Gambling for Success: The Lottery Ticket Hypothesis in Deep Learning-based SCA

Guilherme Perin[1], Lichao Wu[1], and Stjepan Picek[1,2]

[1] Delft University of Technology, The Netherlands
[2] Radboud University, The Netherlands

Abstract. Deep learning-based side-channel analysis (SCA) represents a strong approach for profiling attacks. Still, this does not mean it is trivial to find neural networks that perform well for any setting. Based on the developed neural network architectures, we can distinguish between small neural networks that are easier to tune and less prone to overfitting but could have insufficient capacity to model the data. On the other hand, large neural networks have sufficient capacity but can overfit and are more difficult to tune. This brings an interesting trade-off between simplicity and performance.
This work proposes to use a pruning strategy and recently proposed Lottery Ticket Hypothesis (LTH) as an efficient method to tune deep neural networks for profiling SCA. Pruning provides a regularization effect on deep neural networks and reduces the overfitting posed by overparameterized models. We demonstrate that we can find pruned neural networks that perform on the level of larger networks, where we manage to reduce the number of weights by more than 90% on average. This way, pruning and LTH approaches become alternatives to costly and difficult hyperparameter tuning in profiling SCA. Our analysis is conducted over different masked AES datasets and for different neural network topologies. Our results indicate that pruning, and more specifically LTH, can result in competitive deep learning models.

Keywords: Side-channel Analysis · Deep learning · Lottery Ticket Hypothesis · Pruning

1 Introduction

Several side-channel analysis (SCA) approaches exploit various sources of information leakage from electronic devices. Common examples of side channels are timing [16], power [17], and electromagnetic (EM) emanation [26]. Besides a division based on side channels, it is possible to divide SCA based on the attacker's capabilities into non-profiling and profiling attacks. Non-profiling attacks require fewer assumptions but often require thousands to millions of measurements (traces) to break a target, especially if protected with countermeasures. Profiling attacks are considered one of the strongest possible attacks as the attacker has control over a clone device to build its complete profile [8]. This profile is given by

a parametric statistical model used by the attacker to generalize to side-channel information collected from similar devices to recover the secret information.

The history of profiling side-channel analysis (SCA) spans around 20 years, and it is possible to distinguish among several research directions. The first direction used techniques like (pooled) template attack [8,9] or stochastic models [28] and managed to improve the attack performance over non-profiling attacks significantly. Then, the second direction moved toward machine learning in SCA, and again, a plethora of results [25,13,18] indicated that machine learning could outperform other profiling SCA methods. More recently, as the third direction, we see a change of focus to deep learning techniques. Intuitively, we can find at least two reasons for this: 1) deep learning show superior practical results in breaking targets protected with countermeasures [20], and 2) deep learning does not require pre-processing like feature selection [23] or dimensionality reduction [4]. While the SCA community progressed quite far in the deep learning-based SCA in just a few years, there are many knowledge gaps. One example would be how to successfully and systematically find neural networks that manage to break various targets.

Thus, we still need to find approaches that allow designing neural networks that perform well for various targets. We aim to have an approach that transforms a good-performing architecture for one scenario into a good-performing architecture for a different scenario. Finally, it would be ideal if the top-performing architectures could be small (so they are more computationally efficient, and hopefully, easier to understand). Unfortunately, this is not easy as the search space turns into infinite neural network configuration possibilities. There are no general guidelines on how to construct a neural network that will break a target. Current efforts mainly concentrate on finding better hyperparameters by defining modest and optimal ranges for hyperparameters, resulting in large and exhaustive search spaces. Examples of applied hyperparameters search in profiling SCA are random search [22], Bayesian optimization [32], reinforcement learning [27], or approaches following a specific methodology [35,31]. Still, there are alternatives to how to provide neural networks that are small and perform well. Note that larger neural networks can also perform well for profiling SCA. However, they suffer more from overfitting, and tuning large models becomes more difficult due to the increased hyperparameter search spaces. Regularization techniques are indicated to correct large models by limiting their capacity, although constructing efficient regularizers can be a highly complex task.

In the machine learning domain, there is a technique called *pruning* (or *sparsification*) that refers to a systematical removal of parameters from neural networks. Commonly, pruning is used on large neural networks that show good performance. The goal is to produce a smaller network with similar performance to be deployed in memory-constrained devices. Also, pruning offers an alternative and cheap solution for regularizing large deep learning models. While pruning [14,6] is a rather standard technique in deep learning, it has not been investigated before in the SCA domain to the best of our knowledge. Similarly,

the Lottery Ticket Hypothesis [10] attracted quite some attention in the machine learning community, but none (as far as we know) in the SCA community.

This chapter applies the recent *Lottery Ticket Hypothesis* (LTH) in the profiling side-channel analysis. After training a (relatively large) neural network, we apply the pruning process by removing the activity of small weights from the neural network. We then re-initialize the pruned neural network with the same initial weights set for the original large neural network. The pruned and re-initialized network shows equal or, most of the time, superior performance compared to the baseline trained network. We emphasize:

- Pruning is convenient for deep neural networks that overfit. Finding efficient and small networks is more difficult than starting with a large model and then pruning it. In this chapter, we consider neural network architectures with up to one million trainable parameters.
- Pruning has two main advantages for SCA: (1) When the baseline model is not carefully tuned and overfits or underfits, pruning (and specially LTH process) may "tune" the model size. (2) Pruning acts as a strong regularizer, which is important for noisy and small SCA datasets. Moreover, techniques such as explainability and interpretability can be used to define pruning strategies efficiently.

The results demonstrate that when the large network cannot reach a successful attack (low guessing entropy), applying the Lottery Ticket Hypothesis leads to a successful key recovery, even when the number of profiling traces is low. More importantly, we verify that when training a large deep neural network provides guessing entropy close to a random guess, a pruned and re-initialized neural network can reduce the entropy of the target key. Our main contributions are:

1. We introduce the pruning approach into profiling SCA, enabling us to propose a procedure that can work on top of other approaches. Our approach can be applied to any neural network, regardless of whether it is selected randomly or obtained through some other methodology. Naturally, depending on how good is the original network, the results from our approach can differ.
2. We demonstrate that the Lottery Ticket Hypothesis holds for SCA, which is a significant finding due to different metrics used in SCA. The original publication [10] measures LTH efficiency through test accuracy. Here, our metric is guessing entropy from the attack traces. As reported in this chapter, we can find smaller, better, and stable networks by using the pruning and weight initialization based on LTH, even when the original network does not return successful attack results.

2 Background

2.1 Notation

Let calligraphic letters like $\mathcal{X}$ denote sets, and the corresponding upper-case letters X denote random variables and random vectors $\mathbf{X}$ over $\mathcal{X}$. The corre-

sponding lower-case letters x and $\mathbf{x}$ denote realizations of X and $\mathbf{X}$, respectively. Next, let k be a key candidate that takes its value from the keyspace $\mathcal{K}$, and k^* the correct key. We define a dataset as a collection of traces $\mathbf{T}$, where each trace $\mathbf{t}_i$ is associated with an input value (plaintext or ciphertext) $\mathbf{d}_i$ and a key $\mathbf{k}_i$. When considering only a specific key byte j, we denote it as $k_{i,j}$, and input byte as $d_{i,j}$.

The dataset consists of $|T|$ traces. From $|T|$ traces, we use N traces for the profiling set, V traces for the validation set, and Q traces for the attack set. Finally, θ denotes the vector of parameters to be learned in a profiling model, and $\mathcal{H}$ denotes the hyperparameters defining the profiling model.

2.2 Supervised Machine Learning in Profiling SCA

Supervised machine learning considers the machine learning task of learning a function f mapping an input X to the output Y ($f : \mathcal{X} \rightarrow Y$) based on input-output pairs. The function f is parameterized by $\theta \in \mathbb{R}^n$, where n represents the number of trainable parameters.

