted logarithmic vector of candidate subkeys called the rank vector $\boldsymbol{r} = (r_1, r_2, ..., r_C) = Sort(\boldsymbol{g})$. In the vector, r_1 is considered the most likely subkey and r_C as the least likely. The position of the good subkey k^* in the vector is called the rank of the subkey.

$$Rank_Q(k^*) = \text{i such that } r_i^Q = k^* \tag{2}$$

Guessing entropy is defined as the expected rank of the correct subkey:

$$GE_Q = \mathbb{E}[Rank_Q(k^*)] \tag{3}$$

It may be estimated by the empirical average rank of the subkey k^*, among all the subkey hypothesis. This metric is estimated empirically, by performing the attack several times from different subsets of traces. In this work, we randomly select subsets from all available attack traces and set the number of attacks to 100. To compare the attack performance of our models, we will also look at another metric derived from the GE computation: N_a will denote the number of traces required for a successful attack. In other words, an attack is considered successful using N_a traces if the Guessing entropy is stably equal to 1.

$$N_a^* = min(Q|GE_Q(k^*) = 1) \tag{4}$$

2.3 Ensemble Learning

In ensemble learning, the models used within the ensemble usually perform poorly individually (slightly better than random guessing) and are called *weak models*. Ensemble methods combine the individual predictions of weak models via an aggregation method. The principle is to synergistically use an ensemble of several different weak models and to correctly combine their predictions in order to reduce their variance and their biases to obtain a more accurate and robust model called the *ensemble model*. As explained in [2], if the weak models errors are uncorrelated, the ensemble can be more efficient than any individual weak model. Combining the predictions of complementary weak models can thus improve the generalization, *i.e.* the ability of a trained model to perform on an unseen test set as well as on the training set. Concerning how to combine the weak models and build the ensemble model, in this work we use two types of meta-algorithms [3] called bagging [4] and stacking [25].

Bagging. The principle of bagging (depicted in Figure 1a) is to build several weak models (usually models of homogeneous types) independently, then aggregate them by an averaging or voting process to obtain the final predictions. The training set is usually subsampled to train the weak models on different subsamples of the training data. However in the SCA context the profiling phase requires a high number of traces. Thus, as the previous work of Perin *et al.* [20], we considered in this work the same training set for all weak models. The aggegation method that we used is the method proposed by Perin *et al.* [20] which consists of summing the log-probabilities during the attack phase. The new sum of the log-probabilities e based on the Equation 1 is calculated for each key byte hypothesis k :

$$e_k = \sum_{m=1}^{W} \sum_{i=1}^{Q} log(F[f(k, p_i), t_i]_m), \tag{5}$$

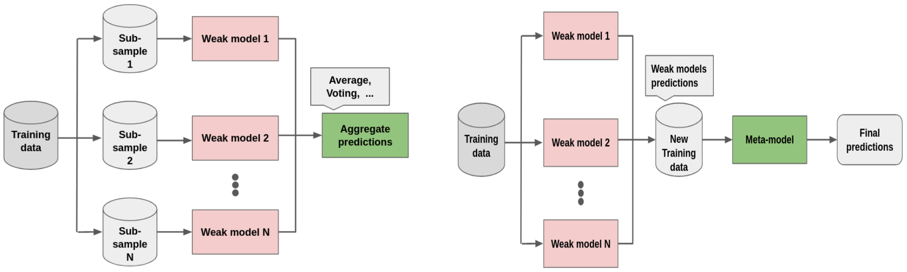
where W is the number of weak models.

Stacking. The principle of stacking (depicted in Figure 1b) differs mainly from bagging in three ways. First, in stacking, it is common to consider weak models of heterogeneous types. Second, weak models are usually trained on the same training set. Finally, to aggregate the weak models predictions, stacking uses a higher-level model, called *meta-model*, which is trained to produce new predictions.

2.4 Datasets

For all datasets, the experiments are implemented in Python 3.9 using the Keras 2.8 library and are run on a workstation equipped with 32GB RAM and a

[3] There are other ensemble methods, in particular the Boosting [9], which we have experimented without obtaining good enough performance for the considered datasets.



(a) Bagging ensemble learning.

(b) Stacking ensemble learning.

Fig. 1: Different ensemble learning methodologies.

NVIDIA Quadro P4000 with 8GB memory.

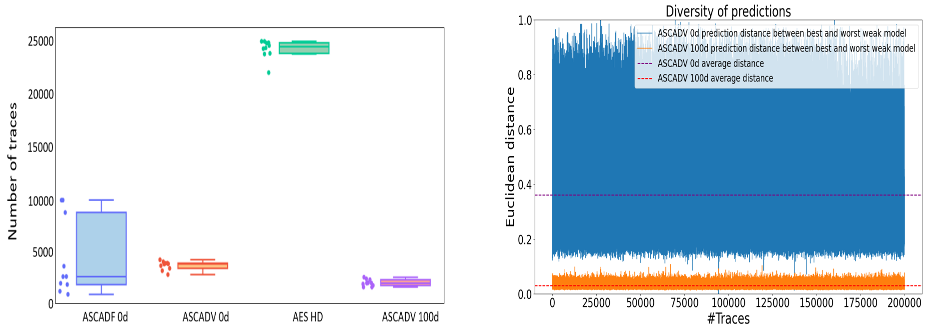
ASCADv1. This dataset contains traces of an 8-bit AVR microcontroller running a masked AES-128 implementation. There are actually two versions of the dataset that we will name ASCADF and ASCADV. ASCADF has a fixed key for training traces and consists of 50,000 traces for profiling and 10,000 for attack. Traces contain 700 time samples, a.k.a. features. ASCADV has variable keys for training traces and consists of 200,000 traces for profiling and 100,000 traces for attack. Traces contain 1400 features. Each of these versions also includes 3 variants to add a desynchronization type countermeasure of 0, 50 and 100 desynchronization samples respectively. The dataset was introduced in [3] and is publicly available at https://github.com/A-NSSI-FR/ASCAD.

