





The Forgotten Hyperparameter: Introducing Dilated Convolution for Boosting CNN-Based Side-Channel Attacks

Servio Paguada^{1,2}(✉)  and Igor Armendariz²(✉) 

¹ Digital Security Group, Radboud University, Nijmegen, The Netherlands
servio.paguada@saula@ru.nl

² Ikerlan Technology Research Centre, Arrasate-Mondragón, Gipuzkoa, Spain
{slpaguada,iarmendariz}@ikerlan.es

Abstract. In the evaluation of side-channel resilience, convolutional neural network-based techniques have been proved to be very effective, even in the presence of countermeasures. This work is introducing the use of dilated convolution in the context of profiling side-channel attacks. We show that the convolutional neural network that uses dilated convolution increases its performance by taking advantage of the leakage distributed through scattered points in leakage traces. We have validated the feasibility of the proposal by comparing it with the state-of-the-art approach. We have conducted experiments using ASCAD (with random key), and as a result the guessing entropy of the attack converges to zero for around 550 synchronized traces and for 3 000 desynchronised traces. In both groups of experiments, we have used the same architecture to train the model, changing just dilatation rate and kernel length, which indicates a reduction of the complexity in the deep learning model.

Keywords: Profiled attacks · Side-channel analysis · Dilated convolutions · CNNs · Dilatation rate

1 Introduction

The profiled attack is considered to be one of the most powerful attacks in Side-Channel Analysis (SCA). The overall idea is to build a model (profile) by using a clone of the target device and then use this model to attack the non-controlled target device. Template Attack is the first example of these types of attacks [6], and some related works followed introducing new attack scenario and improving the execution phase [10, 14, 33]. Profiled attacks became even more powerful with the usage of deep learning techniques. Since profiled attacks can be seen as a classification problem, existing deep learning architectures like VGG [34] were taken as a baseline for applications in SCA [20, 32]. Several publications presented different types of deep learning models such as Multi-Layer Perceptron (MLP) [23, 24], and Convolutional Neural Network (CNN) [4] that were able to compromise the secure implementation of cryptographic algorithms. They both showed the potential to outperform previous results of template attacks.

© Springer Nature Switzerland AG 2020

J. Zhou et al. (Eds.): ACNS 2020 Workshops, LNCS 12418, pp. 217–236, 2020.

https://doi.org/10.1007/978-3-030-61638-0_13

Many efforts are made to improve our understanding of how these deep learning-based attacks work. One direction is to find the best combinations of hyperparameter values of the learning algorithm, e.g. those that improve the attack. Results of those efforts are the different methodologies that explain how evaluators and researchers should build CNN models. Thanks to that, tests over commercial devices for assessing their resilience are becoming more feasible and reliable.

Recently, the methodology presented in [40] has shown that, by reducing the use of relevant features in the convolutional blocks, not only the efficiency but also the effectiveness of the attack increases. Additionally, the experiments made in [21] suggest that an important improvement in CNN-based attacks could be achieved, if features where the intermediate values leak and where the mask leaks are combined in its convolution operation. CNN almost always are built with two main parts, the convolutional part, and the classification part. The former is composed of convolutional blocks, and it is the part where that convolution operation is performed.

We take the arguments from the two papers mentioned above as our starting point. Then, we conducted preliminary experiments using dilated convolutions. Dilated convolution is a type of convolutional block where its kernel is modified (dilated) in a way such that it covers wider areas than the normal convolution, and at the same time does not overuse relevant points. Dilated convolution is used to face the problem of scattered dependencies, meaning that the leak is scattered through sample points in the leakage traces [16, 21, 22]. Some works in high-order side-channel also explain the phenomenon from the perspective of this analysis [2, 11, 36, 38]. The dilatation of the kernel is controlled by a hyperparameter known as *dilatation rate*. It turned out that by using dilated convolution, we increased the performance of the CNN model. The preliminary experiments were conducted using ASCAD fixed key dataset [32], with the CNN model suggested in the latest work [40]. The results have shown that dilated convolution is feasible and might represent a useful hyperparameter when building CNN models for SCA. Then, ASCAD random key dataset [32] was used in experiments where the guessing entropy [35] converges to zero for around 550 traces, whilst the baseline value produced by a model from [21] converges to zero for around 4500 traces. The results of these latter experiments were achieved by a CNN model that uses dilated convolutions.

Contribution

By taking dilated convolution into account, we are aiming to understand better how the architecture of CNNs should be designed for evaluations. To be more specific, we introduce the use of dilated convolution by explaining and demonstrating how this type of kernels is feasible to evaluate implementations of cryptographic algorithms. As it turns out, it brings a new possibility to reduce the complexity of the deep learning models. To prove what we claim, we conducted the following:

1. The first experiment compares state-of-the-art results from [40], showing the feasibility of the dilated convolution. Gradient visualization [25], *Signal-to-Noise Ratio* (SNR) [22], and Weight visualisation [40] techniques are used to evaluate the classification and feature selection of both approaches. Further experiments involve a CNN model that outperforms the state-of-the-art approach. These latter experiments aim to compare CNN’s effectiveness by using different values of kernel length, and dilatation rate. They demonstrate the effect of reducing redundant points in the first convolutional block, but also the importance of combining enough relevant ones.
2. Experiments for mimicking the behaviour of the dilated convolution using small values of kernel lengths and stride are also performed. Proving that in fact, dilated convolution takes advantage of the long-range dependencies leakage when combining the involved sample points in the same convolution operation.
3. Experiments that show how dilated kernels reduce the impact of the desynchronisation. This latter experiment aims at reducing the complexity of the convolutional part. As we show, the deep learning model used for bypassing desynchronisation is almost the same as the experiment without this effect. Changes were only made in the kernel length and the dilatation rate of the first convolutional block.
4. We also propose considerations that serve as a guide when using dilated convolutions.

Paper Organisation

The remainder of this paper is organised as follows. Section 2 includes a background in CNN, theoretical aspect of normal and dilated convolutions, the datasets we used, and the metric to assess the performance as well as the visualisation techniques. Section 3 summarises previous works regarding CNN for SCA. Section 4 presents the consideration when building dilated convolution-based CNN. Section 5 and Sect. 6 contain the result of experiments and the conclusion, respectively.