Supervised learning happens in two phases: training and test, corresponding to SCA's profiling and attack phases. Thus, in the rest of this chapter, we use the terms profiling/training and attack/testing interchangeably. As the function f, we consider a deep neural network with the *Softmax* output layer.

The goal of the training phase is to learn parameters θ' that minimize the empirical risk represented by a loss function L on a dataset T of size N.

In the attack phase, the goal is to make predictions about the classes

$$y(t_1, k^*), \ldots, y(t_Q, k^*),$$

where k^* represents the secret (unknown) key on the device under the attack (or the key byte). The outcome of predicting with a model f on the attack set is a two-dimensional matrix P with dimensions equal to $Q \times c$ (the number of classes c depends on the leakage model as the class label v is derived from the key and input through a cryptographic function and a leakage model). To reach the probability that a certain key k is the correct one, we use the maximum log-likelihood approach:

$$S(k) = \sum_{i=1}^{Q} \log(\mathbf{p}_{i,v}). \tag{1}$$

The value $\mathbf{p}_{i,v}$ denotes the probability that for a key k and input d_i, we obtain the class v.

We are interested in reaching good generalization with machine learning algorithms, denoting how well the concepts learned by a machine learning model apply to previously unseen examples. At the same time, we aim to avoid underfitting and overfitting. Overfitting happens when a model learns the detail and noise in the training data, negatively impacting the model's performance on unseen data. Underfitting happens with a model that cannot model the training data or generalize to unseen data.

In SCA, an adversary is not interested in predicting the classes in the attack phase but in obtaining the secret key k^*. To estimate the effort required to obtain the key, we will use the guessing entropy (GE) or success rate metrics [29]. An attack outputs a key guessing vector $\mathbf{g} = [g_1, g_2, \ldots, g_{|\mathcal{K}|}]$ in decreasing order of probability, which means that g_1 is the most likely key candidate and $g_{|\mathcal{K}|}$ the least likely key candidate. The success rate is the average probability that the secret key k^* is the first element of the key guessing vector $\mathbf{g}$. Guessing entropy is the average position of k^* in $\mathbf{g}$. Commonly, averaging is done over 100 independent experiments to obtain statistically significant results. As common in the deep learning-based SCA, we consider multilayer perceptron (MLP) and convolutional neural networks (CNNs).

2.3 Leakage Models and Datasets

During the execution of the cryptographic algorithm, the processing of sensitive information produces a specific leakage. In this chapter, we consider the Hamming weight leakage model since the considered datasets leak significantly in this model .There, the attacker assumes the leakage is proportional to the sensitive variable's Hamming weight. This leakage model results in nine classes when considering a cipher that uses an 8-bit S-box ($c = 9$).

ASCAD Datasets. The first target platform we consider is an 8-bit AVR microcontroller running a masked AES-128 implementation [5]. There are two versions of the ASCAD dataset. The first version of the ASCAD dataset has a fixed key and 50 000 traces for profiling and 10 000 for testing.The second version of the ASCAD dataset has random keys, and it consists of 200 000 traces for profiling and 100 000 for testing. For both versions, we attack the key byte 3 unless specified differently. For the ASCAD dataset, the third key byte is the first masked byte. For ASCAD with the fixed key, we use a pre-selected window of 700 features, while for ASCAD with random keys, the window size equals 1 400 features. These datasets are available at [1].

CHES CTF 2018 Dataset. This dataset refers to the CHES Capture-the-flag (CTF) AES-128 dataset, released in 2018 for the Conference on Cryptographic Hardware and Embedded Systems (CHES). The traces consist of masked AES-128 encryption running on a 32-bit STM microcontroller. We use 45 000 traces for the training set (CHES CTF Device C), containing a fixed key. The attack set consists of 5 000 traces (CHES CTF Device D). The key used in the training and validation set is different from the key configured for the test set. CHES CTF 2018 trace sets contain the power consumption of the full AES-128 encryption, with a total number of 650 000 features per trace. The raw traces were pre-processed in the following way. First, a window resampling is performed. Later, we concatenated the trace intervals representing the processing of the masks (beginning of the trace) with the samples indicating the processing of S-boxes located after an interval without any particular activity (flat power consumption profile). The resulting traces have 2 200 features. The original dataset is available at [2], and the processed traces are provided at [3].

3 Related Works

The goal of finding neural networks that perform well in SCA is probably the most explored direction in machine learning-based SCA. The first works commonly considered multilayer perceptron and reported good results even though there were not many available details about hyperparameter tuning or the best-obtained architectures [11,21,33,12]. In 2016, Maghrebi et al. made a significant step forward in the profiling SCA as they investigated the performance of convolutional neural networks [20]. Since the results were promising, this paper started a series of works where deep learning techniques (most dominantly MLP and CNNs) were used to break various targets efficiently.

Soon after, works from Cagli et al. [7], Picek et al. [24], and Kim et al. [15] demonstrated that deep learning could efficiently break implementations protected with countermeasures. While those works also discuss hyperparameter tuning, it was still not straightforward to understand the effort required to find the neural networks that performed well. This effort became somewhat clearer after Benadjila et al. investigated hyperparameter tuning for the ASCAD dataset [5]. Indeed, while considering only a subset of possible hyperparameters, the tuning process was far from trivial.

Zaid et al. proposed a methodology for CNNs for profiling SCA [35]. While the methodology has limitations, the results obtained are significant as they reached top performance with never smaller deep learning architectures. This direction is further investigated by Wouters et al. [31] who reported some issues with [35] but managed to find even smaller neural networks that perform similarly well. Still, the proposed methodologies have some issues. First, it is not easy to use those methodologies and generalize for other datasets or neural network architectures. Second, the conflicting results among those methodologies indicate it is difficult to find a single approach that works the best for everything.

Perin et al. conducted a random search in pre-defined ranges to build deep learning models to form ensembles [22]. Their findings showed that even random search (when working on some reasonable range of hyperparameters) could find neural networks that perform extremely well. Finally, van ver Valk et al. used a technique called mimicking to find smaller neural networks that perform like the larger ones [30]. Still, the authors did not use pruning but ran experiments until they found a smaller network that outputs the same results as the larger one. Thus, the approaches are significantly different.

Thus, while the approaches mentioned work as evident from the excellent attack performance, there are still unanswered questions. What is clear is that we can reach good results with (relatively) small neural networks. What remains to be answered is how to adapt those methodologies for different datasets, or can we find even smaller neural network architectures that perform as well (or better). We aim to provide the answers to those questions in this work.

4 The Lottery Ticket Hypothesis (LTH)

The Lottery Ticket Hypothesis (LTH) was originally proposed by Franke and Carbin in [10] as a technique to improve pruned neural network performances. The main goal of pruning is to remove unnecessary weights to achieve the smallest neural network by keeping the original baseline performance. The baseline model refers to the trained neural network architecture that is not pruned. This way, pruned neural networks are suitable for memory-constrained devices and can deliver faster inference. With the LTH, authors verified that re-initializing the pruned neural network with the same initial weights from the baseline neural network shows equivalent or superior performance to the baseline model. In short, authors define the following: *"**Lottery Ticket Hypothesis:** a randomly initialized dense neural network contains a sub-network that is initialized such that - when trained in isolation - it can match the test accuracy of the original network after training for at most the same number of iterations".*

A fixed sparsity level gives the amount of pruned weights and denotes the percentage of the removed network (e.g., 90% sparsity on an MLP would remove 90% of weight connections). The top-performing sub-networks are then called the *winning tickets*. There are two main ways to deploy LTH: one-shot pruning and iterative pruning. The latter defines the next sparsity level according to the results obtained from the previously evaluated sparsity level amount. This way, instead of defining a fixed sparsity level as in the case of one-shot pruning, the process iteratively finds the maximum possible sparsity level that delivers satisfactory results.

4.1 Pruning Strategy

To find an efficient pruned network, the large overparameterized baseline model must be trained before applying pruning to remove unnecessary weights. The pruning process applied in this work removes the smallest weights from the trained weights obtained from training the baseline model for a fixed amount of epochs. The activations in the forward propagation are mostly affected by larger weight values. Therefore, pruning the smallest weights remove those weights that are not significantly impacting the predictions. Different pruning strategies could be considered. Here we show that even the most simple method based on weight magnitude already delivers efficient results.

