Noname manuscript No.
(will be inserted by the editor)

Regularizers to the Rescue: Fighting Overfitting in Deep Learning-based Side-channel Analysis

Azade Rezaeezade · Lejla Batina

Received: date / Accepted: date

Abstract Despite considerable achievements of deep learning-based side-channel analysis, overfitting represents a significant obstacle in finding optimized neural network models. This issue is not unique to the side-channel domain. Regularization techniques are popular solutions to overfitting and have long been used in various domains. At the same time, the works in the side-channel domain show sporadic utilization of regularization techniques. What is more, no systematic study investigates these techniques' effectiveness. In this paper, we aim to investigate the regularization effectiveness by applying four powerful and easy-to-use regularization techniques to six combinations of datasets, leakage models, and deep-learning topologies. The investigated techniques are L_1, L_2, **dropout**, and **early stopping**. Our results show that while all these techniques can improve performance in many cases, L_1 and L_2 are the most effective. Finally, if training time matters, early stopping is the best technique to choose.

Keywords Side-channel Analysis, Deep Learning, Regularization, Overfitting

1 Introduction

Embedded security products like smart cards and IoT devices are used daily. To protect confidential and private information they contain, manufacturers utilize cryptographic solutions. After implementing a cryptographic algorithm (in hardware or software), there is often a dependency between the secret data processed during cryptographic operation and the side-channel measurements such as the power consumption [15] or electromagnetic emanation [28] from the chip running the implementations. An attacker can thus find the secret data by exploiting this dependency and using the techniques introduced as side-channel analysis (SCA) [19]. Implementing countermeasures can decrease this dependency. However, even after using countermeasures, there can still remain some leaks in implementations. As a result, an attacker can still perform SCA by applying more powerful techniques. Evaluators conduct side-channel analysis to ensure that despite the mentioned dependency, the products might still resist known side-channel attacks in a way that if a certain attack is possible, then the attacker needs extensive resources to succeed [25].

SCA is typically divided into non-profiling and profiling methods [5]. In the non-profiling methods, the adversary has access to a large number of side-channel measurements from the device under attack and analyzes these measurements using statistical techniques. Simple and differential power analysis [15], correlation power analysis [3] and mutual information analysis [8] are examples of non-profiling SCA. On the other hand, profiling methods use a clone device to create a profile, then use this profile to analyze the device under attack. Template attack [5] and machine learning-based attacks [11, 17, 24] are examples of profiling SCA.

Deep learning-based side-channel analysis (DL-SCA) is a powerful approach to profiling SCA. Multilayer perceptron (MLP) and Convolutional Neural Networks (CNNs) are two deep learning architectures that have been used widely in DL-SCA. CNN and MLP are capa-

Azade Rezaeezade
Cyber Security Research Group, Delft University of Technology, Mekelweg 2, Delft, The Netherlands
E-mail: a.rezaeezade-1@tudelft.nl

Lejla Batina
Digital Security Group, Radboud University,
Postbus 9010, Nijmegen, The Netherlands
E-mail: lejla@cs.ru.nl

ble of learning even higher-order leakages derived from masking-protected implementations. They can also handle the features pre-processing (finding Points of Interest) implicitly [4,14]. However, using deep learning has certain bottlenecks. One of the well-known problems is overfitting. Overfitting happens when a deep model fits the training measurements perfectly but cannot generalize to previously unseen measurements. Hence, an overfitted model cannot evaluate the product correctly.

Regularization techniques are used during the training to decrease the complexity of the model on the fly and prevent overfitting. So far, many regularization techniques have been introduced. However, while there is a strong recommendation for using regularization techniques in DL-SCA from previous research [26], no comprehensive study shows a practical consequence of using them in DL-SCA. Indeed, while most of the previous works that consider regularization techniques do advocate the usage of those, they do not rely on systematic evaluation or comparison.

This work aims to fill this gap and show how the most commonly used regularization techniques, i.e., L_1, L_2, dropout, and early stopping, can improve deep learning-based SCA. The main contributions of this paper can be summarized as follows:

- We compare the influence of four popular regularization techniques (L_1, L_2, dropout, and early stopping) in DL-SCA. For this analysis, we run hundreds of deep learning models with and without regularization techniques. To make our comparison more complete, and thorough we consider one software-based and one hardware-based dataset (ASCAD and AES_HD). Besides, we utilize two widely used deep learning models (MLP and CNN) and three different leakage models (Hamming Weight, Hamming Distance, and Identity leakage model) depending on the considered dataset.
- We show that the improvements that many techniques in deep learning offer heavily depend on the model's characteristics (its architectural and learning hyperparameters). In many cases, the combination of the hyperparameters and the used technique decreases the performance (while one would expect improvement).
- We introduce the deterioration rate to show how reliable a specific regularization technique is. This metric shows the probability that a model worsens after applying a regularization technique. We also consider profiling time as another metric providing insights into the performance of various techniques.
- We consider the baseline model's performance and its relation to the improvements that the regularized model can offer. We show that when the implicit regularization of the baseline model is high, adding a regularization technique worsens it. In contrast, applying regularization techniques improves the performance when the baseline model's implicit regularization is low.

2 Background

2.1 Deep Learning-based Side-channel Analysis

Profiling SCA runs in two steps. In the first step (profiling phase), a profile is built using the measurements collected from a clone device. In the second step (attack phase), the adversary uses the profile to find the key used on the target device. The assumption is that the measurements collected from clone and target devices follow the same distribution. Profiling and attack steps match the training and test phases of machine/deep learning. [1]

DL-SCA is defined as a classification problem. The profiling set $\mathcal{X}$ contains N_a measurements collected while multiple cryptographic operations were performed on N_p plaintexts and key pairs. Depending on the chosen leakage model, the measurements' labels are calculated using a combination of their corresponding plaintexts and keys. The goal is to find the underlying distribution of the profiling data. The deep learning model learns this distribution by tuning its parameters (the neurons' weights and biases) using the profiling data.

In the attack phase, the learned distribution is used to classify N_a measurements from the target device. For each key candidate, an S_k score is calculated as $S_k = \sum_{i=1}^{N_a} \log p(x_i, c_j)$, where N_a is the number of measurements in the attack set and $p(x_i, c_j)$ indicates the probability that a measurement x_i belongs to class c_j. The final step is sorting the keys based on the calculated scores. The key with the biggest score is considered the most probable key used for cryptographic operation on the target device.

The guessing entropy (GE) and required number of attack traces (NT) metrics can be defined using this score vector. Let us suppose that the correct key ($k^\star$) used during cryptographic operation is placed in the r^{th} place. This place is called the **rank** of the $k^\star$ among all the possible keys. Guessing Entropy is defined as the average rank of $k^\star$ over a number of experiments. The second metric, the required number of attack traces, is defined as the average minimum number of measurements the model needs to place $k^\star$ in the first place (to reach $GE = 1$) [29].

[1] Our focus here is on deep learning. However, all the descriptions are also accurate for machine learning.

2.2 Regularization Techniques

The primary goal of training a deep learning model is to prepare it to predict unseen data accurately. This goal is interpreted as improving the generalization power of a deep neural network. The challenge is that the model is learning from the training data, so it tries to increase the accuracy (or decrease the loss) in training set by fitting the training examples and reducing the training error. Nevertheless, the final goal is to use the model for a test set and decrease the generalization error (test error). Thus, an appropriate learning algorithm 1) should make the training error small and 2) should make the gap between training and test error minimal [9]. Underfitting occurs when a deep learning model does not reach a sufficiently small error value in the training set. Overfitting happens when a model cannot reduce the gap between training and test error. While both underfitting and overfitting should be avoided, the latter is more challenging to control. Especially in deep learning, where much input training data and many input features are available, the best strategy is to fix underfitting by choosing more complex models [7].

Regularization is a well-studied and widely used solution to improve the generalization power of machine learning and deep neural models. In general, **regularization** is defined as "any modification we make to a learning algorithm tending to reduce the test error but not the training error." [9] This definition covers many techniques, from adding penalties to the objective function to multi-task learning and ensemble methods. The shared part of all these techniques is reducing overfitting and improving generalization. This paper inspects the influence of applying four regularization techniques on DL-SCA. Those techniques are: L_1 and L_2 norm penalties, dropout, and early stopping, which have all been used for a long time in machine learning and now in deep learning.