AES HD. This dataset was introduced in [21]. It contains traces from an unprotected AES-128 hardware implementation. The AES HD dataset does not include any countermeasure but has the particularity to be very noisy. 50,000 traces are used for profiling and 25,000 are used for attack. Traces contain 1250 features. The dataset is publicly available at https://github.com/AESHD/AES-_HD_Dataset.

3 Experiments

3.1 Weak models

Number of weak models. Some works has investigated the question of how many weak models should be used in an ensemble [12, 13, 19]. To summarize, it appears that the optimal ensemble size depends on the problem and the performance of the weak models, so generally the ensemble size can be considered as another hyperparameter that can be searched by experimental analysis. Notably, Hansen and Salamon [12] suggested that ensemble with as few as 10 weak models were in general adequate to sufficiently reduce test error and improve general-



(a) Attack performance of the weak models on each datasets.

(b) ASCADV 0d and ASCADV 100d diversity of predictions.

Fig. 2: Weak models informations.

ization. Therefore in this work we have chosen to limit the scope of analysis to ensemble of up to 10 weak models.

Weak models training and results. To evaluate bagging and stacking ensemble learning methods, we trained for each dataset 10 neural networks, including MLPs and CNNs with some hyperparameters randomly selected from defined ranges. Varying models architectures will allow to learn different features from the same training set. Appendix A provides the search spaces of the weak models. An exception to such a random selection of hyperparameters is done in the case of ASCADV 100d dataset. Here, the convolutional part of the CNNs has been fixed once for all in order to deal with the high desynchronization and obtain weak models able to perform successfull (even if poorly performning) attacks independently. However, in Section 3.2.3 we explore some experimental results on ASCADV 50d where the convolutional part has not been fixed. Each network was trained with a stop criterion by monitoring the validation loss. The training and validation sets are obtained from the labeled data with a split ratio of 80%/20%. We focused onto a single byte of the AES secret key, and chose to target directly the corresponding first round Sbox output value as sensitive target variable. It may assume 256 possible values. The attack performance of the weak models are depicted in Figure 2a. It may be remarked that on ASCADF 0d we have a significant performance gap between the weak models. On the other hand on AES HD our weak models are all very poorly performing due to the very high noise level of the dataset. Figure 2b) depicts the Euclidean distance between the predictions of the best and worst weak models for the ASCADV 0d and ASCADV 100d datasets. A striking difference in diversity can be observed through the fact that fixing the convolutional part on ASCADV 100d resulted in weak models with very close predictions.

Choice of weak models. Once obtained 10 weak models, and in order to use a posteriori ensemble learning, we have to choose the models to use in the ensemble. Intuitively, a good ensemble is one where the individual weak models are both accurate and make their errors on different parts of the input space. Nevertheless as with the ensemble size, the choice of weak models to combine must often be found experimentally. Due to the performance gap and the lack of diversity on some datasets, we ranked our 10 weak models from the best performing in the attack (minimal Na) to the worst performing in order to increase ensemble size by decreasing attack performance.[4] We made this choice in order to take the point of view of an evaluator who knows the attack performance of his weak models and wishes to use a posteriori ensemble learning to improve his attack. This also allowed us to consider the ideal conditions for bagging and assess its limitations in this context.

3.2 Stacking implementation and results

Scope of the experimentations. Neural network stacking approach in SCA context has been mentioned by Perin *et al.* [20], but to the best of our knowledge it has never been explored in more detail in the literature. Our intuition is that a meta-model should perform better than an average process resulting from bagging. However, the complexity is increased by the addition of meta-model training, and we believe that this approach may be sensitive to the meta-model configuration. Therefore, rather than using a single hyperparameterized meta-model, our experiment consists in training 30 meta-models with random hyperparameter configurations. By analyzing the variability of the results for the 30 meta-models, we are able to verify if the stacking approach is robust or if it requires a careful meta-model hyperparameterization. We chose to use MLP as meta-models. Our 30 MLPs from the search space shown in Table 1 are the same for all ensemble sizes, in order to check the meta-models behavior when the ensemble size increases. In order to study the impact of stacking on our weak models, we compared the performance for different ensembles size with the performance of bagging and the best trained weak model. Our performance criterion is the convergence of Guessing Entropy. We trained the meta-models with the same training data of the weak models.[5]

Concatenation method. For the sake of completeness, we provide a description of the way we stacked the weak models predictions in our stacking experiments. We stack the weak models predictions in depth-wise sequence: if we have N weak models and each of them produces 256 value per prediction, we finally get a stacked prediction c of shape $256 * N$ to train the meta-models:

[4] Other criteria has been tested during the experimental campaign, but the obtained results were less performant and uninteresting in our opinion. Thus, they have been omitted.

[5] We also tried to train on the validation dataset, but the results were generally worse due to the lack of data. Results have thus been omitted.

$$c = [P_0^0, P_0^1, ..., P_0^N, ..., P_{256}^0, P_{256}^1, ..., P_{256}^N], \tag{6}$$

where P_i^j denotes the i-th compotent of output of the weak model j. Anyway, we remark that this choice should have no impact a priori on the learning process, since we use MLP as meta-models and the input layer of the MLP is fully connected (the order of the input thus not affect the possible functions it could model). Interestingly, we believe that this choice may have an important impact in cases where the meta-model is a different kind of model, for example a CNN. Indeed, CNN extracts information locally from the input, thus the proximity of the probability predictions for a same class could be a benefit for this kind of meta-model. An analysis of these impacts is left for future works.

