Improving machine learning based side-channel analysis: can dropout be dropped out?

Jim Kok[1]
[1]Delft University of Technology, Delft, The Netherlands
j.j.m.kok@student.tudelft.nl

Abstract

Analysing physical leakages (e.g. power consumption and electromagnetic radiation) of cryptographic devices can be used by adversaries to extract secret keys. Over the last couple of years, researchers have shown that machine learning has potential for this process. Machine learning models need to be fine-tuned to enhance key extraction performance. This paper investigates the so-called dropout hyper-parameter, which is proven to reduce overfitting in various domains (e.g. speech recognition). Dropout is examined for two different models: multilayer perceptrons and convolutional neural networks. Regarding the convolutional neural networks, two architectures are examined: one architecture used as a benchmark in various papers and a more uncomplicated one. The findings of this paper showed that adding dropout to the investigated multilayer perceptron architecture led to significant improvements, whereas the convolutional neural network architectures showed negligible improvements.

Keywords: Side-channel analysis, machine learning, overfitting, dropout

1 Introduction

Nowadays, the use of electronic devices such as smart cards continues to grow. This goes hand in hand with the discovery of security related-issues. One of them is side-channel analysis (SCA), which exploits unintentional leakages of a system (e.g. power consumption [11, 14] and electromagnetic radiation [20]). The purpose of this technique is to extract cryptographic keys from certain devices by analyzing physical leakages [7]. In [11], the authors managed to extract secret keys from almost 50 different devices in a variety of physical form factors by using power analysis.

In this paper, the main focus is on profiled SCA, in which the attacker has control over the cryptographic device. This allows the attacker to examine it and construct a profile. The construction can then be used by the attacker to

extract the secret key from the analyzed device. Machine learning algorithms are used for this goal. To be more specifically, deep learning, which is a subset of machine learning, is used. The main advantage of deep learning is that it avoids features to be classified manually [12]. This is of major importance for SCA, since defining features for SCA by hand is hard. To make this clear with an example: the power consumption for encrypting a message with a certain key differs per device, making features hard to classify for humans. This paper investigates two different machine learning models, namely convolutional neural networks (CNNs) and multilayer perceptrons (MLPs). CNNs are MLPs with a special structure: they consider dimensionality of the input (multi-dimensional), whereas MLP does not (1 dimensional).

The extraction process mentioned previously is not yet optimal, which implies that machine learning algorithms need to be fine-tuned. However, fine-tuning is a difficult and time-consuming process. On top of that, it is context-dependent. Nevertheless, it is of importance, since it reduced the number of traces required for disclosure of the secret key. Fine-tuning can be done by scrutinizing so-called hyper-parameters (e.g. the number of layers, loss, batch size and learning rate). The authors in [2] have shown promising results regarding the fine-tuning of hyper-parameters. However, further research is required for the enhancement of machine learning models (e.g. investigation of hyper-parameters in depth). This paper addresses the process of enhancement, by examining the dropout rate in depth, for the two different models mentioned previously. This will be done by firstly examining the problem and background in detail. Secondly, the setup of the research will be described. Thirdly, the results will be provided and discussed. Finally, the conclusion and possible future work will be given.

2 Problem Description

This section elaborates on the problem described in the Introduction. To be more specific, explanation on the concept of overfitting and how to avoid it are provided. Finally, necessary background information is given.

2.1 Overfitting

Overfitting [5] is one major issue regarding machine learning. Overfitting occurs when a model starts to memorize the inputs, instead of learning from them. Several methods have been introduced to reduce this: one of them is called dropout [21]. “The term “dropout” refers to removing units (hidden and visible) from a neural network, along with all its incoming and outgoing edges. The choice of which units to drop is random” [28, p. 201]. This technique reduces the effect of certain neurons to dominate, as they can be excluded at random giving more priority to others.