2.2.1 L_2 and L_1 Norm Penalties

L_2 parameter norm penalty or weight decay adds a penalty term in the form of all the model's squared weights to the objective function. The simplest form of formulating L_2 regularization is shown in Eq. (1):

$$\tilde{\mathrm{E}}(\mathbf{W};\mathbf{X},y) = \mathrm{E}(\mathbf{W};\mathbf{X},y) + \frac{1}{2}\lambda\sum_{ij}\mathbf{w}_{\mathrm{ij}}^2, \tag{1}$$

where E is an arbitrary objective function measuring the training error. $\mathbf{X}$ are the training examples and y are their corresponding labels. $\mathbf{W}$ is the current weights matrix, and λ is a parameter governing how strongly large weights are penalized. When using L_2 regularization, λ is a hyperparameter that should be tuned. $\tilde{\mathrm{E}}$ is the modified objective function. Considering Eq. (1), when updating weights using gradient descent, a constant term in the form of $\lambda\mathbf{W}$ subtracts from updated weights in each step. This term controls the growth of the weights and suppresses irrelevant components of the weight vector [16].

L_1 Norm Penalties

L_1 regularization is another way to penalize the size of the models (number of parameters). The difference between L_1 and L_2 regularization stems from the penalty term added to the objective function. One can see the modified objective function after using L_1 in Eq. (2):

$$\tilde{E}(\mathbf{W};\mathbf{X},y) = E(\mathbf{W};\mathbf{X},y) + \frac{1}{2}\lambda\sum_{ij}\left|\mathbf{w}_{ij}\right|, \tag{2}$$

where $\left|\mathbf{w}_{ij}\right|$ is the sum of all the model's absolute values of weights. Adding this term to the objective function shows itself as $\lambda sign(\mathbf{W})$ term in the weight update operation using gradient descent. In practice, L_1 can be seen as a built-in feature selection mechanism because it tends to shrink some weights toward zero. Using L_1 adds λ to the hyperparameters that should be tuned.

2.2.2 Dropout

The idea of this technique is to temporarily remove random neural units along with their connections from the primary network during the training. Each unit may be removed with a q probability called the dropout rate. The dropout rate is a hyperparameter and should be tuned. Dropout, in essence, is training multiple smaller networks selected randomly from the bigger primary neural network by removing some neural units in each training step. Since these smaller networks share the primary neural network's weights, the averaging at test time is as simple as using the primary network without any dropout [31].

Since regular dropout can not prevent overfitting in CNN, we used spatial dropout [32] for combinations with CNN topology. In CNN models, when a filter applies to an input vector (or matrix in the case of 2D convolution), it extracts a vector (or matrix), of which its elements are highly correlated. Dropping just one of these correlated elements may not prevent overfitting instead we can drop the whole **feature map**. Applying each filter to the convolutional layer input creates a feature map. If a convolution layer has m filters, the outcome of the layer is a set with m feature maps. Spatial dropout drops one or more entire extracted feature maps selected randomly among these m feature maps.

2.2.3 Early Stopping

In the general definition of early stopping, when training a neural network with sufficient capacity for and enough epochs, the training error keeps decreasing, while the validation error starts to rise again after a while. This rise is a sign that overfitting starts. With early stopping, we can stop training as soon as validation error starts to rise. However, the drop and rise of the validation error are not very smooth in reality. The validation error curve shows many ups and downs, and finding the exact point where the validation error starts to increase is not possible. One solution is to stop training if the generalization does not improve after a specific number of epochs. This specific number of epochs is called "patience".

3 Related Work

The competitive performance of the DL-SCA methods compared to more traditional SCA methods has attracted much attention in recent years. Many researchers utilized different notions and techniques used in deep learning like reinforcement learning [29], wight visualization [33], and information bottleneck [21], to improve the performance of DL-SCA even further. Regularizers have been used from the early days of machine learning emergence, and their practicality was noticed in various domains. DL-SCA is not an exception. For example, in [21], Perin et al. proposed a technique based on the information bottleneck to monitor the training evolution. Using this technique, they could find the best epoch to stop the training. In [30], Robissout et al. introduced a metric to evaluate the performance of a deep neural network in the side-channel domain during the profiling phase. They used this metric as a monitoring metric for early stopping. In [14], Kim et al. used several regularization techniques, including dropout, L_2, and data augmentation. In [22], Perin et al. used the ensemble method to improve the model's generalization and reduce overfitting. The ensemble method is a regularization technique combining multiple classifiers to form a better hypothesis. This technique reduces the final model's dependencies on the structural hyperparameters. Perin et al. showed that ensembles of many non-optimal models could even perform better than the best-obtained model. Batch normalization is another regularization technique [12] used in many DL-SCA works, e.g., [20, 27]. We note that while a number of related works used some form of regularization they do not compare the results with and without the regularization. Hence, it is difficult to assess how well those methods performed or whether they were even necessary.

4 Experimental Setup

4.1 Neural Network Topologies

Multilayer Perceptron (MLP) is a simple feed-forward neural network consisting of input, output, and one or more hidden layers. The input layer takes training examples as input and sends them through a fully connected graph of hidden layers to the output layer. The output layer represents output classes in classification problems. MLP approximates the underlying distribution of training examples by updating the network's weights using gradient descent and the backpropagation algorithm.

Convolutional Neural Network (CNN) is another type of feed-forward neural networks. It has one or more convolutional layers after the input layer. The layer highlights the most important features using kernels that are vectors of coefficients (or matrices of coefficients in case of 2D convolution) updated using backpropagation. After each convolution layer, there is an activation function, and then there can be a max/average-pooling layer. The network extracts the most important features using pooling layers. After these layers, there is usually one or more fully connected layers.

4.2 Datasets and Leakage Models

In our experiments, we consider two publicly available datasets: ASCAD random key, which contains measurements from a software implementation of AES-128, and AES_HD, which includes measurements from a hardware implementation of AES-128. For software-based implementation, the leakage can be modeled with Hamming Weight (HW) and Identity (ID) leakage models. In contrast, for hardware-based implementation, Hamming Distance (HD) models the leakage more precisely.

ASCAD Random Keys [2]: This dataset was provided using an assembly implementation of AES-128 published by ANSSI. The implementation is protected with Boolean masking. To collect the measurements, while the cryptographic operation was running on an ATMega8515 MCU target [1], the side-channel measurements were collected by measuring the target's electromagnetic emanations (EM). Each measurement has 250 000 time samples, which were finally reduced to a window containing 1 400 most-leaky time samples. The

[2] `https://github.com/ANSSI-FR/ASCAD/tree/master/ATMEGA_AES_v1/ATM_AES_v1_variable_key`

cryptographic operation was repeated 200 000 times with varying plaintext and keys to collect the profiling set examples, then repeated another 100 000 times with varying plaintext and a fixed key to collect the attack set examples. Details about the cryptographic design are provided in [27]. Since the first and second bytes are masked with zero (so they can be broken with first-order SCA), we attack the third byte using $Y^{(i)} = Sbox[P_3^{(i)} \oplus k_3^{(i)}]$ for the ID leakage model and $HW(Y^{(i)} = Sbox[P_3^{(i)} \oplus k_3^{(i)}])$ for the HW leakage model.

AES_HD[3]**:** This dataset is collected using an FPGA implementation of AES-128 on Xilinx Virtex-5. The implementation is unprotected. The side-channel measurements are the target's electromagnetic emanations (EM), which are represented by 1 250 time samples. In total, 500 000 traces were captured when the target encrypted 500 000 randomly generated plaintexts with a fixed key. From these 500 000 measurements, we select the first 450 000 as profiling examples and the last 50 000 as attack examples. We attacked the last round's $Sbox^{-1}$ output overwriting in a register that contains the previous inverse ShiftRows operation value. The leakage is modeled as $Y^{(i)} = HW(Sbox^{-1}[C_j^{(i)} \oplus k_j^{(i)}] \oplus C_{j'}^{(i)})$, where $C_j^{(i)}$ and $C_{j'}^{(i)}$ are two ciphertext bytes related according to the inverse ShiftRows operation, and $k_j^{(i)}$ is the corresponding round key byte. In our experiments, $j = 10$ and $j' = 6$.

4.3 Analysis Methodology

This paper aims to inspect the impact of applying L_1, L_2, dropout, and early stopping on the performance of DL-SCA. To interpret the results, we compare the average performance of a large number of neural networks that use a regularization technique with their average performance without any regularization technique. The following steps outline the methodology we use to compare the performance with and without regularization techniques:

– **Acquiring baseline models:** In this step, we start by generating 500 neural networks with the random search. We consider a specific combination of the dataset, neural network topology, and leakage model for the search. The ranges used for searching hyperparameters of MLP and CNN are listed in Table 1. These ranges are chosen based on the ranges reported in the previous works [23,27,33]. Since those 500 neural networks are generated randomly, many of them cannot decrease GE. Then, we select the 200 best as the "baseline models".