Table 1: Hyperparameter search space for meta-models.

Hyperparameter	min	max	step
Number of layers	2	8	1
Number of neurons	100	1000	100
Activation	Relu, Elu, Selu, Gelu, Tanh		
Epoch	Early stopping : Val loss Patience 20		
Learning Rate	0.0001		
Mini Batch	100		
Optimizer	RMSprop		
Loss	Categorical Crossentropy : metric accuracy		

3.2.1 Results on ASCADF 0d and ASCADV 0d

Stacking results. The results of our experiments are summarized in Table 2. On ASCADF 0d, the best meta-model was trained on the predictions of the 4 best weak models. This meta-model successfully performed the attack in 203 traces, reducing the number of traces required by 81.69% (compared to 1109 traces for the best weak model). A similar performance improvement was obtained on ASCADV 0d with a meta-model trained on the predictions of the 5 best weak models that successfully performed the attack in 582 traces, reducing the number of traces needed to succeed in the attack by 80.42% (compared to 2973 traces for the best weak model). If we look at the performance of the best meta-models obtained on each ensemble size, we see that stacking improved the overall attacks performance on both datasets by 60% and 70% respectively. We also notice on both datasets that by increasing the ensemble size, the number of meta-models that improved attack performance tends to decrease. This is probably due to the fact that the addition of weak models and their knowledge make the meta-model learning task easier and on these two datasets the meta-models appeared to be too complex for the learning task. Thus the meta-models overfit immediately without learning relevant information. This may be confirmed by

Table 2: Stacking on ASCADF 0d and ASCADV 0d datasets. *Nb success* column refers to the number of meta-models that improved attack performance compared to the best weak model. Min, Max and Mean Na values are estimated considering only such *Nb success* meta-models. The best result is highlighted by a green cell.

Size of Ensemble	Nb success (Na <1109)	Na			Improvement in number of traces
		Min	Max	Mean	
2	30/30	371	853	576	66.54%
3	23/30	368	1098	696	66.81%
4	24/30	203	1064	680	81.69%
5	23/30	342	1062	674	69.16%
6	14/30	452	1043	588	59.24%
7	13/30	450	1070	604	59.42%
8	18/30	357	1086	666	67.80%
9	17/30	377	814	589	66.00%
10	15/30	427	989	631	61.49%

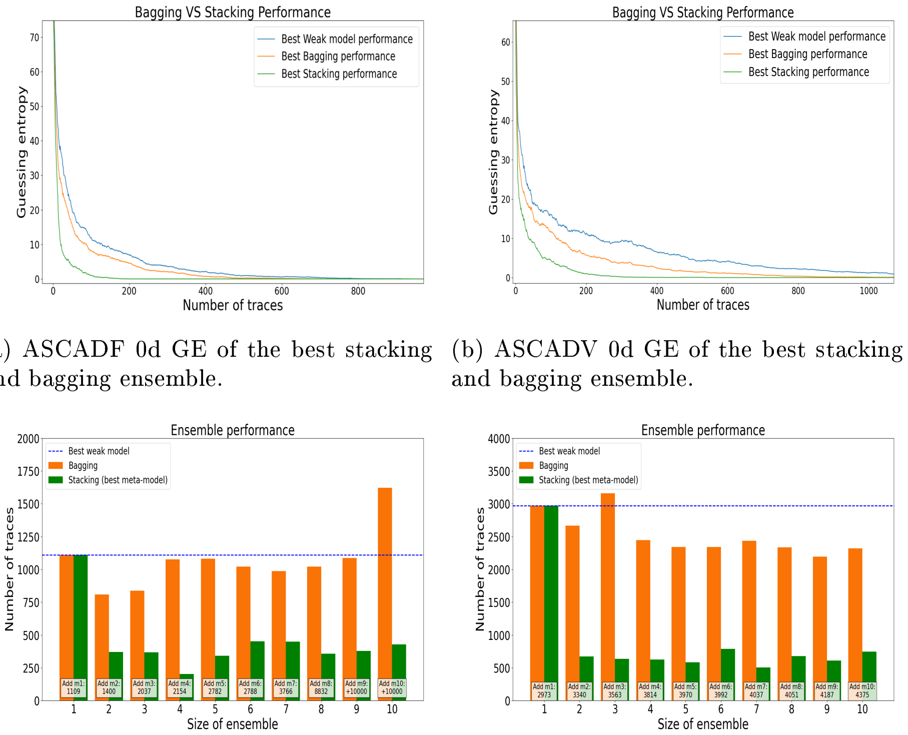
(a) Stacking on ASCADF 0d.

Size of Ensemble	Nb success (Na <2973)	Na			Improvement in number of traces
		Min	Max	Mean	
2	21/30	673	2448	1306	77.36%
3	11/30	635	2306	1533	78.64%
4	11/30	626	2879	1509	78.94%
5	9/30	582	2601	1341	80.42%
6	8/30	789	2693	1427	73.46%
7	7/30	604	2143	1327	79.68%
8	6/30	678	2475	1412	77.19%
9	5/30	607	2909	1606	79.58%
10	3/30	745	2502	1385	74.94%

(b) Stacking on ASCADV 0d.

observing the behaviour of the early-stopping mechanism : on ASCADF 0d up to an ensemble size of 5 weak models, the meta-models began to overfit in an average of 25 epochs, but from an ensemble size of 6, the average learning epoch before overfitting is only 2 epochs. The phenomenon is even more impressive on ASCADV 0d, where the meta-models began to overfit in an average of 3 epochs for the ensemble size of 2 weak models and 2 epochs for the other ensemble sizes.