As shown in the experimental results section, we apply the LTH process on public (and protected) AES datasets, which also works when considering other than accuracy performance metrics (e.g., success rate, guessing entropy). The process starts by training an overparameterized neural network model for a single target AES key byte (note that our baseline models are assumed to be overparameterized in comparison to state-of-the-art works, e.g., [35,31,27]). Afterward, the model is pruned by removing smallest weights, and this pruned model is re-initialized and retrained (with more efficiency, e.g., fewer epochs) for all AES key bytes. Therefore, LTH reduces the complexity of deep neural network tuning in profiling SCA. We give the pruning strategy procedure for

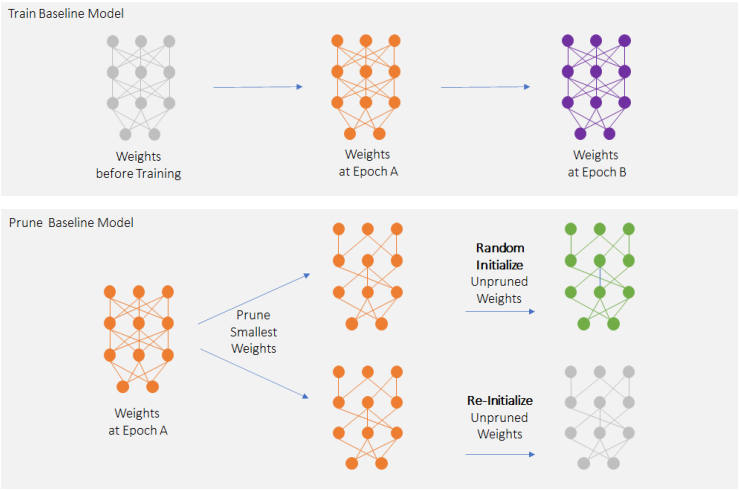


Fig. 1: One-shot pruning procedure for LTH.

LTH in Algorithm 1. Although our process iterates over all sparsity levels s (in our case, from 1% to 99%), we do not consider it as iterative pruning because we do not set a metric to stop the process. Our main goal is to evaluate the profiling attack performance for all evaluated sparsity levels.

Algorithm 1 Pruning Strategy.

procedure PRUNING STRATEGY(original neural network f, original dataset x, random initial weights θ_0, training epoch θ_j, trained weight θ_j, pruning ratio $P_{\%}$, mask m)
  for s=1 to 99 **do**
    $\theta_j \leftarrow$ Pretrain Model $f(x,\theta_0)$ for j epochs
    $m \leftarrow$ Prune s% of the smallest weights from θ_j
    for i=1 to j **do**
      Train $f(x,\theta_0 \odot m)$
    end for
  end for
end procedure

In Figure 1, we depict an one-shot pruning procedure. The first part of the figure displays the reference training procedure with no pruning where the weights at the beginning of the training process are different from those at epochs A and B. The lower figure shows the setup when we prune the smallest weights and are left to choose whether we randomly initialize the remaining weights or re-initialize them from the original weights.

4.2 Winning Tickets in Profiling SCA

In [10], a *winning ticket* is defined as a sub-network that, when trained in isolation (after being re-initialized with the same baseline model initial weights), provides classification accuracy equivalent or superior to the baseline model. For profiling SCA, we define winning ticket as a sub-network that provides a test guessing entropy lower than or equivalent to the guessing entropy obtained from the original baseline model. Note that hyperparameters defined for the baseline model (which affects the total number of training parameters) and the number of profiling traces directly affect the chances to identify a winning ticket as demonstrated in Section 5.

Recall, pruning refers to removing neurons (neuron-based pruning) or weight connections (weight-based pruning) from the neural network activity. The most popular pruning technique consists of keeping a number of weight connections based on their weight value. This means that the smallest weights are pruned out from the model [3]. However, one should note that the concept of *winning ticket* does not imply that pruning is applied to well-selected pruned weights or neurons. For instance, if one prunes a certain percentage of elements selected at random and the remaining sub-network still performs as well as the baseline model, the resulting model is still called a winning ticket. Obviously, pruning techniques should also be explored to find a sub-network with more efficiency. In Section 5, we provide an extensive set of experimental results showing that pruning the smallest weights provides excellent results for SCA. Still, we do not claim that pruning, e.g., random weights, would not give good results for specific settings.

Ideally, deep learning-based SCA requires selecting the smallest possible neural network architecture that provides good generalization for a given target. Small models are faster to train and easier to interpret. The challenge of finding a well-performing small architecture may grow proportionally to the difficulty of the evaluated side-channel dataset (misalignment, noise, countermeasures). Nevertheless, side-channel traces usually provide a low signal-to-noise ratio, and regularization techniques play an important role in leakage learnability. Small models are self-regularized, mainly because they offer less capacity to overfit the training set. This justifies the importance of finding winning tickets in SCA. Regardless of the evaluated dataset, starting from a large baseline model and applying the Lottery Ticket Hypothesis improves the chances to create a small and efficient neural network model.

5 Experimental Results

5.1 Baseline Neural Networks

In our experiments, we define six different baseline models: three MLPs and three CNNs. Here, the main idea is to demonstrate how pruning and weight re-initialization (the Lottery Ticket Hypothesis) provide different SCA results if the

[3] Similarly, pruning can be considered as keeping the largest weights in model.

baseline model varies in size or capacity. The MLP models are selected based on the sizes of commonly used architectures from the related works [20,5,22]. CNN models contain relatively fewer trainable parameters, and we define them based on efficient results obtained with smaller models as presented in [35,31,22].

Table 1 lists the hyperparameter configurations for MLP4, MLP6, and MLP8 models. The main idea is to verify how pruning and re-initialization work for MLP architectures with different numbers of dense layers and, consequently, different number of trainable parameters. Note that we have not selected very large neural network models. All of them contain less than one million trainable parameters. Here, the goal is to demonstrate that even a moderately-sized model can be significantly reduced according to the Lottery Ticket Hypothesis procedure presented in Algorithm 1 and still keep or provide improved profiling SCA results. While it could be said that neural networks with up to one million trainable parameters are small, we note that the state-of-the-art results report significantly smaller architectures (even significantly fewer than 100 000 trainable parameters) [35,31,27].

The principle also holds for the chosen CNN models. Table 2 shows three CNN architectures, denoted as CNN3, CNN4, and CNN4-2. We defined relatively small CNNs (but still larger than state-of-the-art in, e.g., [35]), which are sufficient to break the evaluated datasets. CNN3 has only one convolution layer, while CNN4 and CNN4-2 contain two convolution layers each. In particular, CNN4-2 has larger dense layers than CNN4 to allow more complex relations between the input-output data pairs to be found (and allow more overfitting to happen). It is important to note that we define the same models for three different datasets. It is expected that for baseline models (without pruning), the performance might not be optimal for all cases. Although it is out of this chapter's scope to identify one model that generalizes well for all scenarios, we demonstrate that applying the Lottery Ticket Hypothesis procedure is a step forward in this important deep learning-based profiling SCA research direction.

We also provide experimental results demonstrating that the procedure described in Section 4 depends on several aspects such as the number of profiling traces and the sparsity level in the pruning process. By identifying the optimal sparsity level for pruning, we can drastically improve the performance of re-initialized sub-networks. Moreover, in some scenarios, we show that even when a large baseline model cannot recover the key, the pruned and re-initialized sub-network succeeds, especially when the number of profiling traces is small.

Interpreting Plots: This section's results are given in terms of guessing entropy for different baseline models, datasets, and sparsity levels. The sparsity level is provided in the x-axis, where we apply pruning to the trained baseline neural network from 1% up to 99%. In each plot, there is a dashed green line that represents the average resulting guessing entropy for the baseline model *without pruning*. Thus, the green line is shown together with the plots to indicate the obtained guessing entropy when baseline models are trained for 300 epochs without any pruning. We consider 300 epochs to skip possible underfitting scenarios.