[3] `https://github.com/AISyLab/AES_HD_Ext`

It is worth mentioning that in a fixed number of randomly generated neural networks, the number of MLP models that reached $GE = 1$ was significantly larger than CNN. For example, among 500 randomly generated MLPs with the HW leakage model in the ASCAD dataset, 172 models reached $GE = 1$, and the rest were able to decrease the GE to small numbers (less than 10). In the case of CNNs with the HW leakage model, only 70 models reached $GE = 1$, and many models could not even decrease the GE lower than a random guess. However, at the same time, the best CNN models could converge to $GE = 1$ with far fewer attack traces than MLP models. In the previous example, the five best MLP models ranked the key in first place with 550 attack traces on average, while this metric was around 50 for the five best CNN models. We refer to this observation as the "general ability of MLP models to find the key" and the "potential ability of CNN models to find the key". These names are selected according to this practical experience that MLP models are less dependent on their hyperparameters to find the correct key. In contrast, CNN models are sensitive to slight hyperparameter changes.

Table 1: Searched range of MLP and CNN hyperparameters. For both MLP and CNN dense layers, we used the ranges shown in *Dense layers* part of Table 1.

Hyperparameters	Range
	Dense layers
Number of neurons	[10, 90], step = 10 + [100, 500], step = 100
Number of layers	[1, 8], step = 1
	Convolution layers
Number of layers	[1, 4], step = 1
Number of kernels	[4, 20], step = 1
First layer's filter size	[2, 4, 8, 12, 16]
$i^(th)$ layer filter size	$((i-1)filter\_size)^2$
Pooling	"Average", "Max"
Pooling size	[2, 10], step = 2
Pooling stride	[2, 10], step = 2
	Learning hyperparameters
Optimizer	"Adam", "RMSprop"
Weight initialization	"random_uniform", "glorot_uniform", "he_uniform"
Activation function	"relu", "selu", "elu"
Batch size	$[100, 900]$, $step = 100$
Learning rate	$[0.005, 0.001, 0.0005, 0.0001, 0.00005, 0.00001]$
epochs	200

– **The average performance of baseline models:** We use two metrics to represent the average performance of baseline models. The average GE that 200 baseline models can reach in an attack set with 5 000 attack traces is called "AVERAGE_GE". The average required number of attack traces the baseline models need to reach $GE = 1$ is referred to as "AVERAGE_NT". For those neural networks that cannot reach $GE = 1$ with 5 000 attack traces, we assume $NT = 5\,000$.

– **Re-training regularized models:** We re-train the baseline neural networks in the presence of different regularization techniques (L_1, L_2, dropout, and early stopping). These regularization techniques have their own hyperparameters, which should be tuned for each neural network. To do this tuning, we examine a range for each regularization hyperparameter. Then we consider the best-acquired performance as the performance of that baseline model in the presence of a specific regularization technique. Again, the considered metrics are GE and NT. The tuned baseline model with the best performance is called the "regularized model."
  An example can clarify the process: we want to re-train each baseline model in the presence of L_1 regularization. To do so, while all the other hyperparameters[4] of the baseline models stay the same, we add L_1 regularization to them. L_1 has a regularization constant λ that can take 12 different values listed in Table 2.[5] We build 12 models similar to the baseline model but having L_1 regularization with different λ values. Among the 12 re-trained models, we consider the model with the smallest GE as the best-tuned model and report its performance (the value of GE and its corresponding NT) as the regularized model's performance. If a neural network can reach $GE = 1$ with two or more λ values, we consider the smallest NT and $GE = 1$ as the performance of the regularized model.
– **The average performance of regularized models:** The AVERAGE_GE and the AVERAGE_NT are calculated as performance metrics for the regularized models. We compare the AVERAGE_GE and AVERAGE_NT of the baseline and the regularized models for each regularization technique. This way, the influence of each regularization technique on the performance of DL-SCA can be observed. The examined ranges for hyperparameters of different regularization techniques are listed in Table 2.

5 Experimental Results

First, we explore the modifications of the AVERAGE_NT and AVERAGE_GE for baseline and regularized models in Section 5.1. Then, we evaluate the models' deterioration rate in Section 5.2. Finally, in Section 5.3, we monitor the effect of using different regularization techniques on profiling time.

5.1 Performance Comparison

5.1.1 L_1 Regularization

L_1 regularization penalizes the size of the model parameters in a way that causes a subset of them to become zero. The effect of this is an implicit feature selection. As a result, L_1 regularization will be most effective when the input is noisy and the traces include samples that do not carry information. In other words, L_1 regularization helps the neural networks to extract the point of interest more efficiently. Figure 1 shows the AVERAGE_GE and the AVERAGE_NT for baseline and regularized models for the ASCAD dataset. Notice how regularized models always reach lower average GE than the baseline ones. The difference is especially pronounced for the ID leakage model and CNN. The differences become even more clear once considering the average number of traces to break the target. In three out of four settings, regularization allows for reducing the number of required attack traces in half. The results for the ASCAD dataset show that L_1 regularization is effective for all the combinations, but it tends to perform better with MLP models. That is because the CNN models naturally provide more effective feature selection than MLP models and the influence of feature selection is more tangible in the case of MLP.

Next, Figure 2 demonstrates the AVERAGE_GE and the AVERAGE_NT for baseline and regularized models for the AES_HD dataset. For this dataset, the baseline models perform well on average but require many traces to break the target. Adding L_1 reduces the average GE up to five times on average and the required number of traces to break the target for $\approx 40\%$. We believe we do not see an even larger influence on the number of attack traces due to a limited attack set size. Indeed, when a model fails to find the correct key, we do not have any estimation of the number of traces it would need to find the key, so we use the maximum number of traces in the attack set to show that the model did not succeed in finding the key.

5.1.2 L_2 Regularization

L_2 regularization shrinks the weights to values close to zero but rarely counts irrelevant features out. Therefore, we cannot see the feature selection effect as we

[4] Including the architectural and learning specifications.
[5] These values have been specified based on the grid search strategy.

Table 2: Hyperparameters of L_1, L_2, dropout, and early stopping regularization techniques and their experimented range.

Technique	**Hyperparameter**	**Range**
$Weightdecay(L_1)$	λ	$[5\times10^{-2}, 10^{-2}, 5\times10^{-3}, 10^{-3}, 5\times10^{-4}, 10^{-4}, 5\times10^{-5}, 10^{-5}, 5\times10^{-6}, 10^{-6}, 5\times10^{-7}, 10^{-7}]$
$Weightdecay(L_2)$	λ	$[5\times10^{-2}, 10^{-2}, 5\times10^{-3}, 10^{-3}, 5\times10^{-4}, 10^{-4}, 5\times10^{-5}, 10^{-5}, 5\times10^{-6}, 10^{-6}, 5\times10^{-7}, 10^{-7}]$
$Dropout$	$Dropout\_rate$	[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
$Earlystopping$	$Patience$	[10, 15, 20, 25, 30]

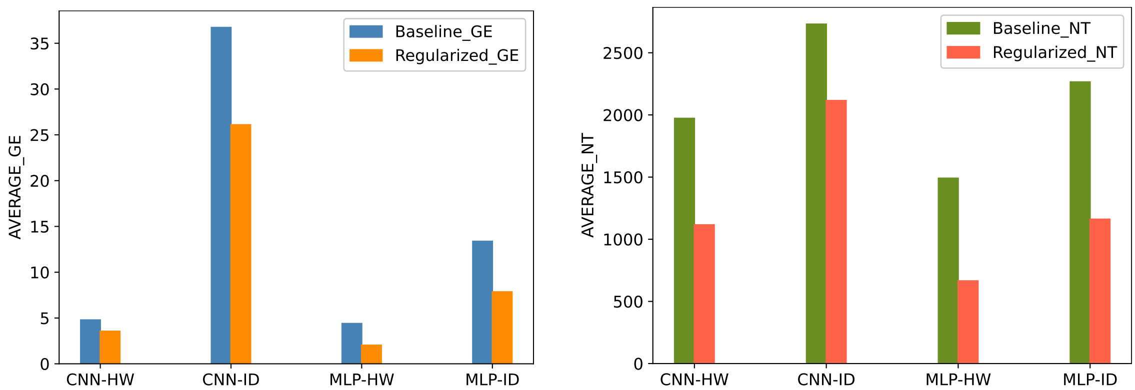

Fig. 1: The average performance with and without L_1 regularization, ASCAD dataset. The AVERAGE_GE (left) is calculated for baseline (blue) and regularized (orange) models. The AVERAGE_NT is calculated for baseline (green) and regularized (red) models.