Comparison with bagging. The best stacking and bagging attack performance on both datasets are depicted in Figure 3a and Figure 3b. The behavior of the attack performance across all ensemble sizes are depicted in Figure 3c and Figure 3d. We can see that stacking converges faster and allows us to obtain higher attack performance than bagging. In particular, we can observe from the results of ASCADF 0d that the bagging process is strongly impacted when our ensemble contains weak models with a significant performance gap. We observe that by adding less and less performing weak models in the ensemble, bagging becomes less and less suitable, until it loses its interest by obtaining lower attack performance than the best individual weak model. Interestingly, stacking aggregation is less impacted by the high variability in weak model performance since the meta-model learns the relevance of each weak model. Our intuition is confirmed by results on ASCADV 0d, where, in absence of a performance gap between weak models, the bagging aggregation works properly. However, we find that stacking aggregation significantly improves generalization and attack performance regardless of ensemble size.



(a) ASCADF 0d GE of the best stacking and bagging ensemble.

(b) ASCADV 0d GE of the best stacking and bagging ensemble.

(c) ASCADF 0d Comparison of stacking and bagging across all ensemble sizes.

(d) ASCADV 0d Comparison of stacking and bagging across all ensemble sizes.

Fig. 3: Comparison of stacking and bagging ensemble on ASCADF 0d and ASCADV 0d. For Figures (c) and (d), each box lying at the bottom of the i-th column, informs about the performance (in terms of Na) of the i-th weak model.

3.2.2 Results on AES HD and ASCADV 100d

Stacking results. The results of our experiments are summarized in Table 3. On ASCADV 100d the best meta-model was trained on the predictions of the 7 best weak models that successfully performed the attack in 351 traces, reducing the number of traces needed to succeed in the attack by 80.41% (compared to 1792 traces for the best weak model). If we look at the performance of the best meta-models obtained on each ensemble size, we see that stacking improved the overall attacks performance by 80%. An even greater performance improvement was obtained on AES HD with a meta-model trained on the predictions of the 9 best weak models. This meta-model successfully performed the attack in 1220 traces, reducing the number of traces required by 94.46% (compared to 22034 traces for the best weak model). If we look at the performance of the best meta-models obtained on each ensemble size, we see that stacking improved the

Table 3: Stacking on ASCADV 100d and AES HD datasets. *Nb success* column refers to the number of meta-models that improved attack performance compared to the best weak model. Min, Max and Mean Na values are estimated considering only such *Nb success* meta-models. The best result is highlighted by a green cell.

Size of Ensemble	Nb success (Na <1792)	Na			Improvement in number of traces
		Min	Max	Mean	
2	30/30	429	1172	808	76.06%
3	30/30	423	1256	735	76.39%
4	30/30	362	1160	763	79.79%
5	30/30	369	1141	711	79.40%
6	30/30	352	1070	700	80.35%
7	30/30	351	1130	742	80.41%
8	30/30	351	1333	741	80.41%
9	30/30	369	1097	737	79.40%
10	30/30	369	1137	717	79.40%

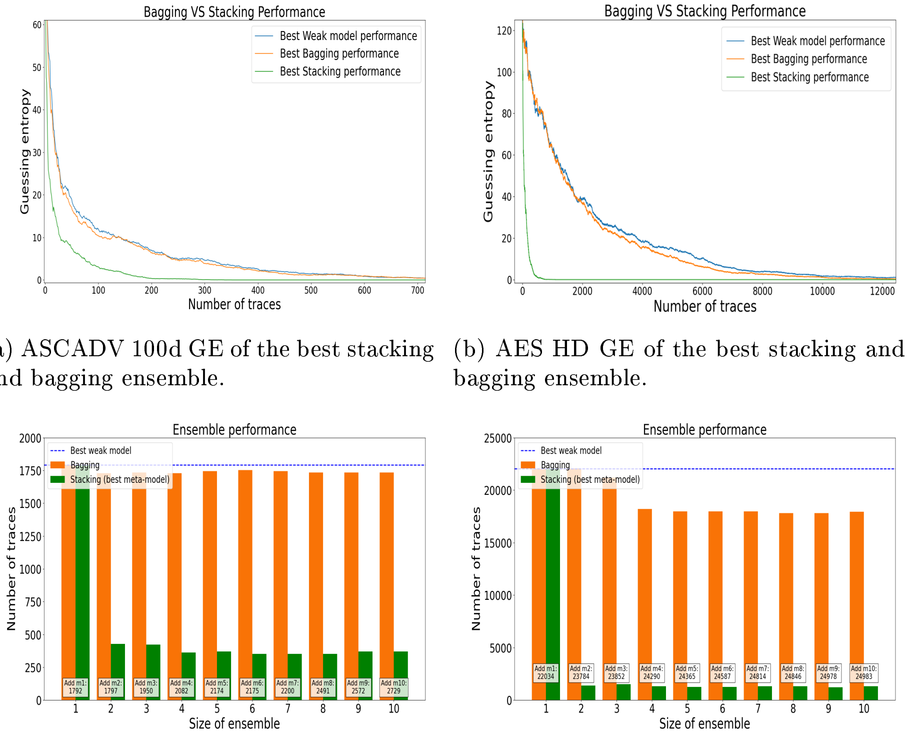
(a) Stacking on ASCADV 100d.