Layer	**MLP4**	**MLP6**	**MLP8**
Dense_1	200 neurons	200 neurons	200 neurons
Dense_2	200 neurons	200 neurons	200 neurons
Dense_3	200 neurons	200 neurons	200 neurons
Dense_4	200 neurons	200 neurons	200 neurons
Dense_5	-	200 neurons	200 neurons
Dense_6	-	200 neurons	200 neurons
Dense_7	-	-	200 neurons
Dense_8	-	-	200 neurons
Softmax	9 neurons	9 neurons	9 neurons
Parameters (ASCAD Random Keys)	402 609	483 009	563 409
Parameters (ASCAD Fixed Key)	262 609	343 009	423 409
Parameters (CHES CTF 2018)	562 609	643 009	723 409

Table 1: MLP architectures (batch size 400, learning rate 0.001, ADAM, *selu* activation functions). Number of parameters vary for different datasets due to different input layer dimensions.

The models we consider range from 32 969 to 723 409 trainable parameters, and with 300 epochs, there are no extreme overfitting cases.

The dashed red line is the resulting average guessing entropy after the trained baseline model is pruned according to the indicated sparsity level (x-axis) and initialized with *random weights* and trained for 50 epochs. Finally, the blue line is the resulting average guessing entropy from the same previous pruned model and re-initialized with *initial weights* from the baseline model according to LTH and trained for 50 epochs. For each sparsity level, each experiment is repeated ten times. Therefore, each plot results from training $98 \times 2 \times 10 = 1\,960$ pruned models. The plots also present the margin variation obtained with ten experiments (depicted as the area in the respective color).

We briefly discuss the limits that pruning and the Lottery Ticket Hypothesis offer regarding the results and their explainability:

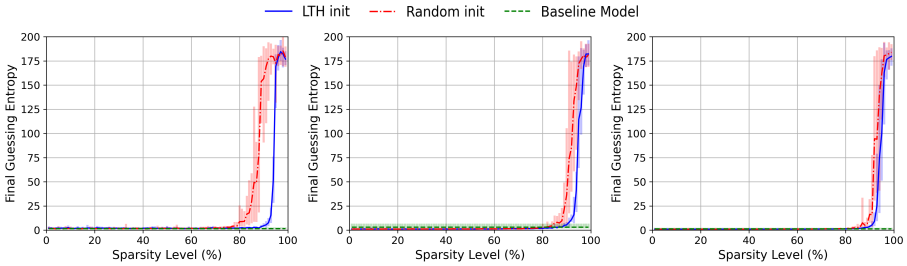
1. Pruning allows smaller neural networks that perform on the level or even better than larger ones. This results from the regularization effect provided by pruning out small weights according to some strategy (random pruning or LTH).
2. The Lottery Ticket Hypothesis assumes there will be smaller, good performing sub-networks, so-called winning tickets. Winning tickets in profiling SCA allow reaching small sub-networks with good attack performance, as measured with GE. This provides an alternative solution for hyperparameter

Layer	CNN3	CNN4	CNN4-2
Conv1D_1	16 filters ks=10, stride=5	16 filters ks=10, stride=5	16 filters ks=10, stride=5
MaxPool1D_1	ks=2, stride=2	ks=2, stride=2	ks=2, stride=2
-	BatchNorm	BatchNorm	BatchNorm
Conv1D_2	-	16 filters ks=10, stride=5	16 filters ks=10, stride=5
MaxPool1D_2	-	ks=2, stride=2	ks=2, stride=2
-	-	BatchNorm	BatchNorm
Dense_1	128 neurons	128 neurons	256 neurons
Dense_2	128 neurons	128 neurons	256 neurons
Softmax	9 neurons	9 neurons	9 neurons
Parameters (ASCAD Random Keys)	302 713	47 305	124 489
Parameters (ASCAD Fixed Key)	159 353	32 969	95 817
Parameters (CHES CTF 2018)	466 553	63 689	157 257

Table 2: CNN architectures (batch size 400, learning rate 0.001, ADAM, *selu* activation function). Number of parameters vary for different datasets due to different input layer dimensions.

tuning in which pruning is used to extract the best possible performance from a model by disabling unnecessary weight connections.

3. In profiling SCA, finding an efficient model is also characterized by determining a good balance between the model's fitting capacity (i.e., its number of trainable parameters) and its generalization. Regularization is the method that provides this balance if one chooses to avoid tuning the model's hyperparameters. However, finding good regularizers might also pose critical difficulties, especially when there are more hyperparameters to be tuned due to the regularizer choice. Therefore, the pruning, and LTH process, offer a cheap and easy-to-deploy alternative to regularize a large model. However, not all neural network sizes will necessarily be converted from a baseline model that performs poorly into an optimal one just by applying pruning strategies as regularizers. Other aspects, such as dataset nature and the number of profiling traces, also will affect the pruned model's performance.
4. Pruning and LTH are not methods to provide explainability. However, explainability and interpretability can be used to improve the pruning strategy. This can be done by analysing, e.g., gradients [19], neuron relevance to classification [34] or simply weight magnitude [10].



(a) 30 000 profiling traces. (b) 40 000 profiling traces. (c) 50 000 profiling traces.

Fig. 2: ASCAD Fixed Key, MLP4

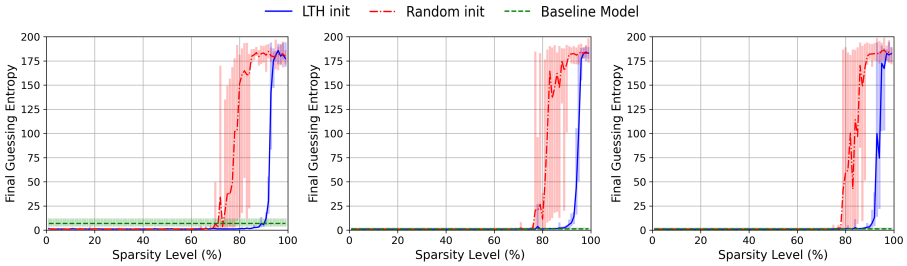
5.2 ASCAD with a Fixed Key

Figures 2, 3, and 4 provide results for the ASCAD Fixed Key dataset when MLP4, MLP6, and MLP8 are used as baseline models, respectively, for different number of profiling traces. There, we can immediately conclude that random initialization (*random init* in figure legends) and LTH initialization (*LTH init* in figure legends) provide different final guessing entropy results for different MLP sizes and the number of profiling traces. For the LTH case, the model size and the number of profiling traces have a small impact, and we can observe that, for all scenarios, pruning up to 90% of the weights show similar key recovery results.

On the other hand, if the pruned models are initialized with random weights, the model's performance is directly related to model size and the number of profiling traces. Adding more profiling traces improves the behavior of the model that is randomly initialized, approaching the model's behavior that is re-initialized according to LTH. The baseline model performs better than the pruned model that uses random initialization if the percentage of pruned weights is larger than 50% (for MLP8), and the number of profiling traces is sufficient to build a strong model. For the pruned model that follows the LTH initialization, the baseline model performs better only if we prune more than 90% of weights.

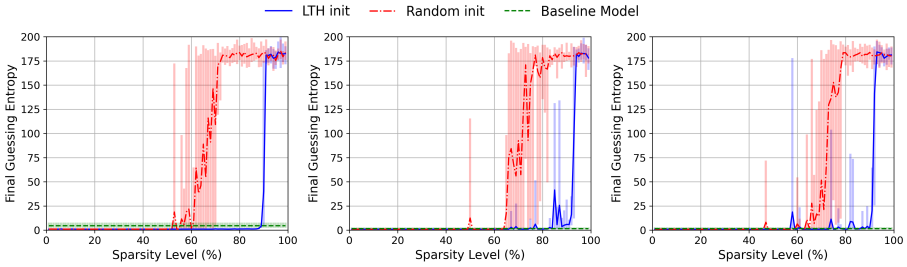
Interestingly, we can observe that pruning and LTH weight initialization show very stable results. Repeating the experiments ten times for each sparsity level tends to provide similar final guessing entropy values. Random weight initialization after pruning clearly shows different final guessing entropy results, which is an obvious consequence of the randomness of weight initialization.