2 Background

In this section, we start by giving an overview of convolutional neural networks. We also include some theory on how the dilatation rate affects the convolutional operation. To show the arithmetic relation, we have used the mathematical expression of the most general case of convolution operation [13], which involves all the possible variables, i.e. padding, kernel and input map lengths, and stride.

2.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a type of neural network; they were initially designed to address classification problems in images. Some recent works

have also shown their performance for time series analysis [8,28]. As depicted in Fig. 1(a), CNNs are composed of two main parts; the convolutional part and the fully-connected part. The convolutional part is where the convolutions take place, and as the name suggests they are the convolution layers of a CNN. In such a layer, a kernel is required to perform the operation using the input map; being the first convolution layer the input signal. The elements of the kernel are also known as weights. The back-propagation operation updates these weights after a loss function determines the error in the classification. Back-propagation is a powerful characteristic of the neural network learning algorithms [15].

The input map contains features that characterise itself; not all features are relevant; in fact, irrelevant features lead to an ineffective neural network [27]. When the input map is convolved with a kernel of length l_k , the sparse combination of features produces a feature map, whose elements represent more abstract features than the ones in the input. Such a resultant feature map serves as input for the next convolutional block. The number of kernels used represents the number of feature maps that will be created; all of them will characterise in an abstract way the input signal. Feature map could also be seen as a reduction of the space volume of information. The length l_k determines how many points are involved in computing such reduction. Both, the kernel length, and the number of kernels in the convolutional block are hyperparameters for the neural network architecture.

As those feature maps progress through the hidden convolution layers, even more abstract feature maps are created. A 1D convolution operation can be expressed as in Eq. 1, where f is the input map and k is the kernel; a graphic example is depicted in Fig. 1(b). Feature map might pass to a pooling layer; this layer acts as a downsampler which takes the output of a convolution layer and creates a spatial feature map. This spatial feature map is an invariant representation of the original sparse features in the input map; we named this pooling feature map in Fig. 1(b). In the convolutional part, the operation must catch relevant features that aim for the proper classification. Which also implies that the goal is not to have a long kernel that mixes many features in the operation; indeed, such a practice could lead to poor results. At the end of the convolutional part, the feature maps are reduced to a vector (i.e. flattened) to feed it into the fully connected part where the classification is conducted.

$$f[x] \otimes k[x] = \sum_{n=-\infty}^{\infty} f[n] \cdot k[x - n] \quad (1)$$

Normal Convolutions

Different hyperparameters setup the convolution operation, and they affect the output dimension l_o of the resultant feature map. To explain this relation, in Eq. 2 it is assumed a convolution layer with kernel of length l_k , a padding parameter p whose value determines the dimension with zero values used to contour input map; this latter would have an original length denoted by l_i . Finally, the

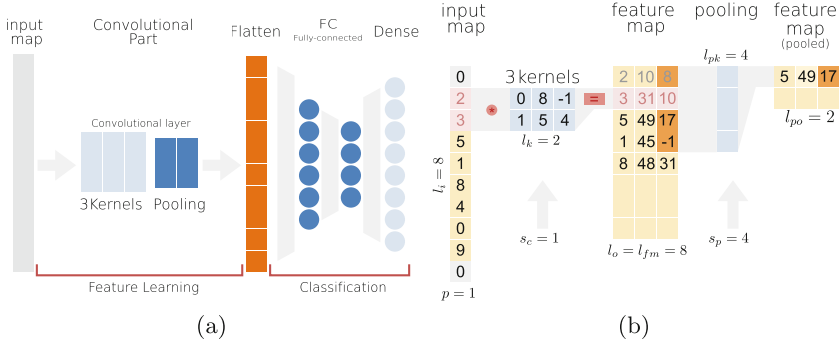


Fig. 1. (a) Convolution neural network common architecture ($N = 0$); (b) Convolution operation example

stride parameter s_c represents the distance between two consecutive applications of the kernel over the input map.

$$l_o = l_{fm} = \left\lfloor \frac{l_i + 2p - l_k}{s_c} \right\rfloor + 1 \tag{2}$$

As we mentioned, the output of the convolution layer could pass through a pooling operation. Commonly, this is the case because such operation grants invariant property to a CNN against small translations of the input maps. Pooling layer uses a window that we called it pooling kernel, its length l_{pk} represents the number of features taken from the feature map to conduct the pooling operation. A stride value s_p (named pooling stride) controls the displacement of the pooling kernel through the feature map. This operation also affects the output dimension l_o that in such a case becomes l_{po} , Eq. 3 shows the relation. Although there are different kinds of pooling operation Eq. 3 applies for all of them. Figure 1(b) also show example of these parameters.

$$l_{po} = \left\lfloor \frac{l_{fm} - l_{pk}}{s_p} \right\rfloor + 1 \tag{3}$$

Dilated Convolutions

A dilated convolution takes place when the effective size of the kernel is increased by a factor, known as dilatation rate dr . Normally, this factor is bigger than 1, (where $dr = 1$ is a normal convolution layer) which allows the convolution operation to cover a wider area, without heavily affecting the original convolution operation performance. The effect could be seen as if we take a kernel and inflate it by inserting zeros between kernel elements; some features are nullified from the operation because of those zero value elements; i.e. there are terms of zero value in the right side of Eq. 1. Therefore, only the non-zero kernel elements contribute to compose the feature maps; using features that are even more sparse. The rest

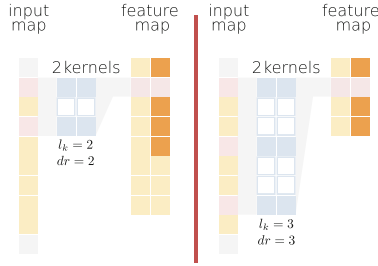


Fig. 2. Dilated convolutions explanatory illustration, two kernels with different lengths and different values of dilatation rate (considering $s_c = 1$)

of the operations remain as they were normal convolutions. Each time kernel moves according to the stride value, the resultant feature map includes less redundant features and at the same time keeps enough relevant ones. Equation 4 shows how the length of a kernel l_k is affected by the dilatation rate dr . Figure 2 illustrates a dilated convolution with two examples.