In previous research, there has been made use of dropout. The authors in [10] have used this regularization technique, however, they never investigated the

effects and optimal value in depth. Furthermore, in [19], the authors performed hyper-parameter tuning only for a limited range of dropout values for the DPAv4 database [3]. To the best of our knowledge, no extensive research has been performed to investigate the effects of this hyper-parameter on side-channel analysis. Nonetheless, dropout is proven to reduce overfitting in various contexts (e.g. speech [16] and handwriting [13] recognition). Concretely, "it reduces the complex co-adaptations of neurons and forces them to learn more robust features" [1, p. 8].

Dropout is a known regularization technique for machine learning. The use of it in a side-channel analysis based environment needs to be further investigated. This study hypothesises that dropout can indeed help machine learning based SCA. To make this explicit: overfitting is a common problem in machine learning, therefore this paper presumes that machine learning based SCA is prone to this concept. Finally, this is tested by analysing the results of the ASCAD database, introduced in Section 4.1, applied to the CNN and MLP architectures.

2.2 Hyper-parameter tuning and loss minimization

As mentioned in the Introduction, machine learning structures need to be fine-tuned to get optimal results. This process is context-dependent (e.g. optimal dropout values vary for different databases and architectures). One can distinguish two types of parameters: trainable parameters and hyper-parameters. Trainable parameters are determined during training and can not be set. Furthermore, they are inherent to the model (e.g. the weights and biases of neurons). The latter needs to be fine-tuned and can be distinguished into two categories [27]:

- Optimizer hyper-parameters, that are related to the training process (e.g. batch size, learning rate and the number of epochs).
- Model hyper-parameters, that are related to the structure of the architecture (e.g. the number of layers, nodes and filters).

When constructing a machine learning architecture, hyper-parameter tuning aims at minimizing the loss. Loss functions provide information about the performance of a network. With the help of an optimizer, loss functions are being minimized. This paper makes use of categorical cross-entropy. Categorical cross-entropy is a commonly used loss function [30] and proven to be suitable for machine learning based SCA [15].

This paper mainly focuses on fine-tuning the dropout rate, that belongs to the model hyper-parameter category. Specifically, the dropout rate will be tuned for MLP and CNN architectures. This will be done by applying an appropriate value range that gives insight into the behaviour of the examined architectures. As mentioned in Section 2.1, there has been done no extensive research regarding dropout in a SCA based context. This work aims at fulfilling this knowledge

gap. Finally, the next two sections provide background information about the two used models.

2.3 Multilayer perceptrons

Multilayer perceptrons (MLPs) [17] are a machine learning model that typically consists of the following layers: the input layer, hidden layer(s) and output layer. All layers are defined to be dense, meaning that each node in layer $n-1$ is connected to every node in layer n. Each node is assigned a weight and bias that are updated per batch size. Furthermore, the output of each node is defined by its inputs and activation function [6]. Section 2.5 provides the necessary information about the used activation functions. Finally, MLPs have shown promising results in various domains (e.g. speech emotion recognition [25] and face gender recognition [23]).

2.4 Convolutional neural networks

Convolutional neural networks (CNN) are a machine learning model that is suitable to handle multi-dimensional inputs by applying convolution and pooling. Convolutional layers output feature maps, which are the results of the convolution operation to the input by sliding a set of filters along the traces [24]. Convolutional layers act as feature extractors. For clarity, the left image in Figure 1 shows an example of the convolution operation. Furthermore, "pooling (POOL) layers are non-linear layers that reduce the spatial size in order to limit the amount of neurons, ..." [4, p. 53]. As a result, pooling layers reduce the cost of training and the probability of overfitting [1]. Several types of pooling layers exist (e.g. average and max-pooling). The right image in Figure 1 shows an example of max-pooling. This paper solely makes use of average-pooling.

Commonly, a convolutional layer is followed by a pooling layer and CNNs consist of multiple of the two. After the input is applied to the convolutional and pooling layers, it is fed to the neural network, consisting of fully-connected layers. Finally, CNNs are shown to be successful in in various domains (e.g. face recognition [26] and text detection [8]).