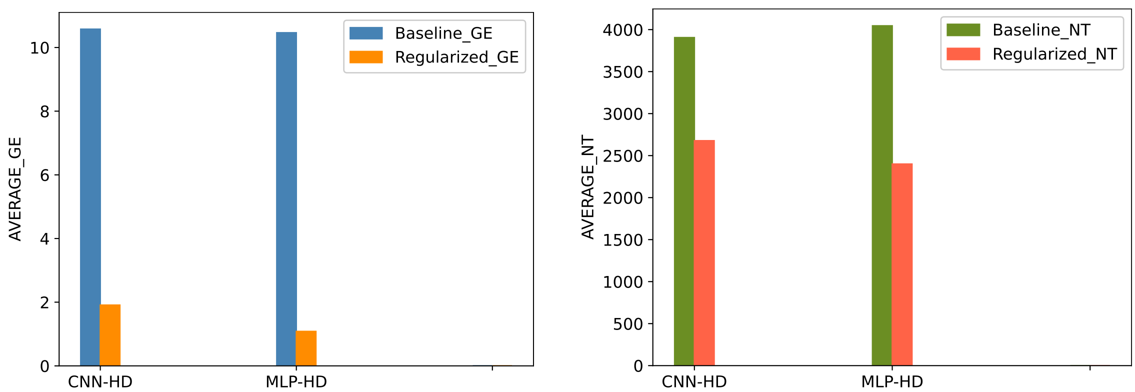

Fig. 2: The average performance with and without L_1 regularization, AES_HD dataset. The AVERAGE_GE (left) is calculated for baseline (blue) and regularized (orange) models. The AVERAGE_NT is calculated for baseline (green) and regularized (red) models.

saw for the L_1 regularization. Figure 3 shows the AVERAGE_GE and the AVERAGE_NT for the baseline and regularized models for the ASCAD dataset. As one can see, the regularized models can always reach lower AVERAGE_GE and AVERAGE_NT. The AVERAGE_GE sharp decrease in all four settings is noticeable. This decrease shows that the L_2 regularization improves models that cannot converge to $GE = 1$. The difference between baseline and regularized AVERAGE_GE is the most pronounced for the CNN with the ID leakage model. The effectiveness of L_2 regularization is also apparent when considering the average number of attack traces. Applying this regularization reduces AVERAGE_NT by half or less in three out of four settings. The improvement for the MLP with the ID leakage model is the most significant one. The ob-

Table 3: Six different combinations considered in the experimental setup.

Dataset	NN topology	Leakage model	Combinations
		HW	ASCAD_HW_MLP
ASCAD	MLP		ASCAD_HW_CNN
		ID	ASCAD_ID_MLP
			ASCAD_ID_CNN
AES_HD	CNN	HD	AES_HD_MLP
			AES_HD_CNN

served improvements are the consequence of reducing overfitting by making the weights smaller and close to zero.

Figure 4 demonstrates the AVERAGE_GE and the AVERAGE_NT for the baseline and regularized models for the AES_HD dataset. As the AVERAGE_GE and the AVERAGE_NT reflect, the average performance of baseline models is almost the same for both combinations. However, the L_2 regularization improves the results more for MLP models. Adding L_2 reduces the average GE five times in the CNN-ID and seven times in the MLP-HD settings. The influence on the required number of attack traces is not as significant, but it is still more in the MLP-HD combination. Besides the reasons mentioned in Section 5.1.1, this limited effect on AVERAGE_NT results from the noise level and type in this dataset.

L_1 and L_2 regularization considerably enhance the performance in both datasets. However, the L_2 regularization is slightly more effective for the ASCAD dataset, while L_1 is more effective for AES_HD. This observation stems from the distinct effect of these regularization techniques and the nature of noise in the considered datasets. L_1 bypasses the influence of irrelevant features by implicit feature selection while L_2 considers almost all the input features. The input in the ASCAD dataset is a narrowed window of the entire measurement, including the time samples corresponding to the first round S-box calculation. In the AES_HD dataset, the input contains all the samples collected during the AES decryption operation. Therefore, input includes many irrelevant samples collected during preprocessing, ten rounds of AES, and the final processing. The results indicate that both regularization techniques have different but equally valuable properties.

5.1.3 Dropout

Figure 5 shows the AVERAGE_GE and the AVERAGE_NT for the baseline and regularized models for the ASCAD dataset. Looking at Figure 5, one can see that the average GE for CNN-HW and MLP-ID combinations increased after applying dropout. This observation indicates that adding dropout may cause inferior performance in many cases. A closer look at the GE and NT measurements for each baseline and dropout regularized model shows that many models deteriorate after applying dropout (more description is in Section 5.2). However, we still can see a decrease in the AVERAGE_NT for these two combinations, which indicates the potential of dropout when it is effective. The effectiveness of this technique when it does not deteriorate a model is so significant that it can compensate for the increase in the required number of attack traces imposed by deteriorated models. In the other two combinations (CNN-ID and MLP-HW), the AVERAGE_GE and AVERAGE_NT decrease slightly, showing model deterioration happens here as well. Still, it is less in comparison with CNN-HW and MLP-ID combinations.

Figure 6 shows the results for the AES_HD dataset. With the decrease in the AVERAGE_GE and the AVERAGE_NT of regularized models, the deterioration effect is not detectable here. Still, one can see that the AVERAGE_GE and AVERAGE_NT improvement after applying dropout is less compared to L_1 and L_2.

Dropout works better when using MLP with fewer output classes (HW or HD leakage models), which is the indirect effect of the deep learning model size. In the case of the ID leakage model, there are 256 output classes, while the number of output classes for HW and HD leakage models is 9. However, since the number of input samples and training examples are the same for all leakage models, significantly larger models cannot be used for the ID leakage model. As a result, dropout regularization cannot produce enough distinct smaller networks to reflect the ensemble effect for 256 output classes.

5.1.4 Early stopping

Early stopping controls overfitting by manipulating the number of training iterations (epochs). This technique is adequate when the initial number of epochs is significantly larger than what the model needs to learn the underlying leakage distribution. Figure 7 indicates the AVERAGE_GE and AVERAGE_NT for baseline and regularized models for the ASCAD dataset. Based on the results, the AVERAGE_GE improvement is minimal in the ASCAD dataset, which means early stopping cannot improve models that do not converge without early stopping. The effect is more evident in CNN-ID and MLP-ID settings. If a baseline model does not reach $GE = 1$ without early stopping, using this technique does not significantly help the model to reach $GE = 1$. On the other hand, the epoch-wise evolution of GE shows that when GE reaches 1, it rarely increases again. Early stopping seems helpful regarding

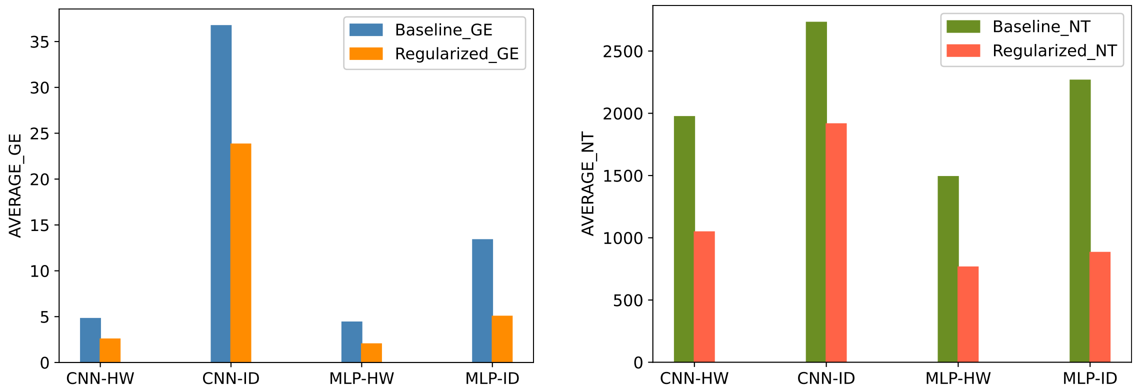

Fig. 3: The average performance with and without L_2 regularization, ASCAD dataset. The AVERAGE_GE (left) is calculated for baseline (blue) and regularized (orange) models. The AVERAGE_NT is calculated for baseline (green) and regularized (red) models.