$$\hat{l}_k = l_k + (l_k - 1)(dr - 1) \tag{4}$$

It is clear that, dilated convolution also changes the output dimension of the feature maps. Equation 5 is a modification of Eq. 2, and shows the relation of the output dimension, when dilatation rate (dr) already changed the kernel length \hat{l}_k . It is also clear that when $l_k = 1$ Eq. 5 becomes exactly as Eq. 2; this is something to be considered for choosing the criteria to build the deep learning architecture (Sect. 4).

$$l_o = l_{fm} = \left\lfloor \frac{l_i + 2p - \hat{l}_k}{s_c} \right\rfloor + 1 \tag{5}$$

2.2 ASCAD Dataset

ASCAD dataset was introduced in [32] with the purpose of being a common dataset, to conduct benchmarking related to side-channel profiled attacks using machine learning techniques. The ATMega8515 was the device from which the traces were collected. The EM radiation was recorded while the device executed an AES-128 [12] software implementation. A masking countermeasure was used to protect the cryptographic operation [3]. In the acquisition campaign, an oscilloscope with the EM sensor sampled the signal at 2 GS/s.

The structure of this dataset allocates traces into two groups; *profiling_traces* which contains traces to perform the profiling stage and *attack_traces*, which contains traces to perform the attack stage. The dataset has two versions, collected traces with fixed key encryption and collected traces with random key

encryption. In the work [40], they used ASCAD fixed key; to establish a comparison we have used this dataset in the first experiment. The *profiling_traces* group contains 50 000 traces and the *attack_traces* group contains 10 000. The traces in both of the groups have 700 sample points, and they are the points of interest of the crypto operation (the masked S-box for the sensitive value).

For the rest of the experiments, we have used ASCAD random key version since it represents a challenging way to conduct a profiling attack. For this version, *profiling_traces* contains 200 000 traces and *attack_traces* contains 100 000 traces. In the experimental section (Sect. 5), Tables 2 and 3 show the amount of traces of each group used to perform training. Each trace has a length of 1 400 sample points. As in the fixed version, these are the points of interest of the crypto operation.

Traces can be desynchronised by applying a threshold (N) that moves traces around x -axis. The common values to perform the benchmarking are $N = 0$, $N = 50$, and $N = 100$. In the experiment, we have only used $N = 0$ and $N = 100$; the latter value lets enough evidence that the proposed method is feasible as well for $N = 50$.

Sensitive value of traces has the model represented by the Eq. 6 where the value of Z represents the class that labelled the traces associated with the same index. The byte that is intended to exploit is the third value ($i = 3$). The p represents the plain text, and k is the possible key hypothesis.

$$Z[i] = \text{Sbox}[p[i] \oplus k[i]] \quad (6)$$

All the samples were standardised and normalised between 0 and 1 to accelerate the learning process [15]. For the first experiment, we have used the same training hyperparameters as in [40] as well as the same setup for the attack phase. For the experiments using ASCAD random key, the attack traces are randomly shuffled and a battery of 100 attacks are performed to obtain the average value.

2.3 Guessing Entropy

The guessing entropy (GE) [35] is commonly used as a metric to assess the performance of a side-channel attack. It represents the average number of the key candidates required to obtain the secret key k after conducting a side-channel analysis. To give a specific example, let consider the 5000 randomly chosen traces, an attack using these traces results in a GE vector $\vec{g} = [g_1, g_2, \dots, g_{|K|}]$ where K represents the keyspace. Each component g_i is ordered from the maximum to the minimum value of probability. Then, the GE is the average position of k in the vector \vec{g} over many experiments. By using a battery of 100 experiments, we obtain an *averaged guessing entropy* for adequately estimating the performance of the attack.

2.4 Visualisation of Feature Selection

The ability of a neural network to extract relevant feature can be visualised using the following techniques. *Gradient visualisation* [25] computes the value

of the derivatives regarding the input trace, the resultant value is used to point out what feature needs to be modified the least, to affect the loss function the most. This technique gives information about what time samples influence the most in the classification; when those time samples are compared with other visualisation techniques, one can evaluate how well the neural network extracts the important features. *Signal-to-Noise Ratio* (SNR) [22] points out the time samples in leakage traces that contain exploitable information. We used it to compare with gradient visualisation and see how the time points match in each result. *Weight Visualisation* [40] helps to understand how the convolutional part of a CNN performs the feature selection. By comparing the shape of this latter technique with gradient visualisation, one can evaluate how well the feature learning part did to help the classification part.

3 Works in CNN for Side-Channel Analysis

In this section, we briefly mention related works in the context of using CNN for SCA.

The first works in using CNN architecture for SCA were [20,32] concluding that VGG [34] was the best architecture in addressing side-channel analysis. After these papers, others became available showing results against countermeasures such as those jitter-based and masking [4,5]. Other publications focused on understanding theoretically, and visually how deep learning models are capable of bypassing the countermeasure and compromising the security such as [25,29,40]. CNN has also been used for non-profiled attacks [37]. Additionally, in the same contribution, the ability of deep neural networks to fit high-order side-channel leakages was shown.

Recent works have looked into modifications of the architecture of the CNN, not just on changing the hyperparameters, but trying to feed the neural network with more inputs (with additional data) to improve the performance [18]. Feature selection has also been covered in SCA, Picek et al. show the relevance of applying techniques for choosing features to increase the performance of the learning algorithm [30]. In [21], the authors present several experiments for building the first convolutional block related to scattered leaks¹.

One of the most recent works in showing a methodology to build a CNN for side-channel analysis is [40]. In that paper, the conducted analysis shows how to setup kernels in the first convolutional block; for avoiding composing feature maps that include many irrelevant as well as redundant features. As we already mentioned in the previous section, such mis practice impacts negatively on the performance. The work assesses other aspects concerning pooling operation, and they claim that a *pooling stride* should also be set in a way that the pooling kernel does not take repetitive features.

All these works have integrated reliable conclusions that are still used in the state-of-the-art. To the best of our knowledge, dilated convolutions have not been presented into SCA context yet. Some references on using them in image classification applications exist [7,17,28,39].

¹ i.e. intermediate value and mask leaks.

4 Dilated Convolutions Design Considerations

Here we discuss the design considerations for CNN architectures with dilated convolutions for SCA, as well as the potential pitfall of using them.