Next, we give results for the three different CNN architectures. Figures 5 and 6 indicate that for 30 000 training traces, as the dataset is small, the baseline model generally performs well but shows signs of overfitting. Then, pruning up to 60% of weights improves the performance regardless of the weight initialization procedure, although the LTH approach shows more stable and superior results. Increasing the number of traces shows improved behavior for the baseline model. Still, carefully selected sub-networks are sufficient to break the target, even when pruning 80% of weights.



(a) 30 000 profiling traces. (b) 40 000 profiling traces. (c) 50 000 profiling traces.

Fig. 3: ASCAD Fixed Key, MLP6



(a) 30 000 profiling traces. (b) 40 000 profiling traces. (c) 50 000 profiling traces.

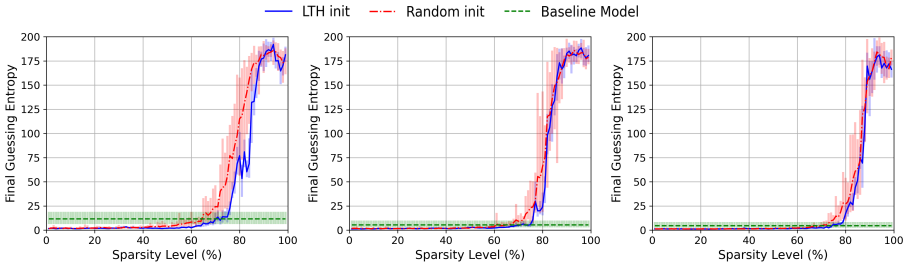
Fig. 4: ASCAD Fixed Key, MLP8

Going to a more complex architecture (CNN4), the baseline model performs well and can reach a guessing entropy of one. However, this baseline model shows more variation from the ten repeated experiments. Simultaneously, pruning enables similar performance where the larger the training set, the smaller the differences between weight initialization procedures (LTH initialization or random). In Figure 7, we consider the most complex CNN architecture. Interestingly, for 40 000 and 50 000 traces, we observe an even better performance of pruned networks when compared to the baseline model for up to 60% pruned weights. Again, LTH initialization tends to provide more stable results.

5.3 ASCAD with Random Keys

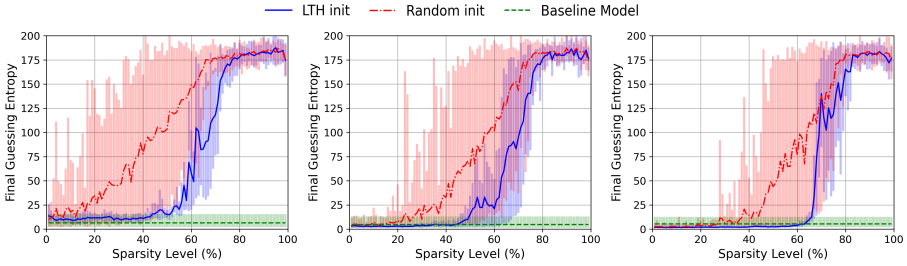
In this section, we provide results for the ASCAD Random Keys dataset, as introduced in Section 2.3. Again, we apply the LTH procedure for a different number of profiling traces (60 000, 100 000, and 200 000) on the six different baseline models (MLP4, MLP6, MLP8, CNN3, CNN4, and CNN4-2).

Figure 8 shows results for different number of profiling traces and the MLP4 baseline model. With four dense layers, this MLP can be considered a small model, which is sufficient to break the ASCAD dataset for a large number of



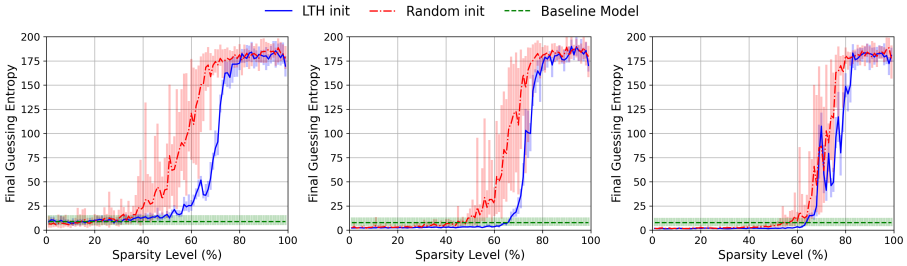
(a) 30 000 profiling traces. (b) 40 000 profiling traces. (c) 50 000 profiling traces.

Fig. 5: ASCAD Fixed Key, CNN3



(a) 30 000 profiling traces. (b) 40 000 profiling traces. (c) 50 000 profiling traces.

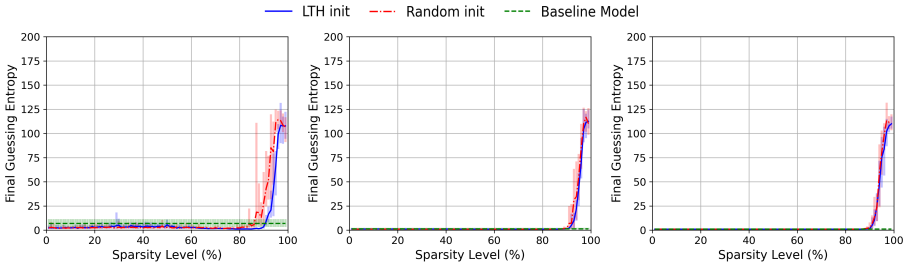
Fig. 6: ASCAD Fixed Key, CNN4



(a) 30 000 profiling traces. (b) 40 000 profiling traces. (c) 50 000 profiling traces.

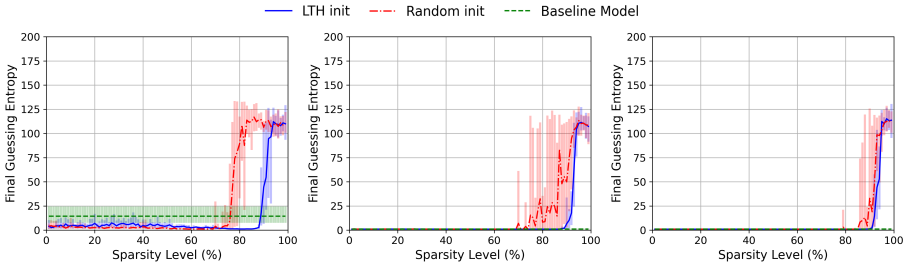
Fig. 7: ASCAD Fixed Key, CNN4-2

profiling traces (above 100 000), as indicated by the baseline model guessing entropy results. However, if the number of profiling traces is reduced (60 000), the guessing entropy result for the baseline model trained for 300 epochs is worse due to overfitting. On the other hand, applying the LTH process on this MLP4 baseline model shows good results even when the number of profiling traces is reduced. A natural alternative to fix the baseline model training would be to



(a) 60 000 profiling traces. (b) 100 000 profiling traces. (c) 200 000 profiling traces.

Fig. 8: ASCAD Random Keys, MLP4



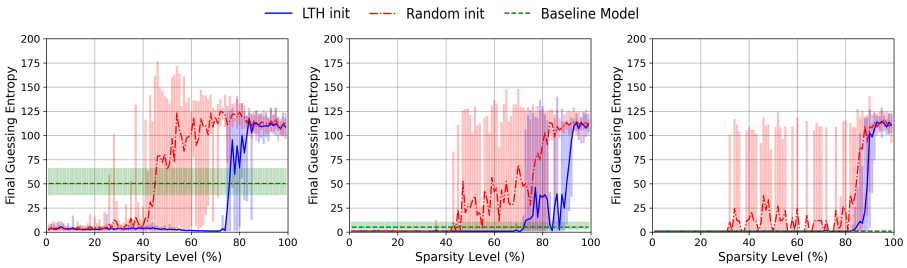
(a) 60 000 profiling traces. (b) 100 000 profiling traces. (c) 200 000 profiling traces.