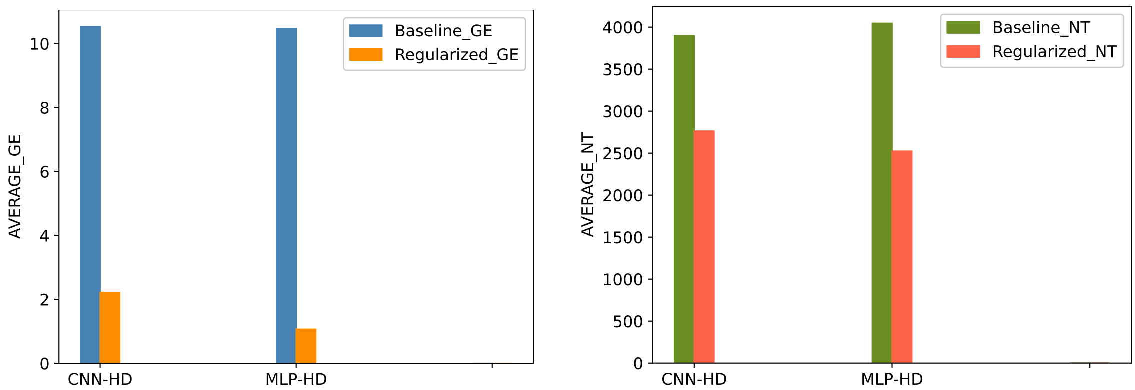

Fig. 4: The average performance with and without L_2 regularization, AES_HD dataset. The AVERAGE_GE (left) is calculated for baseline (blue) and regularized (orange) models. The AVERAGE_NT is calculated for baseline (green) and regularized (red) models.

the required number of attack traces. The regularized models managed to achieve smaller AVERAGE_NT in all four settings. Stopping a model as soon as it reaches $GE = 1$ reduces overfitting and helps the model find the key with fewer attack traces.

Finally, Figure 8 depicts the AVERAGE_GE and AVERAGE_NT for baseline and regularized models for the AES_HD dataset. The early stopping effectiveness on this dataset is similar to L_1 and L_2 regularization. This outcome shows that even early stopping regularization can improve GE and the required number of attack traces in a noisy dataset like AES_HD. This technique is considerably helpful in settings with MLP and limited output classes (HW and HD). To conclude, early stopping can perform better than dropout but not as well as L_1 and L_2 regularization.

5.2 Deterioration Rate

Although all results (Figure 1 to Figure 8) give insights into the influence of regularization techniques on DL-SCA attack performance, we can extract even more information from the experiments. One example is the percentage of models that deteriorate after using a regularization technique. We call this metric the "deterioration rate," which is the percentage of the regularized models that perform worse than their baseline counterparts. This metric shows how confident we can be that adding specific regularization techniques helps to improve the final performance. One can see the deterioration rates for L_1, L_2, dropout, and early stopping techniques in Table 4.

The deterioration rate for L_1 is a bit higher for CNN-ID and MLP-ID combinations for the ASCAD dataset compared to the other four settings. The selected leakage model (ID) causes this higher deterioration rate. The larger number of output classes makes

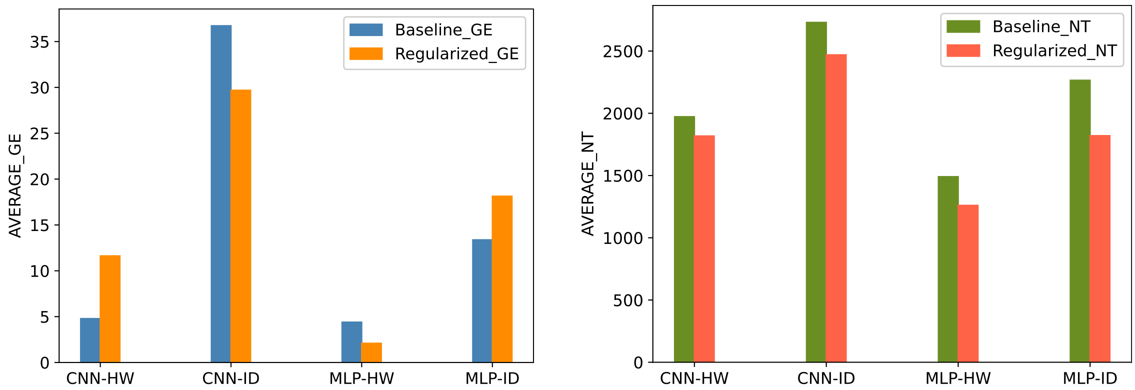

Fig. 5: The average performance with and without dropout regularization, ASCAD dataset. The AVERAGE_GE (left) is calculated for baseline (blue) and regularized (orange) models. The AVERAGE_NT is calculated for baseline (green) and regularized (red) models.

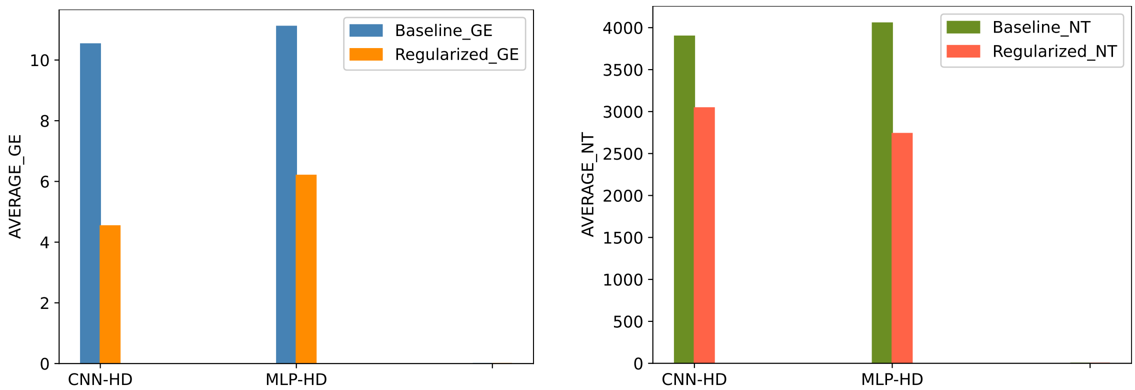

Fig. 6: The average performance with and without dropout regularization, AES_HD dataset. The AVERAGE_GE (left) is calculated for baseline (blue) and regularized (orange) models. The AVERAGE_NT is calculated for baseline (green) and regularized (red) models.

Table 4: The deterioration rate.

	ASCAD				**AES_HD**	
	CNN-HW	CNN-ID	MLP-HW	MLP-ID	CNN-HD	MLP-HD
L_1	1.5%	4.5%	0%	6.5%	3%	0.5%
L_2	0%	2%	0%	1.5%	2%	0.5%
Dropout	29.5%	19%	7.5%	23.5%	22.5%	10.5%
Early stopping	8.5%	18%	0.5%	19%	14%	4%

the model more sensitive to changes. Looking carefully, one can see that the columns with the ID leakage model in Table 4 show higher deterioration rates for all the regularization techniques.

The deterioration rate for L_2 regularization is less than 2% for all the settings showing that L_2-regularized models almost always perform better than the baseline model regardless of the selected settings. This outcome confirms that adding L_2 regularization will improve the performance or, in the worst case, will simply not help.

The situation is different when dropout is used. Deterioration rates in Table 4 indicate that dropout degrades many regularized models. CNN with the HW leakage model suffers the most from applying dropout. After that, MLP with ID leakage model and CNN with HD leakage models are the combinations that worsen considerably after using dropout. The variation in the leakage models and network topologies that experience the highest deterioration rate shows that the root cause of this observation is beyond the selected settings. The recognized deterioration after applying dropout is not

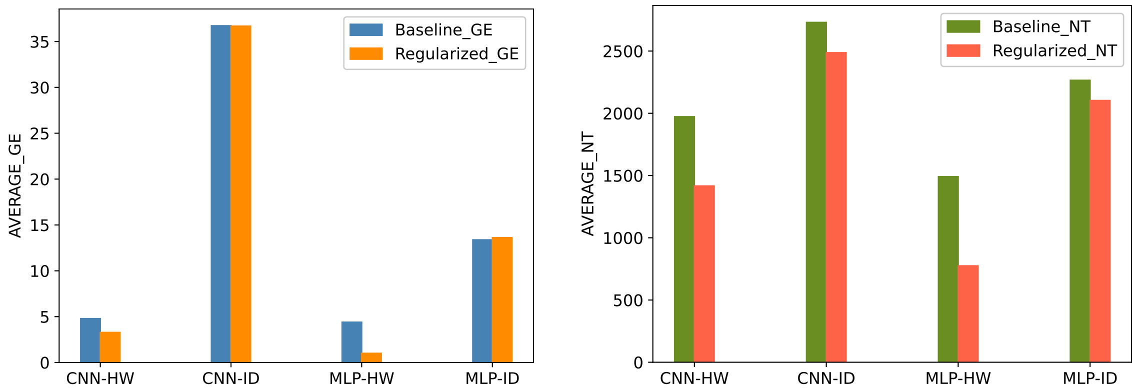

Fig. 7: The average performance with and without early stopping regularization, ASCAD dataset. The AVERAGE_GE (left) is calculated for baseline (blue) and regularized (orange) models. The AVERAGE_NT is calculated for baseline (green) and regularized (red) models.