Size of Ensemble	Nb success (Na <22034)	Na			Improvement in number of traces
		Min	Max	Mean	
2	25/30	1365	4179	2212	93.80%
3	27/30	1507	20542	2704	93.16%
4	28/30	1324	11394	2286	93.99%
5	28/30	1251	8014	2038	94.32%
6	27/30	1253	9641	1988	94.31%
7	29/30	1324	12604	2377	93.99%
8	26/30	1315	8947	1962	94.03%
9	27/30	1220	4556	1865	94.46%
10	27/30	1318	9092	2106	94.01%

(b) Stacking on AES HD.

overall attacks performance by more than 90%. The use of stacking has shown significant interest for this dataset, allowing to obtain for all ensemble sizes performances below 2000 traces with ensemble of weak models that have independently attack performance above 20,000 traces. Unlike previous experiments, on these two datasets, stacking proved to be more robust, adding weak models did not decrease the number of meta-models that improved attack performance. This may be explained by the fact that the prediction tasks are more complex on these two datasets, due to the presence of desynchronization for ASCADV 100d and a very high level of noise for AES HD. Therefore the meta-models did not immediately overfit. On ASCADV 100d, we observe that all meta-models improved attack performance for all ensemble sizes.

Comparison with bagging. The best stacking and bagging attack performance on both datasets are depicted in Figure 4a and Figure 4b. The behavior of the attack performance across all ensemble sizes are depicted in Figure 4c and Figure 4d. We can see that stacking converges faster and allows us to obtain higher attack performance than bagging. On ASCADV 100d, another limit of the bagging process can be visualized. As reported in Section 3.1, in this experiment we fixed the convolutional part of the CNNs in order to quickly obtain weak models able to perform attacks independently. This resulted in weak models with very close predictions. Since the bagging process draws its strength from the diversity of the weak models, the lack of diversity in this case leads to a poorly performing or even worthless ensemble attack. Interestingly, compared to bagging, we noticed that stacking was not affected by the lack of diversity between the weak models. On AES HD, the weak models present naturally a good diversity, thus bagging process is performant. However, as for ASCADV 0d, the performance of bagging is limited by the individual performance of the weak models. In comparison, meta-model training results in better association of weak model predictions and much greater improvement in attack performance.



(a) ASCADV 100d GE of the best stacking and bagging ensemble.

(b) AES HD GE of the best stacking and bagging ensemble.

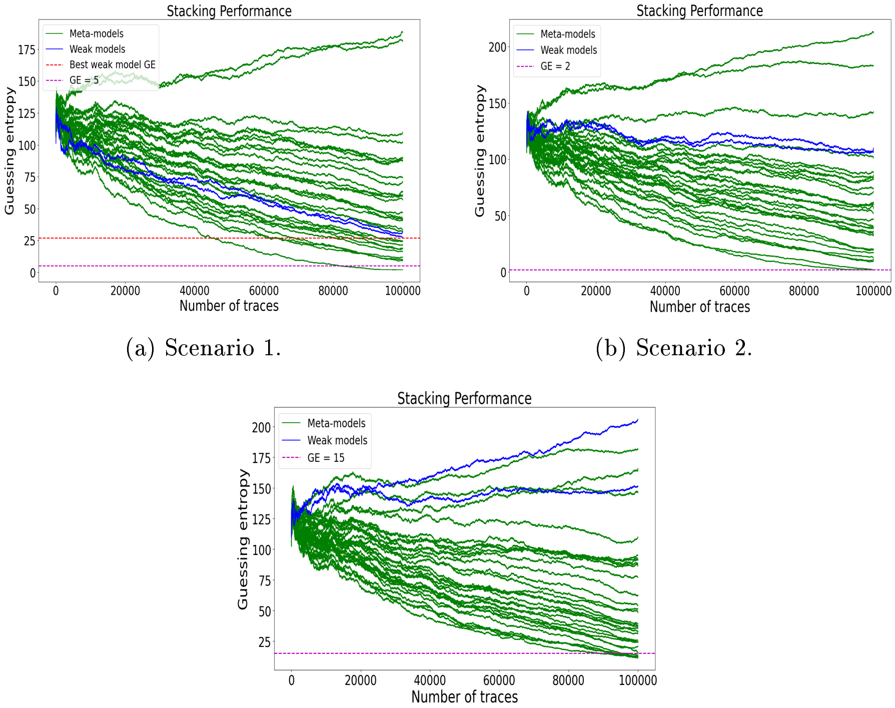
(c) ASCADV 100d Comparison of stacking and bagging across all ensemble sizes.

(d) AES HD Comparison of stacking and bagging across all ensemble sizes.

Fig. 4: Comparison of stacking and bagging ensemble on ASCADV 100d and AES HD. For Figures (c) and (d), each box lying at the bottom of the i-th column, informs about the performance (in terms of Na) of the i-th weak model.