Fig. 9: ASCAD Random Keys, MLP6

reduce the number of epochs to limit the overfitting. However, we expose this result (Figure 8a) to demonstrate how pruning (even from 1% of weights) already regularizes the model and delivers successful attack results (we also must mention that pruned model is trained for fewer epochs, also reducing overfitting).

The observations are confirmed in Figures 9 and 10 for MLP models with more capacity (MLP6 and MLP8). Indeed, profiling sets that are too small cause overfitting for the baseline model, which can be easily resolved following the pruning method. Notice that random initialization always works worse than LTH initialization, and it also gives more irregular behavior due to the randomness in the process. This is even more evident in Figure 10 where the variation of random initialization after pruning is very significant. This confirms that the LTH is valid in the profiling SCA context. As shown in Figure 9, pruning approximately 90% of the weights from the baseline model results in a successful attack when weights are initialized with the LTH process.

Comparing Figures 9 and 10, the larger baseline models tend to provide less successful results when the LTH procedure is applied. Larger baseline models may overfit training data more easily, and, as a consequence, the pruning process is applied to a model that might overfit. The solution for this problem is to consider early stopping for the baseline model training. This way, pruning would



(a) 60 000 profiling traces. (b) 100 000 profiling traces. (c) 200 000 profiling traces.

Fig. 10: ASCAD Random Keys, MLP8

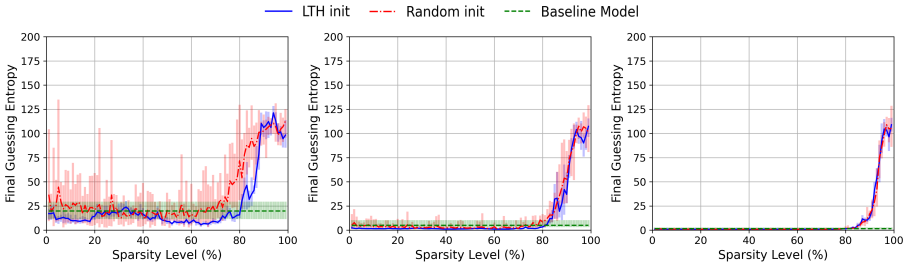
be applied to the baseline model weights when they reach the best training epoch. To confirm our hypothesis, we can consider Figure 10c. The baseline model (MLP8) is trained on 200 000 profiling traces for 300 epochs and does not overfit, as seen in the final baseline model's guessing entropy. In this case, the pruned model performance with LTH initialization is as good as for smaller baseline models trained on the same number of profiling traces (see, e.g., Figure 9c).

The CNN architectures selected for this analysis show better guessing entropy results for the baseline model when more profiling traces are used, as shown in Figures 11, 12, and 13. However, when less profiling traces are used, as is the case of results provided in Figures 11b, 12b, and 13b, the baseline guessing entropy is not reaching one on average. Adding more profiling traces helps, but the number of profiling traces should align with the model complexity. The evaluated CNN models worked well for the ASCAD Fixed Key dataset, as shown in the last section. However, these models (especially CNN4 and CNN4-2) appear less appropriate for the ASCAD Random Keys dataset. In such cases, pruning plays an important role in (partially) overcoming this. After pruning, it is possible to reach very low GE values (under 5) for a specific percentage of pruned weights. In particular, results show that pruning plus LTH initialization is better than pruning plus random initialization. For all cases, we can prune up to around 50% of weights and still reach good performance even though we use (relatively) simple CNN architectures.

5.4 CHES CTF 2018

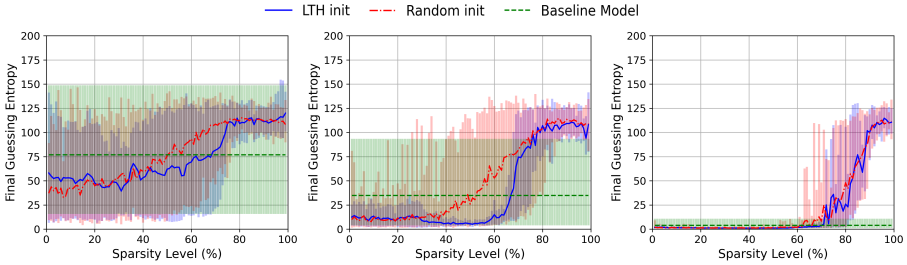
For the CHES CTF 2018 dataset, we repeated the experiments on the same neural network architectures defined in Tables 1 and 2. In this case, we observed much better results for the three selected MLPs and CNN3 than results obtained for CNN4 and CNN4-2. These results again confirm the practical advantage of the LTH procedure in profiling SCA.

Figure 14 shows the guessing entropy for different sparsity levels on three different number of profiling traces: 20 000, 30 000, and 40 000. As indicated by the dashed green line in Figures 14a, 14b, and 14c, the baseline guessing entropy



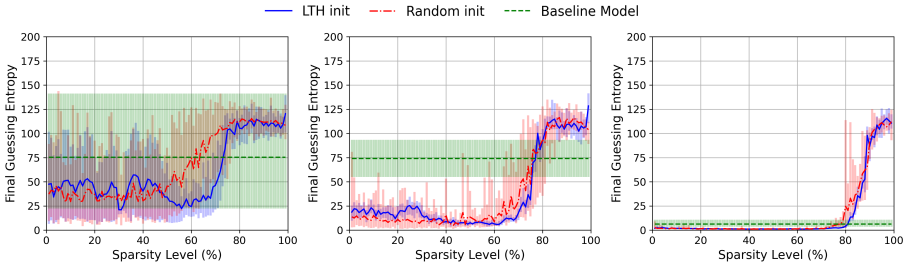
(a) 60 000 profiling traces. (b) 100 000 profiling traces. (c) 200 000 profiling traces.

Fig. 11: ASCAD Random Keys, CNN3



(a) 60 000 profiling traces. (b) 100 000 profiling traces. (c) 200 000 profiling traces.

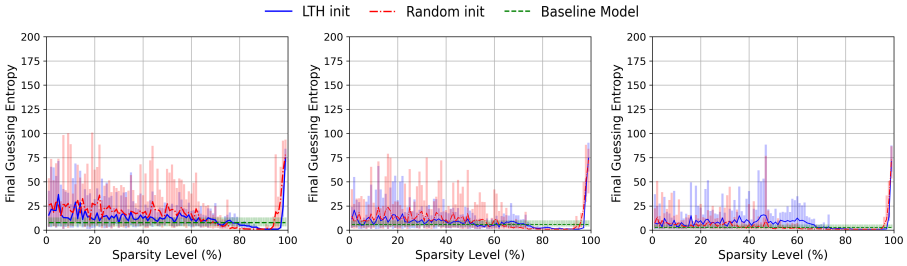
Fig. 12: ASCAD Random Keys, CNN4



(a) 60 000 profiling traces. (b) 100 000 profiling traces. (c) 200 000 profiling traces.