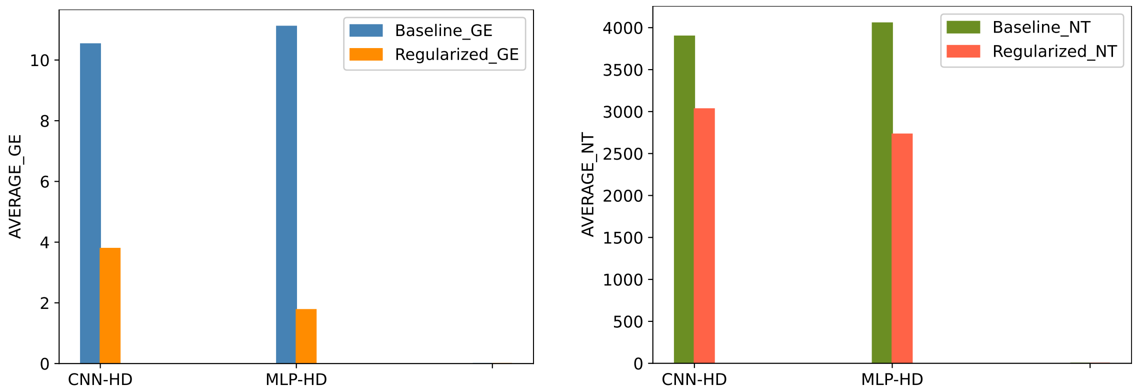

Fig. 8: The average performance with and without early stopping regularization, AES_HD dataset. The AVERAGE_GE (left) is calculated for baseline (blue) and regularized (orange) models. The AVERAGE_NT is calculated for baseline (green) and regularized (red) models.

unique to DL-SCA. In [6], Gabrin et al. reported reduced test accuracy after using dropout. Li et al. [18] showed that dropout could help accuracy but not in all cases. In [31], Srivastava et al. noted the necessity of changing the training and architectural hyperparameters to tune the model again after using dropout.

While the situation is better for early stopping compared to dropout, using it can still degrade some models. Again, CNN-ID and MLP-ID are two combinations that are worsened the most, resulting from the leakage model selection. MLP-HW and MLP-ID show low deterioration rates after applying early stopping. Looking at Table 4 column-wise, MLP-HW and MLP-HD combinations for the ASCAD dataset have the lowest deterioration rates for all regularization techniques. As mentioned in Section 4.3, MLP models are "generally able to find the key", while CNN models are "potentially able to find the key", and they should be tuned to work well. As a result, MLPs "absorb" changes in hyperparameters or added regularization techniques while applying small changes prevents CNNs from finding the key. In essence, the changes imposed on MLP models by regularization techniques do not deteriorate the models. This is why the combinations containing MLP show more improvements after applying regularization techniques, especially when models are smaller.

5.3 Profiling Time Changes

The average profiling time is the last considered metric that gives us useful information about the influence of regularization techniques on the models' performance. Table 5 shows the calculated profiling time for baseline and regularized models. The numbers show that early stopping can reduce profiling time. The only change this technique imposes on the models is forcing them to stop the training as soon as the accuracy does not change for a number of epochs. This way, it can stop the training process earlier and reduce the profiling time.

Table 5: Average profiling time for baseline and regularized models with different regularization techniques.

	ASCAD				AES_HD	
	CNN-HW	CNN-ID	MLP-HW	MLP-ID	CNN-HD	MLP-HD
Baseline models	933	986	651	663	825	465
L_1	3739	4620	1854	2603	6027	4832
L_2	3934	3958	1396	2097	6881	4194
Dropout	2420	3299	1209	1626	3759	1967
Early stopping	632	898	165	579	432	256

All the other techniques increase the profiling time considerably.

6 Discussion

So far, the experiments have investigated the influence of different regularization techniques on DL-SCA. Based on the experiments, the overall view confirms the dependency of regularized model improvement on different factors like the level of the dataset's noise, leakage model, and neural network topology. In other words, the improvement that a specific regularization technique offers differs per model and depends on the model's characteristics. However, it is still relevant to answer these two general questions:

- What is the most effective regularization technique among L_1, L_2, dropout, and early stopping?
- When does a regularization technique work at its best?

This section tries to find an answer to these two questions.

6.1 Different Techniques Effectiveness in General

Considering the results in Section 5, it is not easy to say which regularization technique is the most effective among the four experimented ones. As mentioned earlier, the effectiveness of a regularization technique depends on different factors. However, Figures 9 and 10 try to give an overall comparison of L_1, L_2, dropout, and early stopping effectiveness in DL-SCA. The plots present the required number of attack traces (NT) over the size of all the baseline or regularized models for ASCAD_HW_MLP and AES_HD_CNN combinations.

As shown in Figure 9, the baseline models spread almost all around the plot (the green spots) for the ASCAD_HW_MLP combination. Adding L_1 (red) or L_2(yellow) regularization pushes the spots to the part of the plot so that the NT is less than 2000 traces for all L_1 and L_2-regularized models. In contrast, while adding dropout (blue) increases the density at the bottom of the plot, many spots remain on the upper side, which means it cannot improve many models as good as L_1 and L_2. Early stopping (purple) influence is considerably better than dropout but not as good as L_1 and L_2. Based on the dispersion of spots, L_1 improves the models slightly more than L_2.

The plots in Figure 10 (AES_HD_CNN combination) confirm the mentioned conclusion. In this combination, most of the selected models are small.[6] Therefore, the spots group in the left part of the plots. However, one can still see the sparseness of baseline models along the y-axis. Adding L_1 and L_2 regularization pushes the red and yellow spots to the lower corner. The better effectiveness happens for L_1 again. Early stopping pushes down the purple spots, and its effectiveness is close to L_2 regularization. In the case of dropout, while there are a few spots lower than 1500 (the other three regularization techniques could not push more than one spot under 1500), the rest of the spots mostly spread from 2000 to around 3500. This spreading range shows the poorer effectiveness of dropout.

6.2 Implicit and Explicit Regularization Comparison

Recent works have studied implicit and explicit regularizations and their connections. Here, we use simplified definitions of those terms without getting into the details to specify when it is a good idea to use regularization techniques. Implicit regularization[7] is the effect imposed by the characteristics of the neural network architecture and the learning algorithm. This regularization does not change the objective function [2, 10]. The gradient descent algorithm (and stochastic gradient descent as its extension, which even works better) offers implicit regularization inherently [13]. On the other hand, explicit regularization modifies the expected loss and objective function and reduces the effective capacity of a given model to reduce overfitting.

[6] The small size of neural networks is imposed during the hyperparameter search and baseline model selection. In general, hyperparameter tuning imposes selecting smaller models that offer a better implicit regularization.

[7] Also, algorithmic regularization.