3.2.3 Results on ASCADV 50d with full random CNNs

In previous experiments on the ASCADV dataset including desynchronization, in order to obtain individually performing weak models, we fixed the convolutional part of our weak models. In this experiment, in order to have a more objective analysis of the applicability of stacking in the presence of desynchronization, we trained new weak models in a completely random way by varying the convolutional part (*i.e.* including random hyperparameters selection for the convolutional part). As a result, we obtained very weak models that are not able to independently succeed in the attack with the 100,000 available attack traces. After applying stacking on different groups of two weak models, we found that in each case several meta-models were anyway able to improve GE convergence. The results of this experiment are shown in Figure 5. In the first scenario, the two weak models show a converging trend, but had not enough traces to succeed in the attack. We observe that the best trained meta-model reaches the same performance as the best weak model in less than 50,000 traces (instead of 100,000 traces). Moreover, when we consider all traces, it obtains an average rank under 5 instead of an average rank of 25 for the best weak model. In the second scenario, stacking is particularly interesting: the two weak models are just starting to converge after 100.000 attack traces, while some meta-models reached a rank of 2 with the available traces. Finally, in the third scenario, one of the two weak models converged by placing the correct subkey guess in last position. Interestingly, stacking was able to correct this effect to provide a correct convergence of the GE. Therefore, stacking appears to be robust independently of the weak models performance. This is encouraging for the interest of the method and its application in a real attack cases.



(a) Scenario 1.

(b) Scenario 2.

(c) Scenario 3.

Fig. 5: Stacking on ASCADV 50d with very weak models.

4 Discussion

4.1 Stacking aggregation : pros and cons

The comparison of the attack performance improvements of bagging and stacking obtained on each datasets are summarized in Table 4. If we compare the best performance obtained with the two aggregation methods, we observe that for all datasets, stacking has always been a better choice, allowing to obtain significant attack performance improvements compared to bagging and non-ensemble models. This significant improvement is due to the fact that stacking allows more sophisticated combinations and transformations of the weak model predictions by the training of the meta-model. Indeed, the meta-model is able to capture non-linear relationships between weak model predictions, which allows for more granular and accurate ensemble predictions. When the bagging was constrained by the individual performance of the weak models or the lack of diversity within the ensemble, some meta-models proved to be robust and allowed to improve the attack performance. This can be explained by the fact that a properly trained meta-model is able to correctly learn the relevance of each weak model while being able to learn consistent information even over small variations in ensemble predictions. Thus, the choice as well as the number of weak models is less determining when considering the stacking ensemble, which makes it a more flexible method for evaluators interested in using a posteriori ensemble learning. Interestingly, our experiments revealed that even with very few weak models, significant performance gains can already be achieved. This suggests that stacking has the ability to extract relevant information from a small subset of diverse weak models and that there is no need to consider an overly complex ensemble model. On the other hand, stacking has some drawbacks. Since the meta-model training is determinant in the success of the ensemble, this adds a new constraint to the success of the attack. The ensemble model has a higher complexity than the weak ones due to the addition of the meta-model. Finally, we observed that the ensemble model often proved to be too complex for the problem. Therefore, the meta-models tends to overfit quickly. Furthermore, the meta-model needs a lot of data to generalize properly. For example, our omitted experiments in training the meta-model on the validation dataset did not work well due to the lack of data.

Table 4: Comparison of bagging and stacking results on all datasets.

Dataset	Best weak model	Bagging improvement in number of traces	Stacking (best meta-model) improvement in number of traces
AES HD	22034	17798 (20%)	1220 (94%)
ASCADF 0d	1109	709 (28%)	203 (81%)
ASCADV 0d	2973	2194 (26%)	582 (80%)
ASCADV 100d	1792	1730 (3%)	351 (80%)

Table 5: Comparison in terms of performance with state-of-the-art architectures.

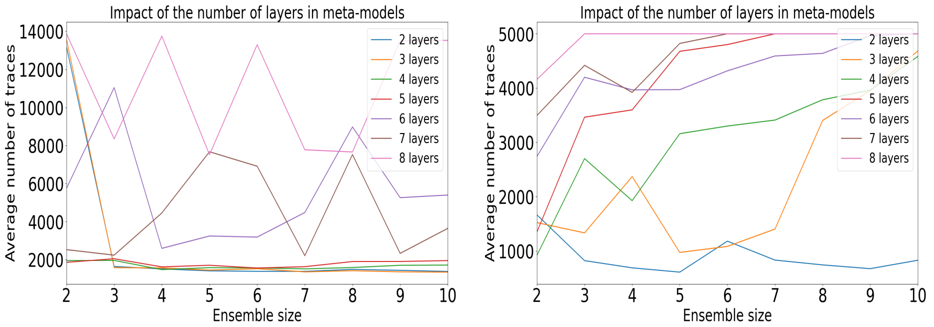
Dataset	Reference	Hyperparameterization method	Na
ASCADF 0d	Arch. in [3]	-	1146
	Arch. in [22]	Reinforcement learning.	202
	Our best Meta-model	4 random weak models with Na between [1109-2154]	203
ASCADV 0d	Arch. in [3]	-	1275
	Arch. in [22]	Reinforcement learning.	490
	Our best Meta-model	5 random weak models with Na between [2973-3970]	582
ASCADV 100d	Arch. 1 in [23]	-	3333
	Arch. 2 in [23]	Regularization technique.	347
	Our best Meta-model	7 random weak models with Na between [1792-2200]	351
AES HD	Arch. in [15]	-	25000
	Arch. in [27]	Visualization tools	1050
	Our best Meta-model	9 random weak models with Na between [22034-24983]	1220