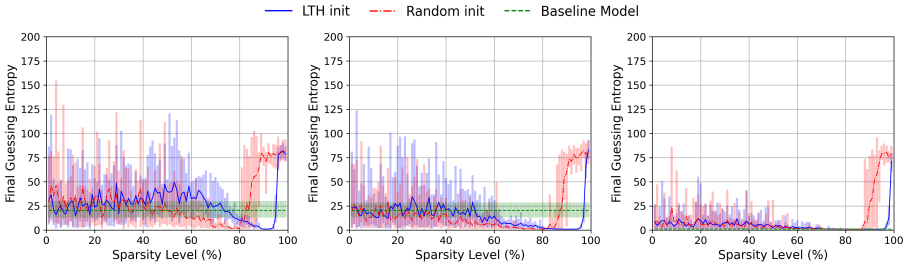
Fig. 13: ASCAD Random Keys, CNN4-2

cannot reach one for MLP4 trained on 300 epochs. Adding more profiling traces helps, but still, GE stays slightly above one on average. When the network is pruned, we can immediately see how GE improves, especially for sparsity levels around 80% to 95%. The LTH initialization shows better (at least more stable) results than random initialization. Figures 15 and 16 confirm our observations as more profiling traces is required for good attack performance for the baseline



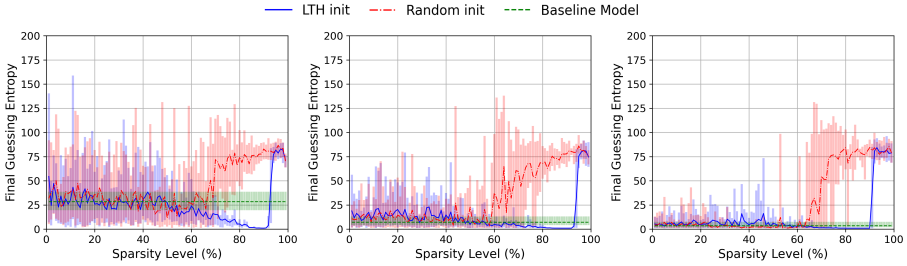
(a) 20 000 profiling traces. (b) 30 000 profiling traces. (c) 40 000 profiling traces.

Fig. 14: CHES CTF 2018, MLP4



(a) 20 000 profiling traces. (b) 30 000 profiling traces. (c) 40 000 profiling traces.

Fig. 15: CHES CTF 2018, MLP6

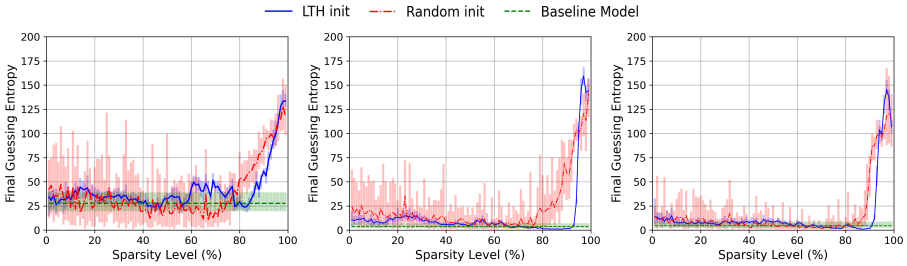


(a) 20 000 profiling traces. (b) 30 000 profiling traces. (c) 40 000 profiling traces.

Fig. 16: CHES CTF 2018, MLP8

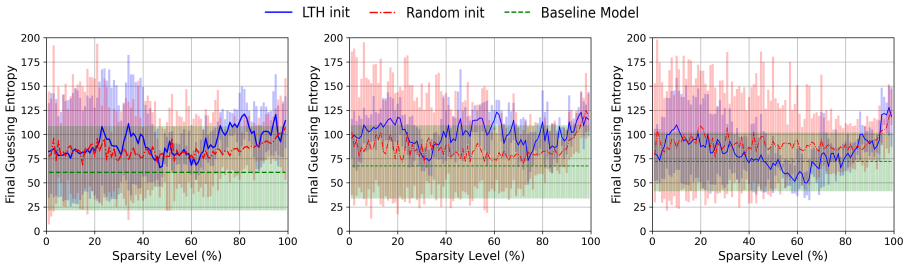
model, especially as the architecture becomes more complex. On the other hand, we can prune up to 95% of weights if we follow the LTH initialization and still reach superior attack performance.

Results for CNNs on the CHES CTF 2018 dataset are acceptable (i.e., converging to GE close to one) for the CNN3 architecture only, as shown in Figure 17. There, we see the benefit of adding more profiling traces as the baseline



(a) 20 000 profiling traces. (b) 30 000 profiling traces. (c) 40 000 profiling traces.

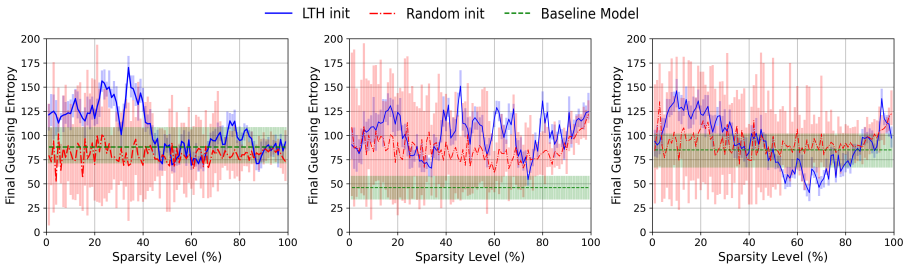
Fig. 17: CHES CTF 2018, CNN3



(a) 20 000 profiling traces. (b) 30 000 profiling traces. (c) 40 000 profiling traces.

Fig. 18: CHES CTF 2018, CNN4

model overfits. Still, some sub-networks are providing better attack performance. For CNN4 and CNN4-2 (Figures 18 and 19), the baseline model provides poor performances when trained on 300 epochs. We postulate this happens as the baseline model has a significantly larger capacity than needed, so it either overfits or underfits, becoming similar to random guessing. In other words, CNN4 and CNN4-2 on smaller profiling sets (lower than 30 000 traces) show no generalization for the baseline model, indicating that these two models are not compatible with the target dataset. We can observe how the LTH procedure reduces guessing entropy for specific sparsity level ranges even with those models. Observing Figures 18 and 19, for sparsity levels around 70%, LTH initialization reach significantly lower guessing entropy values (GE $\leq$ 70) after training for 50 epochs. Increasing the number of attack traces (we consider only 2 000 attack traces) could lead to successful key recovery, which is particularly interesting if a baseline model provided performance close to random guessing. When the number of profiling traces is increased to 40 000 traces (Figures 18c and 19c), the baseline model shows slightly better results and the LTH initialization still improves the attack performance. In this case, we can verify that random initialization might not be a good procedure, as the guessing entropy results are inferior to the baseline model results.



(a) 20 000 profiling traces. (b) 30 000 profiling traces. (c) 40 000 profiling traces.

Fig. 19: CHES CTF 2018, CNN4-2

5.5 General Observations

Based on the conducted experiments, we provide several general observations:

- If the baseline model works poorly for a limited set of attack traces, pruning might still improve performance.
- If the baseline works well and does not overfit, then pruning maintains the performance but produces smaller and regularized networks.
- If there are not enough profiling traces for the model capacity, it will overfit, and pruning can help avoid that.
- More profiling traces improve pruning results, but it also reduces differences between weight initialization techniques.
- Pruning and LTH initialization procedure works the best, provided the neural network architectures are large enough to utilize the winning tickets.
- Pruning can improve the attack results as indicated by the SCA performance metrics.

6 Conclusions and Future Work

This chapter discussed how pruning could improve the attack performance for deep learning-based side-channel analysis. We considered the recently proposed Lottery Ticket Hypothesis that assumes there are small sub-networks in the original network that perform on the same level as the original network. To the best of our knowledge, both of those concepts were never before investigated in profiling SCA. Our experimental investigation confirms this hypothesis for profiling SCA, which allows us to prune up to 90% of weights and still reach good attack performance. Thus, we manage to reach the same attack performance for significantly smaller networks (easier to tune and faster to train). What is more, we show how pruning helps when a large network overfits or has issues due to imbalanced data. In such cases, pruning, besides resulting in smaller architectures, enables improved attack performance.

As future work, we plan to consider more sophisticated pruning techniques and different leakage models. Finally, as discussed, pruning allows smaller neural networks and good performance but does not provide insights into neural

networks' explainability. It could be interesting to consider various feature visualization techniques to evaluate the important features before and after the pruning. Also, explainability and interpretability techniques could be efficiently applied here to select weights to be pruned.