2.5 Activation functions

Activation functions non-linearly map the input of a certain node to an output. These functions allow machine learning models to improve the classification. Several activation functions exist, of which some are provided below.

This paper makes use of the 'Softmax', 'ReLU' and 'SELU' activation function. The 'Softmax' function is used to calculate the probabilities of the possible keys in the final layer. Furthermore, the output of Rectified Linear Units (ReLUs) is defined as the positive part of its input. Whenever the input is negative, the output is defined to be equal to zero. Finally, the formulas for the Scaled

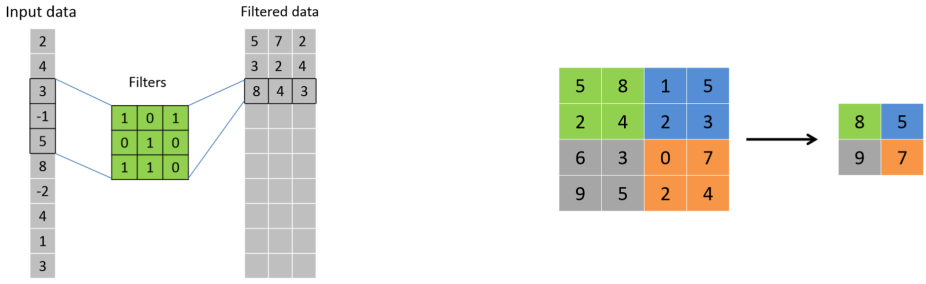


Figure 1: Example of the convolution operation (left), where the number and size of filters are set to 3 and the stride to 1. The image on the right shows an example of the max-pooling operation. This example has a filter of size 2 by 2, with a stride of 2. The image is retrieved from [24].

Exponential Linear Unit (SELU) and the previously mentioned are given by the following equations:

$$softmax(a)_j = \frac{e^{a_j}}{\sum_{i=1}^{I} e^{a_i}} \tag{1}$$

$$ReLU(a)_j = max(0, a_j), \tag{2}$$

$$SELU(a)_j = \lambda \begin{cases} a_j, & \text{if } x > 0 \\ \alpha e^x - \alpha, & \text{if } x \leq 0 \end{cases} \quad \text{for } j = 1, 2, ..., I, \tag{3}$$

with α and λ being constants and I the number of nodes in the layer.

2.6 Leakage model

As mentioned in the Introduction, SCA exploits unintentional leakages of a system. In machine learning based SCA, the objective of a leakage model is to determine the secret key. The leakage model $Y(T, k)$ is dependent on the used key k and plaintexts T. During training, the leakage model is constructed by giving traces, included with their known keys, as input to the neural network. In the attacking phase, one can make predictions of the correct key k^*, by analyzing the leakage model $Y(T, k)$ for each $k \in K$.

Since it is assumed that physical leakages are uniquely defined for each $k \in K$, leakage models are bijective functions. When making use of multiple samples to determine the correct key the log-likelihood principle is used [10]:

$$g_i = \sum_{j=1}^{A} \log p_{ij} \quad \text{for } i = 1, 2, ..., |K|, \tag{4}$$

where g_i is the summed log probability of the ith key over the attacking samples. Furthermore, with A being the number of attacking samples and p_{ij} the probability of the ith key for the jth sample.

3 Enhancing side-channel analysis

This section provides information about the experimental setup. Furthermore, the main contribution of this paper is outlined. Finally, the evaluation metric used to assess the results of the experiments is described.

3.1 Experimental work

This paper aims to provide insight into the influence of dropout on machine learning based side-channel analysis. Since time and computational resources are a limiting factor for this research, experiments are only performed on the ASCAD database, that is introduced in Section 4.1. However, as mentioned in Section 7, to strengthen experimental results various databases need to be investigated. For the first experiment a relatively simple architecture, described in [27], is examined. Furthermore, to possibly substantiate the research question, two architectures, introduced by [2], are examined by applying a range of dropout values.