4.2 Relieving hyperparameterization effort

In this section, we provide arguments to promote stacking as a technique to relieve the hyperparameterization effort for a security evaluator. To do so, we compare here for all datasets the performances of our best meta-model with different architectures, finely tuned, proposed in literature Table 5. Even if performance optimization was not at the core of the experiments, we observed that stacking ensemble can provide with less effort similar attack performance to rigorously hyperparameterized architectures. For ASCADF 0d and ASCADV 0d, we observe that with ensembles of weak models whose performance are similar (or worse) to those obtained with slightly hyperparameterized architectures [3], we obtain performance similar to high-performance architectures, which are hyperparameterized using reinforcement learning [22]. The interest of stacking is even more striking for AES HD, where with an ensemble of weak models (with individual performance higher than 20,000 traces), we obtain with less effort similar performance to high-performance architecture proposed by Zaid *et al.* [27] that are properly hyperparameterized. The main interest of stacking ensemble is to limit the hyperparameterization effort. The methodology proposed by Zaid *et al.* [27] make it possible to efficiently hyperparameterize its architecture and thus obtain a high-performance attack. However, they also assume much knowledge about datasets and an in-depth study of the impact of hyperparameters using data visualization tools (which can be time-consuming). Alternatively, the use of reinforcement learning proposed by Rijsdijk *et al.* [22] to automate hyperparameter search is effective, but the related process is extremely time-consuming. Furthermore, the experiments described in Sec. 3.2.3 are very representative: stacking is able to improve the attack performance even with very weak models that had (almost) not started to converge. This indicates an interest in the method in a more realistic attack scenario. Therefore, stacking may be considered as a suitable approach to avoid the need for the security evaluator to perform a fine tuned hyperparameterization of the neural network architecture.

4.3 Generalizable meta-model

In this section, we propose a quick analysis of the different meta-models we used in our experiment. The goal here was to deduce some general property to construct a good meta-model. If we take a look at a generalizable meta-model, we identified two meta-model architectures that proved to be suitable on all datasets. These two architectures, shown in Table 6, do not always achieve the best performance, but they always improved attack performance, on all datasets and all ensemble sizes. These appear to be the only two-layer architectures. We interpret this fact as a consenquence of the overfitting phenomenon: more complex meta-models often overfitted in our experiments. Especially on datasets that do not include desynchronization, where we can clearly see that increasing the number of layers in the meta-models degrades the attack performance (Figure 6). Therefore, a small architecture with few layers is more appropriate for the meta-model.



(a) AES HD.

(b) ASCADV 0d.

Fig. 6: Impact of the number of layers on meta-model performance.

Table 6: Meta-models that always improve attack performance.

Hyperparameter	Architecture 1	Architecture 2
Number of layers	**2**	**2**
Number of neurons	600	300
Activation	elu	tanh
Epoch	Early stopping : Val loss Patience 20	
Learning Rate	0.0001	
Mini Batch	100	
Optimizer	RMSprop	
Loss	Categorical Crossentropy : metric accuracy	

5 Conclusion and Future works

In this work, we propose a new study of Deep Ensemble Learning in the side-channel context. We extend the preliminary results of Perin *et al.* [20] who used bagging to aggregate the predictions of the weak models and we propose to use stacking as a more suitable choice in the aggregation method. Our experimental exploration on several publicly available datasets highlights some of the limitations of the bagging process and shows that stacking can significantly improve attack performance while providing a flexible solution to address these limitations. During our experiments, we observed that stacking ensemble can provide with less effort attack performance similar to those of rigorously hyperparameterized architectures. Therefore, stacking may be considered as a suitable approach to avoid the need for the security evaluator to finely tune the neural network architecture. However, stacking ensemble has proven to be extremely sensitive to overfitting, making it crucial to avoid using overly complex meta-models. In our experiments, two-layer meta-models have always succeeded in improving attack performance. We also noticed that the improvement in attack performance was not correlated with the number of weak models in the ensemble. Indeed, we often found similar improvements across all ensemble sizes. Thus, since the complexity increases with the addition of weak models, we recommend using an ensemble with few weak models.

Future works. In our experimental campaign, our simplest meta-models consisted of two layers of 100 neurons. We think it would be interesting to further simplify the networks by experimenting with single-layer architectures with even fewer neurons. We also plan to extend this work by using a small CNN as a meta-model to take advantage of the concatenation in depth-wise sequence. Moreover, it would be interesting to study the applicability of the boosting ensemble methodology in SCA.

Acknowledgements

This work was financially supported by the Defense Innovation Agency (AID) from the french ministry of armed forces.

References

1. Acharya, R.Y., Ganji, F., Forte, D.: Information theory-based evolution of neural networks for side-channel analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 401–437 (2023)
2. Ali, K.M., Pazzani, M.J.: On the link between error correlation and error reduction in decision tree ensembles (1995)
3. Benadjila, R., Prouff, E., Strullu, R., Cagli, E., Dumas, C.: Deep learning for side-channel analysis and introduction to ascad database. Journal of Cryptographic Engineering **10**(2), 163–188 (2020)