We offer a takeaway when one opts for the presented approach. As the reader can see, they are substantially similar to the state-of-the-art. This is because dilated convolution follows the basis of avoiding collecting irrelevant features. What follows are the criteria and their explanation for building deep learning architecture. At the same time, they justify the usage of the dilated convolution. It must be understood that dilatation rate is also a hyperparameter so that the process to find the best value for it relies on a trade-off between it and the others.

- **Reduce overusing of relevant features:** It has been shown that a kernel which covers long areas of the signal, tends to build feature maps that contain a lot of redundant features. When that happens, the elements in the feature map do not represent the actual relevance of the essential points [19]. Their values are close or equivalent to each other. Dilated convolution reduces the overuses of relevant points because zeros between kernel weights nullify a portion of them.
- **Kernel length and dilatation rate:** Having evidenced that the leak could be scattered through sample points [2, 11, 38], and a kernel length should cover the leak of the intermediate values as well as the leak of the mask [21, 36]. It is not enough to avoid the overuse of relevant points by setting the first convolutional block with small kernel length. The kernel must be able to take these scattered leaks and conducts the operation; dilated convolution covers this issue.
- **Pooling stride:** Keeping the pooling stride S_p value big enough is also mandatory to have in mind. This resolution was already addressed in [40]; a pooling operation should not compromise relevant features already refined by the convolution layer.
- **Desynchronisation:** The conclusions in [40] also stresses the fact that avoiding deep architecture could have a positive impact on the presence of desynchronisation. This is an aspect that we cover by using dilated convolutions, recall that having gathered better feature maps in earlier convolutional blocks reduce the need for adding more of them. We depict this fact in the last experiment, we have used the same architecture for both ASCAD sync and ASCAD desync with $N = 100$, achieving good results. We have only changed values for each training stage and values for the kernel length as well as the dilatation rate for the first convolutional block.
- **Do not dilate too much:** Regarding the fact that we can lose too much information. It is clear that there isn't total control about how relevant the points are being nullified when using dilated convolution, so it's also possible to lose too many of them; this is a potential pitfall of using dilated convolution.

Takeaway Message:

1. For the two first points, we suggest evaluating with a small kernel length and dilatation rate values, e.g. $l_k = 7$ and $dr = 2$. Then, one should increase them iteratively finding a good trade-off. Recall, those values are related to the way the leaks are scattered in the leakage signal. A leakage analysis might help to identify the leaky points [19].
2. A recommended value for the *pooling stride* is at least the length of the pooling kernel i.e. $l_{pk} = S_p$ (as is exemplified in Fig. 1(b)).
3. In the presence of desynchronization, it is feasible to go for longer kernels, since the leakage is even more scattered. As in the first point the values to begin with heavily depend on the length of the input map, set these values having in mind that a low dilatation rate, dilates the kernel in a multiplicative way.
4. To avoid the pitfall, we recommend finding a trade-off in the number of kernels specified for the convolution layer that uses dilated convolutions. By doing so, one composes enough feature maps and preserves as many relevant features as possible. Another recommendation is to try with different kernel lengths to see the impact of changing the value. We provide one experiment to exemplify this.

Taking these considerations, the suggested architecture is summarised in Table 1 in Sect. 5.2. Note that the proposed architecture follows the rule of thumbs about the number of kernels. We use this single architecture to perform different experiments. Some values in Table 1 are fixed according to experiments, and they are set to the respective ones. To build deep learning models for the experiment we have used Python *Keras* library [9], and TensorFlow as back-end [1]. As a classification problem that has more than one class, we have use *Softmax* as the activation function for the output layer, and categorical cross-entropy [26] as the loss function. The optimiser was set to *Adam* [15] using *batch size* of 256 and a *learning rate* of 10^{-3} . Recall that these training hyperparameters and this CNN architecture are only used in the experiment with ASCAD random key version.

5 Experimental Results and Discussions

In this section, we report on the results that were achieved by using dilated convolution using synchronised and desynchronised ASCAD traces.

5.1 ASCAD Fixed Key ($N = 0$)

To compare with state-of-the-art results², we conducted experiments using different values of l_k and dr to train the CNN suggested in the latest work in [40].

² <https://github.com/gabzai/Methodology-for-efficient-CNN-architectures-in-SCA>.

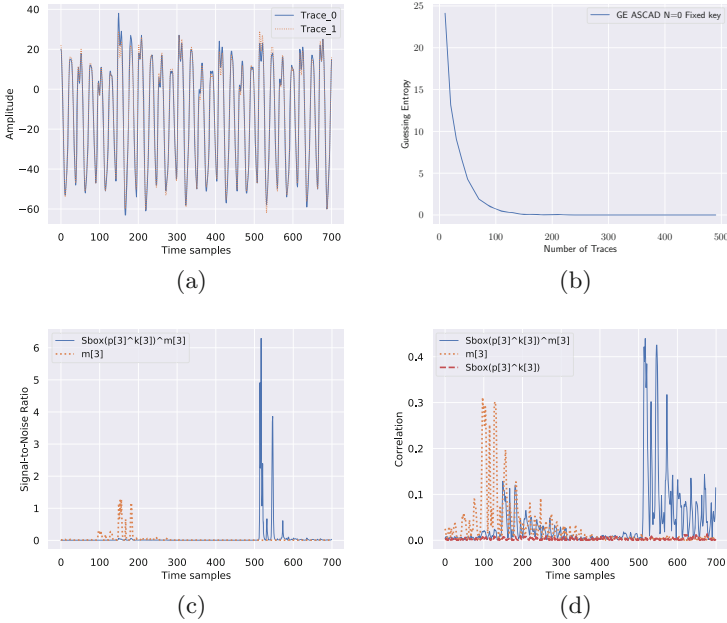


Fig. 3. (a) Two traces from ASCAD fixed key ($N = 0$); (b) GE baseline using CNN from [40]; (c) SNR of unmasked sensitive value and mask; (d) Correlation analysis

We have kept all the training hyperparameters as in that work³. Then, the experiments are conducted with the following hyperparameters of the first convolutional block: $A1 : [l_k = 1]$, $A2 : [l_k = 16, dr = 4]$, $A3 : [l_k = 16, dr = 6]$, $A4 : [l_k = 32, dr = 3]$, $A5 : [l_k = 64, dr = 2]$.