As mentioned in Section 2.1, dropout reduces the effect of certain neurons to dominate. The main problem with overfitting is that the model becomes too complex. By dropping out neurons, at random, this technique reduces the complexity of it. The contribution of this paper is to provide insights into whether machine learning based SCA can be improved by applying dropout. To be more concrete: three publicly available architectures, which act as benchmarks in various papers, are examined in depth. Previously, it has been proved that machine learning in certain areas (e.g. speech recognition) can indeed be improved by dropout [29]. However, this hyper-parameter has never been studied in depth for SCA.

Guessing entropy (GE) is used as an evaluation metric for the experiments. GE is defined as the average position of the correct key when the set of all possible keys are ranked in descending order. To make this clear: the most likely key takes the first and the least likely key the last position. Guessing Entropy has been proposed to act as a standard to assess side-channel attacks [22]. The positions of the keys are determined by randomly picked samples. Furthermore, the log-likelihood principle, explained in Section 2.6, is used to calculate the probabilities of the keys over multiple samples. Finally, the guessing entropy is then defined as the average key rank of the correct key over multiple runs.

4 Experimental Setup and Results

Firstly, the database that is used for training, validating and testing is introduced. Secondly, the base algorithms used for the experiments are described. Thirdly, experimental details are given and discussed. Finally, the results are provided and observations are made.

4.1 ASCAD database

The ASCAD database provides a benchmark for evaluating machine learning algorithms for side-channel attacks [18]. This research makes use of the ASCAD database [2], which consists of 50,000 training and 10,000 test traces. Both test and training traces are labelled with the correct encryption key. Furthermore, each trace consists of 700 samples, that are related to the leakage model in Equation 5. The first experiment is conducted with 45,000 training and 5,000 validation traces, which are randomly selected from the training set. For training and validation of the second experiment, there is not been made use of validation traces. These divisions are chosen since these architectures are fine-tuned in this way, and thus leading to better results [2, 27]. Moreover, one aims at using as many traces as possible.

Regarding the leakage model, defined in Section 2.6, the ASCAD database leakage is dependent on the 3rd byte of the plaintext. Besides, an unmasked S-box is used as leakage model, which is inherent to the database:

$$Y(k) = \text{S-box}[P_3^k], \tag{5}$$

where P_3 is the third byte of the plaintext and $k \in K$ the used encryption key. This paper uses the fixed key version of the ASCAD database. Finally, for reproducibility, the ASCAD database can be downloaded from `https://www.data.gouv.fr/s/resources/ascad/20180530-163000/ASCAD_data.zip`.

4.2 Baseline algorithms

For the first experiment, a CNN architecture is used, which is known to be efficient and uncomplicated. The architecture is given in [27], and according to the authors, 3,930 times less complex than the CNN architecture used in the second experiment. It is used to test the dropout rate performance of low complexity CNN networks. Table 1a provides the hyper-parameter settings of the architecture. This architecture makes primarily use of the 'SELU' activation function. The reason for this is that this function avoids gradient related problems [27]. The required background information about activation functions can be found in Section 2.5. The complexity of this architecture is relatively low. Therefore, it's training time is considerably faster than the second CNN architecture, introduced in the next paragraph. Consequently, there is no need to reduce the number of epochs. The baseline algorithm of the first experiment is available at `https://github.com/gabzai/Methodology-for-efficient-CNN-architectures-in-SCA`.

To further test the hypothesis mentioned in Section 2.1, a second experiment is conducted, in which a more complex CNN architecture is examined. On top of that, a MLP architecture is investigated. As mentioned in Section 2.3 and 2.4, respectively, MLPs and CNNs have shown promising results in various domains. Furthermore, both architectures are known benchmarks in numerous literature related works. Table 1b and 2 provide detailed information about the CNN and MLP architecture, respectively. In contrast to the first CNN

architecture, this architecture primarily uses the 'ReLU' activation function. According to the authors in [2], this activation function provided state-of-the-art results. Furthermore, the computation time of this function is relatively fast. Originally, for MLP one of the most optimal results are obtained when the number of epochs is set to 200. However, to reduce training time, it is set to 150. Likewise, the epochs for CNN is set to 30, whereas one of the most optimal trade-offs is acquired when the epoch hyper-parameter is set up to 100 [2]. However, as mentioned in Section 3.1, time and computational resources were a limiting factor. Since this paper aims to provide insights into the enhancement of machine learning, by applying dropout, reduction of epochs to improve training time is irrelevant. For reproducibility, the baseline algorithm of the second experiment is available at `https://github.com/ANSSI-FR/ASCAD`.