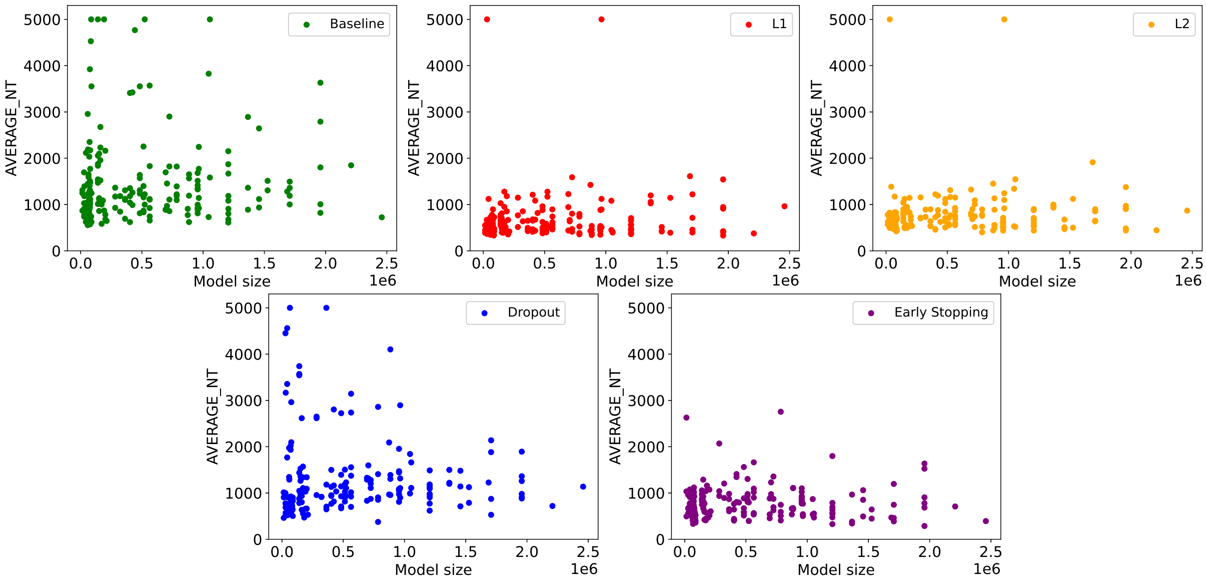

Fig. 9: The dispersion of baseline and regularized models NT over their size in ASCAD_HW_MLP combination.

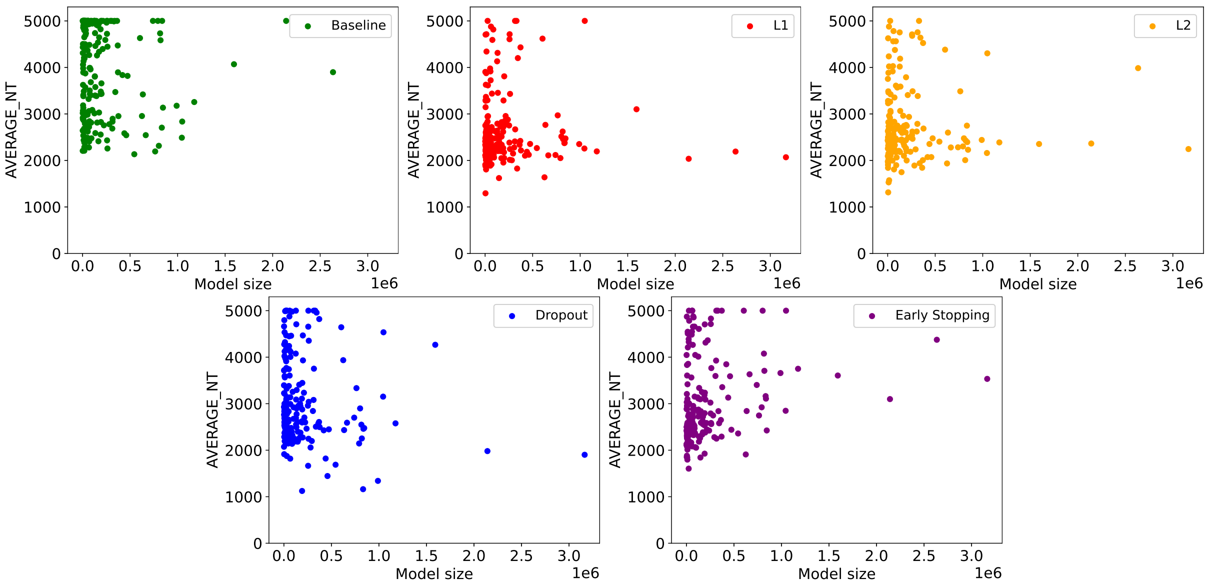

Fig. 10: The dispersion of baseline and regularized models NT over their size in AES_HD_CNN combination.

Explicit regularization is mostly provided by regularization techniques like dropout and norm penalties [2, 10].

Implicit and explicit notions and the best and the worst baseline models in each combination are used to state when applying regularization techniques is effective. Among the baseline models selected for each combination, some models work very well and can rank the correct key in the first place with a few attack traces. These neural network models are models that offer adequate implicit regularization by themselves. They reduce overfitting and increase generalization sufficiently with the learning algorithm. Also, some other models can only find the key with a significant number of traces or cannot rank the key in the first place even after using 5000 attack traces. The implicit regularization of these models is insufficient, and they usually need extra regularization to reduce overfitting (and increase generalization).

In Figure 11, one can see the AVERAGE_NT for ten baseline models that perform the best (green spots in the left plot) and ten baseline models that perform the worst (green spots in the right plot) among 200 selected baseline models in ASCAD_HW_CNN combination. In the left plot in Figure 11, one can see that the best

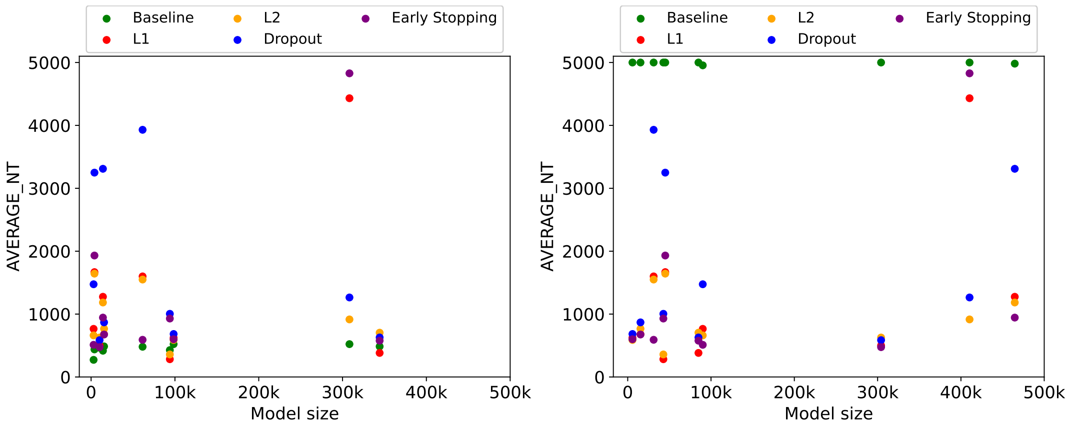

Fig. 11: Left: Ten best baseline models in ASCAD_HW_CNN combination along with their L_1, L_2, dropout, and early stopping regularized counterparts. Baseline models have better performance than their regularized counterparts. Right: Ten worst baseline models in ASCAD_HW_CNN combination along with their L_1, L_2, dropout, and early stopping regularized counterparts. Regularized models have better performance than their baseline counterparts

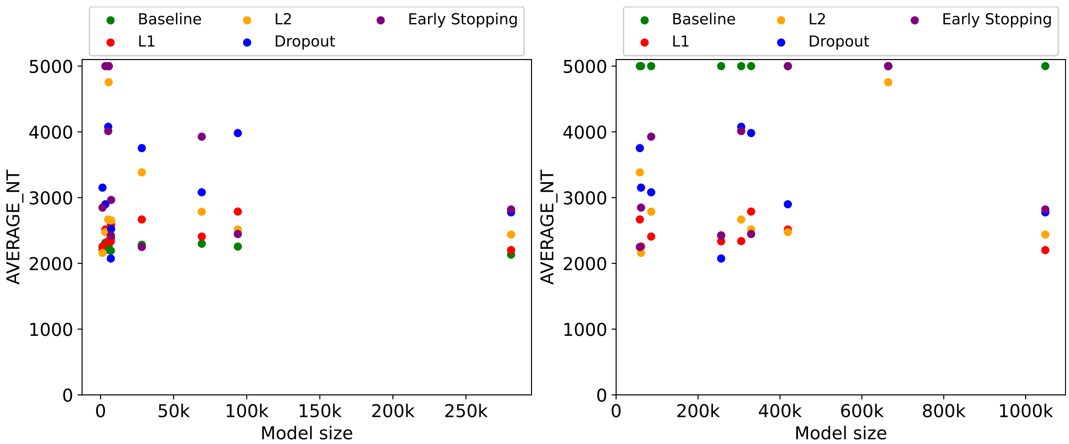

Fig. 12: Left: Ten best baseline models in AES_HD_CNN combination along with their L_1, L_2, dropout, and early stopping regularized counterparts. Baseline models have better performance than their regularized counterparts. Right: Ten worst baseline models in AES_HD_CNN combination along with their L_1, L_2, dropout, and early stopping regularized counterparts. Regularized models have better performance than their baseline counterparts

ten selected baseline models have an acceptable performance before applying any regularization techniques. On the other hand, their counterpart regularized models perform worse in almost all cases. The right plot of Figure 11 indicates the worst ten selected baseline models in the ASCAD_HW_CNN combination. As the opposite of best-selected models, the AVERAGE_NT for these baseline models is around 5000, while regularized models' performance is far better. In many cases, the performance of the worst models after applying a regularization technique is comparable with the best baseline models. Figure 12 shows the same behavior for the AES_HD_CNN combination. As depicted in Figure 12, the best ten baseline models worsen after adding regularization techniques. At the same time, the worst ten baseline models show good performance after applying regularization techniques.