We have in Fig. 3(b) the value of GE obtained from the CNN model in [40]. Thanks to the fact that the information of the masks and the sensitive values are available in the ASCAD dataset [37], we can compute the SNR of the unmasked sensitive value and the mask. Fig. 3(c) and Fig. 3(d) show the correlation analysis, both with regard to the third byte. As we said, SNR gives us information about the points of the leakage signal that are exploitable. In ASCAD fixed key, two intervals are remarkable: $I_1 = [90, 300]$ and $I_2 = [450, 600]$. Although I_1 regarding the unmasked sensitive value is barely visible in the SNR plot, the correlation analysis emphasizes that in this area, there is some exploitable information [37]. In fact, I_1 is the interval where the two signals overlap, which represents a convenient situation because both leaks are matched.

Figure 4(a) depicts all the five attacks. Looking at these results, we argue that by having achieved the same outcome with most of the cases, the dilated convolution approach is feasible. Recall that the CNN model presented in [40] is

³ Including *One Cycle Policy* to deal with the learning rate.

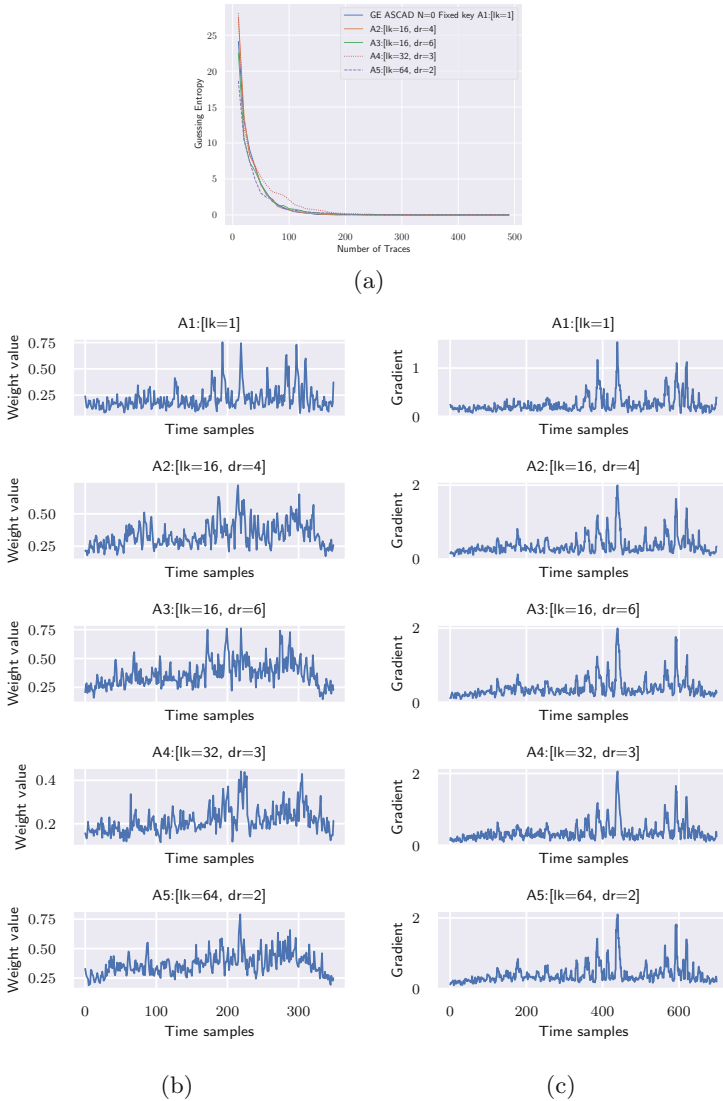


Fig. 4. (a) GE from different values of l_k and dr ; (b) Weight visualisation; (c) Gradient visualisation

one of the minimal models regard to the complexity ever presented. The model relies on the analysis and performance of having a $l_k = 1$. It indicates that the dilatation rate is a useful hyperparameter for building CNN for SCA.

Figure 4(b) and (c) depict the weight visualisation and gradient visualisation of the five models trained respectively. In general, all the models detect points of interest in similar intervals of time samples, some of those points match with

the SNR and correlation analysis in Fig. 3(c) and (d). The gradient visualisation gives us a sign that no feature information was lost from the feature learning process (convolutional part) to the classification part [40].

It’s worth mentioning that the configuration of *A4* appears to be an outlier. Its GE converges to zero in [200, 300] and not in [100, 200] like the others. The reader can interpret it as an example of finding a trade-off of the kernel length and dilatation rate.

5.2 Attack over Synchronised ASCAD Random Key (N = 0)

In the following experiments, we show a CNN with dilated convolution that performs better than previous approaches. To set a baseline to compare with Fig. 5(a) depicts the shape of two *synchronised* traces from ASCAD dataset random key (profiling_traces) and Fig. 5(b) shows a guessing entropy achieved by the model presented in [21]. As we did in the previous experiment, we have computed the SNR (and the correlation) to visualise the intervals in the leakage traces which are exploitable; the results are depicted in Fig. 5(c) and (d).

Table 1. Architecture of the CNN for experiments using ASCAD random key

Hyperparameter	Value	Additional info of values
Input shape	(1400, 1)	
Conv layers	(32, 64, 128)	SeLU, He uniform, l_{k1} and dr varies according to experiments, $l_{k2} = 25$, $l_{k3} = 3$
Regulatization	Batch normalisation	
Pooling type	Average	($l_{pk1} = s_{p1} = 2$, $l_{pk2} = s_{p2} = 25$, $l_{pk3} = s_{p3} = 4$)
FC	3 FC layers of 15 units each	SeLU, He uniform
Dense	256 units	Softmax

Comparing Lengths and Dilatation Rate

We consider the length of the kernel that could be the best choice, to achieve an efficient and effective attack (in terms of number of traces), and at the same time to show that is not only about covering all the time samples where the leak happens. To show this in practice, we performed 4 experiments under the next combinations of values; $l_k = 1$, $l_k = 32$, $l_k = 64$, $l_k = 32$, $dr = 2$ with the deep learning architecture as presented in Table 1. In Table 2, a summary of the values used in the training stage is presented.