4. Breiman, L.: Bagging predictors. Machine learning **24**, 123–140 (1996)
5. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Cryptographic Hardware and Embedded Systems-CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings 6. pp. 16–29. Springer (2004)
6. Cagli, E., Dumas, C., Prouff, E.: Convolutional neural networks with data augmentation against jitter-based countermeasures: Profiling attacks without preprocessing. In: Cryptographic Hardware and Embedded Systems–CHES 2017: 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings. pp. 45–68. Springer (2017)
7. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Cryptographic Hardware and Embedded Systems-CHES 2002: 4th International Workshop Redwood Shores, CA, USA, August 13–15, 2002 Revised Papers 4. pp. 13–28. Springer (2003)
8. Destouet, G., Dumas, C., Frassati, A., Perrier, V.: Wavelet scattering transform and ensemble methods for side-channel analysis. In: Constructive Side-Channel Analysis and Secure Design: 11th International Workshop, COSADE 2020, Lugano, Switzerland, April 1–3, 2020, Revised Selected Papers 11. pp. 71–89. Springer (2021)
9. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of computer and system sciences **55**(1), 119–139 (1997)
10. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: Cryptographic Hardware and Embedded Systems—CHES 2001: Third International Workshop Paris, France, May 14–16, 2001 Proceedings 3. pp. 251–261. Springer (2001)
11. Gao, F., Mao, B., Wu, L., Wang, Z., Mu, D., Hu, W.: Leveraging ensemble learning for side channel analysis on masked aes. In: 2021 7th International Conference on Computer and Communications (ICCC). pp. 267–271. IEEE (2021)
12. Hansen, L.K., Salamon, P.: Neural network ensembles. IEEE transactions on pattern analysis and machine intelligence **12**(10), 993–1001 (1990)
13. Hernández-Lobato, D., Martínez-Muñoz, G., Suárez, A.: How large should ensembles of classifiers be? Pattern Recognition **46**(5), 1323–1336 (2013)
14. Heuser, A., Zohner, M.: Intelligent machine homicide: Breaking cryptographic devices using support vector machines. In: Constructive Side-Channel Analysis and Secure Design: Third International Workshop, COSADE 2012, Darmstadt, Germany, May 3-4, 2012. Proceedings 3. pp. 249–264. Springer (2012)
15. Kim, J., Picek, S., Heuser, A., Bhasin, S., Hanjalic, A.: Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 148–179 (2019)
16. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings 19. pp. 388–397. Springer (1999)
17. Lerman, L., Bontempi, G., Markowitch, O.: A machine learning approach against a masked aes: Reaching the limit of side-channel attacks with a learning model. Journal of Cryptographic Engineering **5**, 123–139 (2015)
18. Maghrebi, H., Portigliatti, T., Prouff, E.: Breaking cryptographic implementations using deep learning techniques. In: Security, Privacy, and Applied Cryptography Engineering: 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings 6. pp. 3–26. Springer (2016)

19. Opitz, D., Maclin, R.: Popular ensemble methods: An empirical study. Journal of artificial intelligence research **11**, 169–198 (1999)
20. Perin, G., Chmielewski, Ł., Picek, S.: Strength in numbers: Improving generalization with ensembles in machine learning-based profiled side-channel analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 337–364 (2020)
21. Picek, S., Heuser, A., Jovic, A., Bhasin, S., Regazzoni, F.: The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. IACR Transactions on Cryptographic Hardware and Embedded Systems **2019**(1), 1–29 (2019)
22. Rijsdijk, J., Wu, L., Perin, G., Picek, S.: Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 677–707 (2021)
23. Robissout, D., Bossuet, L., Habrard, A., Grosso, V.: Improving deep learning networks for profiled side-channel analysis using performance improvement techniques. ACM Journal on Emerging Technologies in Computing Systems (JETC) **17**(3), 1–30 (2021)
24. Standaert, F.X., Malkin, T.G., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Advances in Cryptology-EUROCRYPT 2009: 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings 28. pp. 443–461. Springer (2009)
25. Wolpert, D.H.: Stacked generalization. Neural networks **5**(2), 241–259 (1992)
26. Wu, L., Perin, G., Picek, S.: I choose you: Automated hyperparameter tuning for deep learning-based side-channel analysis. IEEE Transactions on Emerging Topics in Computing (2022)
27. Zaid, G., Bossuet, L., Habrard, A., Venelli, A.: Methodology for efficient cnn architectures in profiling attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 1–36 (2020)
28. Zaid, G., Bossuet, L., Habrard, A., Venelli, A.: Efficiency through diversity in ensemble models applied to side-channel attacks:–a case study on public-key algorithms–. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 60–96 (2021)

A Weak models

Hyperparameter	min	max	step
Learning Rate	0.0001	0.001	0.0001
Mini Batch	100	1000	100
Nb conv layers	2	8	1
Filters	8	32	4
Kernel size	10	20	2
Stride	5	10	5
Nb FC layers	2	3	1
Nb FC neurons	100	1000	100
Activations	Relu, Elu, Selu, Gelu, Tanh		

(a) Search space for ASCADF 0d / AES HD CNN hyperparameters.

Fixed Conv part			
Conv(32, 1) , AveragePooling(2, 2)			
Conv(64, 25), AveragePooling(25, 25)			
Conv(128, 3), AveragePooling(4, 4)			
Random Dense part			
Hyperparameter	min	max	step
Learning Rate	0.0001	0.001	0.0001
Mini Batch	100	1000	100
Nb FC layers	2	4	1
Nb FC neurons	500	4000	500
Activations	Relu, Elu, Selu, Gelu, Tanh		

(b) Search space for ASCADV 50d / ASCADV 100d CNN hyperparameters.

Hyperparameter	min	max	step
Learning Rate	0.0001	0.001	0.0001
Mini Batch	100	1000	100
Kernel size	16	128	16
Nb conv layers	1	4	1
Nb FC layers	1	3	1
Nb FC neurons	500	4000	500
Filters	1		
Strides	2		
Activations	Relu, Elu, Selu, Gelu, Tanh		

(c) Search space ASCADV 50d full random CNN.

Hyperparameter	min	max	step
Learning Rate	0.0001	0.001	0.0001
Mini Batch	50	1000	100
Nb FC layers	2	8	1
Nb FC neurons	100	1000	100
Dropout	0.0	0.4	0.1
Activations	Relu, Elu, Selu, Gelu, Tanh		

(d) Search space for ASCADF 0d/ AES HD/ ASCADV 0d MLP hyperparameters.

Fig. 7: Search space for weak models.