This observation shows that regularization techniques are more effective when the selected baseline model does not offer enough regularization by itself. Thus, it seems more efficient to use regularization techniques instead of investing significant time and effort into searching for models.

7 Conclusions and Future Work

This work provides an in-depth study of L_1, L_2, dropout, and early stopping influence on the performance of DL-

SCA (six different combinations of datasets, leakage models, and deep learning network topologies). Our experimental results show that while all these techniques can improve the DL-SCA performance, some of them are more effective than others. Considering the average required attack traces (AVERAGE_NT), the average guessing entropy (AVERAGE_GE), and the deterioration rate, we observe that L_1 and L_2 are the most effective regularization techniques. While early stopping has moderate effectiveness, it can reduce training time. In comparison, other techniques increase training time considerably. Since the dropout deterioration rate is very high compared to the other techniques, and it increases the training time, we recommend using it carefully. Overall, there is potential in using regularization techniques to resolve the overfitting issue in SCA, but they should be used with care and consideration for their strength and weakness.

In future work, it would be interesting to compare the influence of other, more advanced regularization techniques, like batch normalization and data augmentation, with the current work results. Another interesting direction is combining different techniques and checking if adding two or more regularization techniques can improve the performance further. Besides, while we used ASCAD random key and AES_HD for our experiments, other datasets can be used to investigate regularization techniques' effectiveness further.

Acknowledgments

This work received funding in the framework of the NWA Cybersecurity Call with project name PROACT with project number NWA.1215.18.014, which is (partly) financed by the Netherlands Organisation for Scientific Research (NWO). Additionally, this work was supported in part by the Netherlands Organization for Scientific Research NWO project DISTANT (CS.019).

References

1. Agence nationale de la sécurité des systèmes d'information (ANSSI). ASCAD. Github repository, 2018. `https://github.com/ANSSI-FR/ASCAD`.
2. S. Arora, N. Cohen, W. Hu, and Y. Luo. Implicit regularization in deep matrix factorization. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 7411–7422, 2019.
3. E. Brier, C. Clavier, and F. Olivier. Correlation power analysis with a leakage model. In M. Joye and J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.
4. E. Cagli, C. Dumas, and E. Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures - profiling attacks without preprocessing. In W. Fischer and N. Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 45–68. Springer, 2017.
5. S. Chari, J. R. Rao, and P. Rohatgi. Template attacks. In B. S. K. Jr., Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.
6. C. Garbin, X. Zhu, and O. Marques. Dropout vs. batch normalization: an empirical study of their impact to deep learning. *Multim. Tools Appl.*, 79(19-20):12777–12815, 2020.
7. A. Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow.* " O'Reilly Media, Inc.", 2022.
8. B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel. Mutual information analysis. In E. Oswald and P. Rohatgi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, volume 5154 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2008.
9. I. J. Goodfellow, Y. Bengio, and A. C. Courville. *Deep Learning.* Adaptive computation and machine learning. MIT Press, 2016.
10. A. Hernández-García and P. König. Data augmentation instead of explicit regularization. *CoRR*, abs/1806.03852, 2018.
11. A. Heuser and M. Zohner. Intelligent machine homicide - breaking cryptographic devices using support vector machines. In W. Schindler and S. A. Huss, editors, *Constructive Side-Channel Analysis and Secure Design - Third International Workshop, COSADE 2012, Darmstadt, Germany, May 3-4, 2012. Proceedings*, volume 7275 of *Lecture Notes in Computer Science*, pages 249–264. Springer, 2012.
12. S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In F. R. Bach and D. M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015.
13. N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.
14. J. Kim, S. Picek, A. Heuser, S. Bhasin, and A. Hanjalic. Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(3):148–179, 2019.
15. P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19,*

1999, Proceedings, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
16. A. Krogh and J. A. Hertz. A simple weight decay can improve generalization. In J. E. Moody, S. J. Hanson, and R. Lippmann, editors, *Advances in Neural Information Processing Systems 4, [NIPS Conference, Denver, Colorado, USA, December 2-5, 1991]*, pages 950–957. Morgan Kaufmann, 1991.
17. L. Lerman, G. Bontempi, and O. Markowitch. A machine learning approach against a masked AES - reaching the limit of side-channel attacks with a learning model. *J. Cryptogr. Eng.*, 5(2):123–139, 2015.
18. X. Li, S. Chen, X. Hu, and J. Yang. Understanding the disharmony between dropout and batch normalization by variance shift. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 2682–2690. Computer Vision Foundation / IEEE, 2019.
19. S. Mangard, E. Oswald, and T. Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.
20. L. Masure, C. Dumas, and E. Prouff. A comprehensive study of deep learning for side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1):348–375, 2020.
21. G. Perin, I. Buhan, and S. Picek. Learning when to stop: A mutual information approach to prevent overfitting in profiled side-channel analysis. In S. Bhasin and F. D. Santis, editors, *Constructive Side-Channel Analysis and Secure Design - 12th International Workshop, COSADE 2021, Lugano, Switzerland, October 25-27, 2021, Proceedings*, volume 12910 of *Lecture Notes in Computer Science*, pages 53–81. Springer, 2021.
22. G. Perin, L. Chmielewski, and S. Picek. Strength in numbers: Improving generalization with ensembles in machine learning-based profiled side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(4):337–364, 2020.
23. G. Perin and S. Picek. On the influence of optimizers in deep learning-based side-channel analysis. In O. Dunkelman, M. J. J. Jr., and C. O'Flynn, editors, *Selected Areas in Cryptography - SAC 2020 - 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers*, volume 12804 of *Lecture Notes in Computer Science*, pages 615–636. Springer, 2020.
24. S. Picek, A. Heuser, A. Jovic, S. A. Ludwig, S. Guilley, D. Jakobovic, and N. Mentens. 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*, pages 4095–4102. IEEE, 2017.
25. S. Picek, A. Heuser, G. Perin, and S. Guilley. Profiled side-channel analysis in the efficient attacker framework. In V. Grosso and T. Pöppelmann, editors, *Smart Card Research and Advanced Applications - 20th International Conference, CARDIS 2021, Lübeck, Germany, November 11-12, 2021, Revised Selected Papers*, volume 13173 of *Lecture Notes in Computer Science*, pages 44–63. Springer, 2021.
26. S. Picek, G. Perin, L. Mariot, L. Wu, and L. Batina. Sok: Deep learning-based physical side-channel analysis. *ACM Comput. Surv.*, oct 2022. Just Accepted.
27. E. Prouff, R. Strullu, R. Benadjila, E. Cagli, and C. Dumas. Study of deep learning techniques for side-channel analysis and introduction to ASCAD database. *IACR Cryptol. ePrint Arch.*, page 53, 2018.
28. J. Quisquater and D. Samyde. Electromagnetic analysis (EMA): measures and counter-measures for smart cards. In I. Attali and T. P. Jensen, editors, *Smart Card Programming and Security, International Conference on Research in Smart Cards, E-smart 2001, Cannes, France, September 19-21, 2001, Proceedings*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001.
29. J. Rijsdijk, L. Wu, G. Perin, and S. Picek. Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):677–707, 2021.
30. D. Robissout, G. Zaid, B. Colombier, L. Bossuet, and A. Habrard. Online performance evaluation of deep learning networks for profiled side-channel analysis. In G. M. Bertoni and F. Regazzoni, editors, *Constructive Side-Channel Analysis and Secure Design - 11th International Workshop, COSADE 2020, Lugano, Switzerland, April 1-3, 2020, Revised Selected Papers*, volume 12244 of *Lecture Notes in Computer Science*, pages 200–218. Springer, 2020.
31. N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014.
32. J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient object localization using convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 648–656. IEEE Computer Society, 2015.
33. G. Zaid, L. Bossuet, A. Habrard, and A. Venelli. Methodology for efficient CNN architectures in profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(1):1–36, 2020.