During training, 75 *epochs* were used in all the experiments. However, when we had found out the presence of overfitting or underfitting, we adjusted these values, until we notice almost the same tendency in the loss and validation loss for all of them. Besides, as those two metrics have been shown to be not

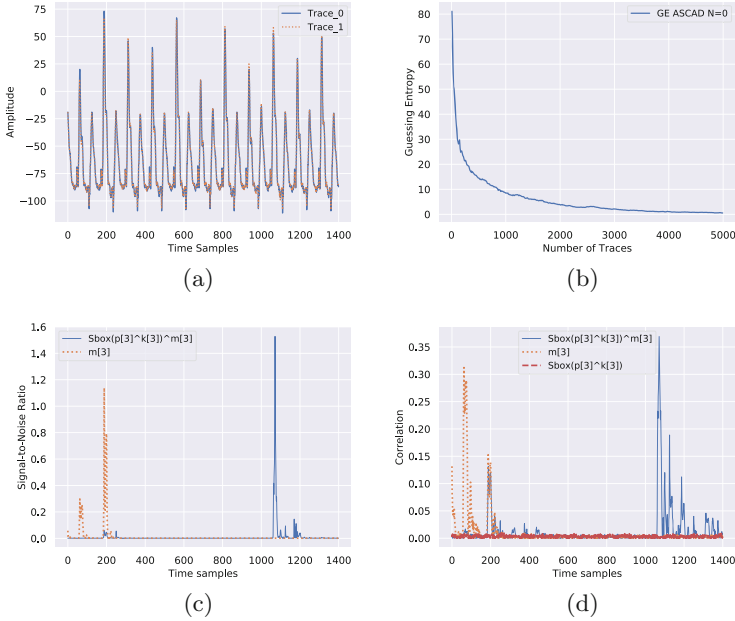


Fig. 5. (a) Two traces from ASCAD random key ($N = 0$); (b) GE baseline using CNN from [21]; (c) SNR of unmasked sensitive value and mask; (d) Correlation analysis

reliable as a side-channel metric [31], also a cross-validation setup was used. All four results are depicted in Fig. 6(a). Our goal is to show how a dilated convolution outperforms the attack effectiveness. So, the comparison was using the same architecture with values of kernel lengths suggested in the state of the art publications.

Note how the GE result of $l_k = 32$ and $l_k = 64$ are worse in approximating to the baseline. It might be caused for the accumulation of irrelevant features. In contrast, observe how the result with $l_k = 32, dr = 2$ outperform the GE. Note, the length of the kernel is practically the same as in the third one, but

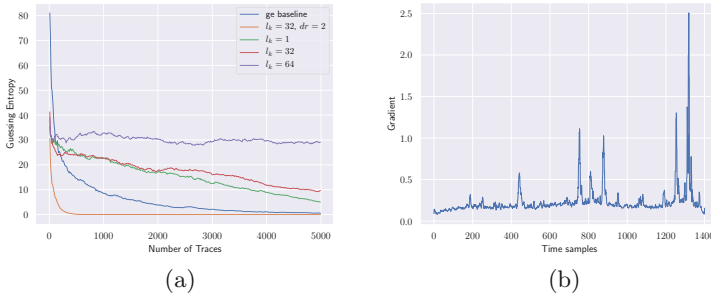


Fig. 6. (a) Guessing entropy of the four experiments; (b) Gradient visualisation result of $l_k = 32, dr = 2$

here we include less irrelevant and redundant features because of the dilatation. Figure 6(b) shows the gradient visualisation of the latter result; the relevant time samples match with the SNR in Fig. 5(c).

Table 2. Values of the training stage for synchronised traces

Parameter	Value
Number of profiling traces	45000
Number of validation traces	5000
Epochs	75

Mimic Dilated Convolution with Stride Values

In this experiment, we show the achieved performance when the behaviour of the dilated convolution is imitated with small kernel sizes, and stride values that allow skipping some features. The GE of this experiment is depicted in Fig. 7(a).

The result tells us, that although the efficiency achieved by doing this imitation is less than using dilated convolutions, kernel length of 1 with a stride value of 3 tends towards a successful attack⁴, demonstrating the effect of taking many times the same features. By comparing with dilated convolution, we demonstrate that reducing redundant and irrelevant features is not the total answer. In some cases, the efficiency could improve if the evaluator considers the fact that features in the input map might present long-range dependencies, i.e. the scattered leakage of the mask and intermediate values, so a dilated convolution comes to outperform the results of the evaluation. By comparing the gradient results in Fig. 6(b), and Fig. 7(c)–(d) one can see how efficiently each model performs the feature selection. Each one of them matches respectively with its GE.

Different Lengths with Same Dilatation Rate

Dilatation rate is also under the heuristic nature of deep learning, and it must be included in the tuning process of the deep learning architecture. Although dilated convolution discards features being or not irrelevant, there is no way (at least at the moment) to say that it is performing feature engineering, the feature selection is still not under the control of the evaluator or designer. The experiment, whose result is depicted in Fig. 7(b), was conducted to show that by using the approach of dilated convolution, one could also damage the performance. The kernel is taking not enough relevant features, or a lot of relevant features are being nullified in the interval covers by the kernel.

As we mentioned in Sect. 4, it is recommended to apply a longer kernel or increase the number of kernels, a trade-off should be found over these two hyperparameters. Keep in mind that by increasing the number of kernels in a

⁴ As well as kernel length of 3 with stride value of 6.

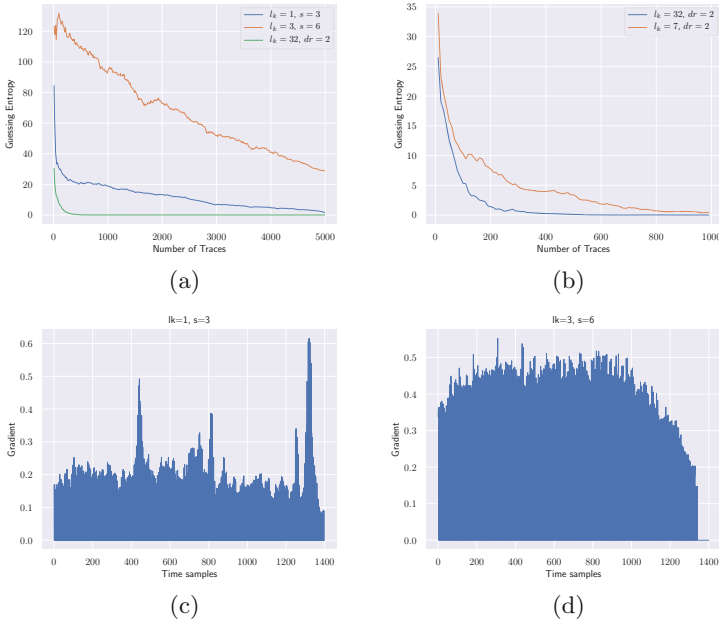


Fig. 7. (a) GE of examples that mimic dilated convolution; (b) Effect over GE with same dilatation rate and different lengths; (c) Gradient visualisation result of $l_k = 1, s = 3$; (d) Gradient visualisation result of $l_k = 3, s = 6$