Acknowledgements

This work was supported in part by the Netherlands Organization for Scientific Research NWO project DISTANT (CS.019) and project PROACT (NWA.1215.18.014).

References

1. ASCAD GitHub Repository. Website (2018), https://github.com/ANSSI-FR/ASCAD
2. CHES CTF 2018. Website (2018), https://chesctf.riscure.com/2018/news
3. Preprocessed CHES CTF 2018 dataset. Website (2021), http://aisylabdatasets.ewi.tudelft.nl/
4. Archambeau, C., Peeters, E., Standaert, F.X., Quisquater, J.J.: Template attacks in principal subspaces. In: Goubin, L., Matsui, M. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2006. pp. 1–14. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
5. Benadjila, R., Prouff, E., Strullu, R., Cagli, E., Dumas, C.: Deep learning for side-channel analysis and introduction to ASCAD database. J. Cryptographic Engineering **10**(2), 163–188 (2020). https://doi.org/10.1007/s13389-019-00220-8, https://doi.org/10.1007/s13389-019-00220-8
6. Blalock, D., Ortiz, J.J.G., Frankle, J., Guttag, J.: What is the state of neural network pruning? (2020)
7. Cagli, E., Dumas, C., Prouff, E.: Convolutional neural networks with data augmentation against jitter-based countermeasures. In: Fischer, W., Homma, N. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2017. pp. 45–68. Springer International Publishing, Cham (2017)
8. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Jr., B.S.K., Koç, Ç.K., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers. Lecture Notes in Computer Science, vol. 2523, pp. 13–28. Springer (2002). https://doi.org/10.1007/3-540-36400-5_3, https://doi.org/10.1007/3-540-36400-5_3
9. Choudary, O., Kuhn, M.G.: Efficient template attacks. In: Francillon, A., Rohatgi, P. (eds.) Smart Card Research and Advanced Applications. pp. 253–270. Springer International Publishing, Cham (2014)
10. Frankle, J., Carbin, M.: The lottery ticket hypothesis: Training pruned neural networks. CoRR **abs/1803.03635** (2018), http://arxiv.org/abs/1803.03635
11. Gilmore, R., Hanley, N., O'Neill, M.: Neural network based attack on a masked implementation of AES. In: 2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). pp. 106–111 (May 2015). https://doi.org/10.1109/HST.2015.7140247

12. Heuser, A., Picek, S., Guilley, S., Mentens, N.: Side-channel analysis of lightweight ciphers: Does lightweight equal easy? In: Hancke, G.P., Markantonakis, K. (eds.) Radio Frequency Identification and IoT Security - 12th International Workshop, RFIDSec 2016, Hong Kong, China, November 30 - December 2, 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 10155, pp. 91–104. Springer (2016). https://doi.org/10.1007/978-3-319-62024-4_7, https://doi.org/10.1007/978-3-319-62024-4_7
13. Heuser, A., Zohner, M.: Intelligent Machine Homicide - Breaking Cryptographic Devices Using Support Vector Machines. In: Schindler, W., Huss, S.A. (eds.) COSADE. LNCS, vol. 7275, pp. 249–264. Springer (2012)
14. Janowsky, S.A.: Pruning versus clipping in neural networks. Phys. Rev. A **39**, 6600–6603 (Jun 1989). https://doi.org/10.1103/PhysRevA.39.6600, https://link.aps.org/doi/10.1103/PhysRevA.39.6600
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.C.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: Proceedings of CRYPTO'96. LNCS, vol. 1109, pp. 104–113. Springer-Verlag (1996)
17. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology. pp. 388–397. CRYPTO '99, Springer-Verlag, London, UK, UK (1999), http://dl.acm.org/citation.cfm?id=646764.703989
18. Lerman, L., Poussier, R., Bontempi, G., Markowitch, O., Standaert, F.X.: Template attacks vs. machine learning revisited (and the curse of dimensionality in side-channel analysis). In: International Workshop on Constructive Side-Channel Analysis and Secure Design. pp. 20–33. Springer (2015)
19. Liu, C., Wu, H.: Channel pruning based on mean gradient for accelerating convolutional neural networks. Signal Processing **156**, 84–91 (10 2018). https://doi.org/10.1016/j.sigpro.2018.10.019
20. Maghrebi, H., Portigliatti, T., Prouff, E.: Breaking cryptographic implementations using deep learning techniques. In: International Conference on Security, Privacy, and Applied Cryptography Engineering. pp. 3–26. Springer (2016)
21. Martinasek, Z., Hajny, J., Malina, L.: Optimization of power analysis using neural network. In: Francillon, A., Rohatgi, P. (eds.) Smart Card Research and Advanced Applications. pp. 94–107. Springer International Publishing, Cham (2014)
22. Perin, G., Chmielewski, L., 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 **2020**(4), 337–364 (Aug 2020). https://doi.org/10.13154/tches.v2020.i4.337-364, https://tches.iacr.org/index.php/TCHES/article/view/8686
23. Picek, S., Heuser, A., Jovic, A., Batina, L.: A systematic evaluation of profiling through focused feature selection. IEEE Transactions on Very Large Scale Integration (VLSI) Systems **27**(12), 2802–2815 (2019)
24. 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), 209–237 (Nov 2018). https://doi.org/10.13154/tches.v2019.i1.209-237, https://tches.iacr.org/index.php/TCHES/article/view/7339

25. Picek, S., Heuser, A., Jovic, A., Ludwig, S.A., Guilley, S., Jakobovic, D., Mentens, N.: Side-channel analysis and machine learning: A practical perspective. In: 2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017. pp. 4095–4102 (2017)
26. Quisquater, J.J., Samyde, D.: Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In: Attali, I., Jensen, T. (eds.) Smart Card Programming and Security. pp. 200–210. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
27. Rijsdijk, J., Wu, L., Perin, G., Picek, S.: Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2021**(3), 677–707 (2021). https://doi.org/10.46586/tches.v2021.i3.677-707, https://doi.org/10.46586/tches.v2021.i3.677-707
28. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2005. pp. 30–46. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
29. Standaert, F.X., Malkin, T.G., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Joux, A. (ed.) Advances in Cryptology - EUROCRYPT 2009. pp. 443–461. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
30. van der Valk, D., Krcek, M., Picek, S., Bhasin, S.: Learning from a big brother - mimicking neural networks in profiled side-channel analysis. In: 2020 57th ACM/IEEE Design Automation Conference (DAC). pp. 1–6 (2020). https://doi.org/10.1109/DAC18072.2020.9218520
31. Wouters, L., Arribas, V., Gierlichs, B., Preneel, B.: Revisiting a methodology for efficient cnn architectures in profiling attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems **2020**(3), 147–168 (Jun 2020). https://doi.org/10.13154/tches.v2020.i3.147-168, https://tches.iacr.org/index.php/TCHES/article/view/8586
32. Wu, L., Perin, G., Picek, S.: I choose you: Automated hyperparameter tuning for deep learning-based side-channel analysis. Cryptology ePrint Archive, Report 2020/1293 (2020), https://eprint.iacr.org/2020/1293
33. Yang, S., Zhou, Y., Liu, J., Chen, D.: Back propagation neural network based leakage characterization for practical security analysis of cryptographic implementations. In: Kim, H. (ed.) Information Security and Cryptology - ICISC 2011. pp. 169–185. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
34. Yeom, S., Seegerer, P., Lapuschkin, S., Wiedemann, S., Müller, K., Samek, W.: Pruning by explaining: A novel criterion for deep neural network pruning. CoRR **abs/1912.08881** (2019), http://arxiv.org/abs/1912.08881
35. Zaid, G., Bossuet, L., Habrard, A., Venelli, A.: Methodology for efficient cnn architectures in profiling attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems **2020**(1), 1–36 (Nov 2019). https://doi.org/10.13154/tches.v2020.i1.1-36, https://tches.iacr.org/index.php/TCHES/article/view/8391