convolutional block, the next one should follow the rule of thumb, i.e. increase the number of kernels by the power of 2. In Fig. 7(b) we have the result of having two dilated kernel whose lengths are considerably different in terms of dilatation rate, i.e. using a dilatation rate of 2 with a kernel size of 7, its effective size becomes 14, whilst the effective size of a kernel of 32 becomes 64. As the reader can see, different performances are achieved.

5.3 Attack over Desynchronised ASCAD Random Key (N = 100)

The last experiment considers the impact of having desynchronised traces. For this, we have used the same architecture as in the previous experiments. Figure 8 shows four combinations of values, a kernel length of 64 with a dilatation rate of 3 being the best where these were the only changes in the architecture. It is demonstrated that dilated convolution can bypass desynchronisation, reaching a considerable good performance. Mention that by having used the same architecture as in the experiment without it, we have evidenced that dilated convolution also reduces the complexity of the deep learning model. Over certain circumstances, a convolutional block that uses dilated convolution composes better feature maps. Those lead to a better characterisation of the leakage traces. So, there is no need to add more layers to the architecture.

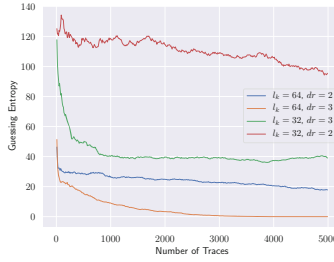


Fig. 8. Guessing entropy over desynchronised ASCAD traces ($N = 100$)

Table 3. Values of the training stage for desynchronised traces ($N = 100$)

Parameter	Value
Number of profiling traces	55000
Number of validation traces	7000
Epochs	100

Recall that to attack desynchronised signals successfully is trickier. Even though the same architecture was used, we did a few changes in the kernel length and dilatation rate, as well as changes in the training values. Table 3 summarises the latter. As we said, to deal with the effect of the desynchronisation, a trade-off of kernel length and dilatation rate must be found. Being advantageous is the fact that one can find an architecture that is useful for different scenarios.

These values make sense; longer kernel is required to combine the leaks cause relevant points between traces are more scattered when they are desynchronised, the same reason why the kernel needs to be more dilated. One could argue that if this is the case, an even simpler architecture could be found for the previous experiments. While we do not question that statement, being in a scenario where the evaluator must address synchronised and desynchronised tests, he could rely on the fact that the same architecture could achieve suitable results in both.

6 Conclusions and Perspectives

In this paper, we have used dilated convolution to build up a CNN. The results show that by using this type of convolutions, a boosting effect over the performance of a deep learning-based side-channel attack is achieved. The arguments that support our theory are taken from the already addressed aspect about decreasing the redundancy of relevant points of the input signal. These points are used for composing feature maps in the convolutional blocks of a CNN. Having a useful feature map that characterises well enough the input signal, leads to a reduction of the number of convolutional blocks in the architecture, which directly represents a reduction of the deep learning model complexity.

The potential of dilated convolutions is in the capability to inflate the kernel, covering wider areas than normal convolutions. The leakage in the trace could imply samples that are separated from different amounts of samples in between, more if the leakage signals are not synchronised. This capability is not presented in large kernels as well as small kernels with a stride value big enough, that allows them to avoid points between convolution operations. The lack of this in normal convolutions has the opposite effect, causing a negative impact on not being able to reach the performance of the dilated convolution.

From the evaluator's perspective, it has been shown that a single dilated convolutional-based model could reach enough performance boost to address the requirement of a test in both synchronised and desynchronised scenarios. However, one should bear in mind that small changes in the hyperparameter are still required. Dilated convolutions have demonstrated to be a hyperparameter that could lead to new CNN architectures which increase the threat of profiled attacks. In future works, we will systematically exploit the effect of using this approach for building deep learning models. Also, we will study other architectures that may present different behaviour when this alteration is applied, as well as study more complex data, i.e. 32-bits platforms, covering scenarios where the noise becomes an important factor when evaluating implementations of cryptographic algorithms.

Concerning the way, we could protect the latter; we are aiming for evaluating a combination of countermeasures, i.e. higher level of masking and hiding. Since hiding countermeasures try to stabilise the power consumption, the learning algorithm will detect fewer variations; this is going to impact its classification score, which could be a way to affect the performance of the attack.

References

1. Abadi, M., et al.: TensorFlow: large-scale machine learning on heterogeneous systems, software available from tensorflow.org (2015). <https://www.tensorflow.org/>
2. Belgarric, P., et al.: Time-frequency analysis for second-order attacks. In: Francillon, A., Rohatgi, P. (eds.) CARDIS 2013. LNCS, vol. 8419, pp. 108–122. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08302-5_8
3. Blömer, J., Guajardo, J., Krummel, V.: Provably secure masking of AES. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 69–83. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30564-4_5
4. Cagli, E., Dumas, C., Prouff, E.: Convolutional neural networks with data augmentation against jitter-based countermeasures. In: Fischer, W., Homma, N. (eds.) CHES 2017. LNCS, vol. 10529, pp. 45–68. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66787-4_3
5. Cagli, E., Dumas, C., Prouff, E.: Kernel discriminant analysis for information extraction in the presence of masking. In: Lemke-Rust, K., Tunstall, M. (eds.) CARDIS 2016. LNCS, vol. 10146, pp. 1–22. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54669-8_1
6. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Kaliski, B.S., Koç, K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36400-5_3

7. Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Trans. Pattern Anal. Mach. Intell.* **40**(4), 834–848 (2018)
8. Choi, K., Fazekas, G., Sandler, M., Cho, K.: Convolutional recurrent neural networks for music classification. In: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 2392–2396. IEEE (2017)
9. Chollet, F., et al.: Keras (2015). <https://keras.io>
10. Choudary, M.O., Kuhn, M.G.: Efficient, portable template attacks. *IEEE Trans. Inf. Forensics Secur.* **13**(2), 490–501 (2018)
11. Coron, J.S., Prouff, E., Rivain, M., Roche, T.: Higher-order side channel security and mask refreshing. In: Moriai, S. (ed.) *Fast Software Encryption*, pp. 410–424. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43933-3_21
12. Daemen, J., Rijmen, V.: *The Design of Rijndael*. Springer, Heidelberg (2002). <https://doi.org/10.1007/978-3-662-04722-4>
13. Dumoulin, V., Visin, F.: A guide to convolution arithmetic for deep learning. arXiv preprint [arXiv:1603.07285](https://arxiv.org/abs/1603.07285) (2016)
14. Fan, G., Zhou, Y., Zhang, H., Feng, D.: How to choose interesting points for template attacks more effectively? In: Yung, M., Zhu, L., Yang, Y. (eds.) *INTRUST 2014*. LNCS, vol. 9473, pp. 168–183. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27998-5_11
15. Goodfellow, I., Bengio, Y., Courville, A.: *Deep Learning*. MIT press (2016)
16. Hajra, S., Mukhopadhyay, D.: Multivariate leakage model for improving non-profiling DPA on noisy power traces. In: Lin, D., Xu, S., Yung, M. (eds.) *Inscrypt 2013*. LNCS, vol. 8567, pp. 325–342. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-12087-4_21
17. Hamaguchi, R., Fujita, A., Nemoto, K., Imaizumi, T., Hikosaka, S.: Effective use of dilated convolutions for segmenting small object instances in remote sensing imagery. In: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 1442–1450 (2018)
18. Hettwer, B., Gehrler, S., Güneysu, T.: Profiled power analysis attacks using convolutional neural networks with domain knowledge. In: *Selected Areas in Cryptography - SAC 2018–25th International Conference*, Calgary, AB, Canada, 15–17 August 2018, Revised Selected Papers, pp. 479–498 (2018)
19. Hettwer, B., Gehrler, S., Güneysu, T.: Deep neural network attribution methods for leakage analysis and symmetric key recovery. In: Paterson, K.G., Stebila, D. (eds.) *SAC 2019*. LNCS, vol. 11959, pp. 645–666. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-38471-5_26
20. 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 Cryptology ePrint Archive* **2018**, 1023 (2018)
21. Maghrebi, H.: Deep learning based side channel attacks in practice. *IACR Cryptology ePrint Archive* **2019**, 578 (2019)
22. Mangard, S., Oswald, E., Popp, T.: *Power Analysis Attacks: Revealing the Secrets of Smart Cards*, vol. 31. Springer, Boston (2008). <https://doi.org/10.1007/978-0-387-38162-6>
23. Martinasek, Z., Dzurenda, P., Malina, L.: Profiling power analysis attack based on MLP in DPA contest V4.2. In: 2016 39th International Conference on Telecommunications and Signal Processing (TSP), pp. 223–226 (2016)
24. Martinasek, Z., Zapletal, O., Vrba, K., Trasy, K.: Power analysis attack based on the MLP in DPA contest v4 (07 2015)

25. Masure, L., Dumas, C., Prouff, E.: Gradient visualization for general characterization in profiling attacks. In: Polian, I., Stöttinger, M. (eds.) *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pp. 145–167. Springer (2019). https://doi.org/10.1007/978-3-030-16350-1_9
26. Masure, L., Dumas, C., Prouff, E.: A comprehensive study of deep learning for side-channel analysis. *IACR Trans. Cryptographic Hardware Embed. Syst.* **2020**, 348–375 (2020)
27. Ng, A.Y.: Feature selection, L1 vs. L2 regularization, and rotational invariance. In: *Proceedings of the Twenty-First International Conference on Machine Learning, ICML 2004*, p. 78. Association for Computing Machinery, New York (2004)
28. van den Oord, A., et al.: WaveNet: a generative model for raw audio. In: *SSW* (2016)
29. Perin, G., Ege, B., Chmielewski, L.: Neural Network Model Assessment for Side-Channel Analysis. *IACR Cryptology ePrint Archive* **2019**, 722 (2019)
30. Picek, S., Heuser, A., Jovic, A., Batina, L., Legay, A.: The secrets of profiling for side-channel analysis: feature selection matters. *IACR Cryptology ePrint Archive* **2017**, 1110 (2017)
31. 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 Trans. Cryptogr. Hardw. Embed. Syst.* **2019**, 209–237 (2018)
32. Prouff, E., Strullu, R., Benadjila, R., Cagli, E., Canovas, C.: Study of deep learning techniques for side-channel analysis and introduction to ASCAD database. *IACR Cryptology ePrint Archive* **2018**, 53 (2018)
33. Rechberger, C., Oswald, E.: Practical template attacks. In: Lim, C.H., Yung, M. (eds.) *WISA 2004*. LNCS, vol. 3325, pp. 440–456. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31815-6_35
34. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. *CoRR* abs/1409.1556 (2014)
35. Standaert, F.-X., Malkin, T.G., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Joux, A. (ed.) *EUROCRYPT 2009*. LNCS, vol. 5479, pp. 443–461. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01001-9_26
36. Thiebeauld, H., Vasselle, A., Wurcker, A.: Second-order scatter attack. *IACR Cryptology ePrint Archive* **2019**, 345 (2019)
37. Timon, B.: Non-profiled deep learning-based side-channel attacks with sensitivity analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2019**(2), 107–131 (2019)
38. Waddle, J., Wagner, D.: Towards efficient second-order power analysis. In: Joye, M., Quisquater, J.-J. (eds.) *CHES 2004*. LNCS, vol. 3156, pp. 1–15. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28632-5_1
39. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. *CoRR* abs/1511.07122 (2016)
40. Zaid, G., Bossuet, L., Habrard, A., Venelli, A.: Methodology for efficient CNN architectures in profiling attacks. *IACR Trans. Cryptographic Hardware Embed. Syst.* **2020**(1), 1–36 (2019)