4.3 Experimental details

During the experiments, the same configurations are used as provided by Table 1 and 2. For experiment 1 and 2, respectively, 45,000 randomly chosen samples and the entire ASCAD training set (i.e. 50,000 samples) are used for training. Both experiments use the 10,000 provided test samples to determine the results, expressed in guessing entropy.

For the CNN architectures, dropout is applied to the second-last fully-connected layer. Dropout could also have been applied to the convolutional layers. However, these layers have relatively few hyper-parameters and applying dropout to it could result in too much information loss [10]. Furthermore, applying dropout to the last layer would result in a decrease in performance. The reason for this is that dropout is random, and thus could exclude the correct key prediction. Applying dropout to other fully connected layers could influence the behaviour of the network. To make this principle more clear: applying dropout to layer n also influences layers $n+1, n+2, ..., f$, with f being the final layer. The authors in [13] showed that different positions of dropout could improve Arabic handwriting recognition, for a special type of Recurrent Neural Network. However, for machine learning based SCA further research is required, as being stated in Conclusions and Future Work. Likewise, for MLP, dropout is applied to the second-last fully connected layer.

Guessing entropy, explained in Section 3.1, is used as a performance metric for both experiments. This performance metric averages the rank over a certain number of runs. For these experiments, it is averaged over 100 runs. Using more runs lead to more representative results, however, a trade-off between speed and representativeness needs to be made. When rank is being used as a performance metric, which is the case in [2], results are not averaged. This could lead to unstable outcomes. Finally, speeding up the process was done by making use of the Geforce GTX 1060 graphics processing unit (GPU).

Convolution			Pooling		Fully-connected layers		Training		
Size	Padding	Channel size (per layer)	Type	Size	Number of nodes (per layer)	Dropout {Tested values}	Epochs	Batch size	Learning rate
11	'same'	(4)	'average'	2	(10, 10)	$\{0, 0.05, 0.06, 0.075, 0.1, 0.2\}$	50	50	OneCycleLR

(a)

Convolution			Pooling		Fully-connected layers		Training		
Size	Padding	Channel size (per layer)	Type	Size	Number of nodes (per layer)	Dropout {Tested values}	Epochs	Batch size	Learning rate
11	'same'	(64,128,256, 512,512)	'average'	2	(4096, 4096)	$\{0, 0.1, 0.2, 0.3, 0.4, 0.6\}$	30	200	0.0001

(b)

Table 1: Configuration of the CNN architectures used for first (a) and second (b) experiment. According to the authors of [27], the learning rate is one of the most difficult hyper-parameters to tune. Therefore, they made use of the one cycle learning rate (a). Before training, it is set to 5×10^{-3} and is updated each epoch. Both CNN architectures use the flattening operation after the convolutional layers, which reduces the obtained multi-dimensional input to a 1-dimensional vector. This vector is then fed to the fully-connected layers. Furthermore, (a) applies batch normalization after the first convolutional layer. This normalizes the input and speeds up the learning process [9]. The first CNN architecture (a) makes use of the 'SELU' activation function, whereas (b) uses the 'ReLU' activation function for each but final layer. Similarly, they use the 'Softmax' activation function for the final layer. For more information about activation functions see Section 2.5. Besides, (a) uses the 'he uniform' kernel initializer, for improved results [27]. The "ADAM" and "RMSprop" optimizers are used by (a) and (b), respectively. These optimizers are used since they provided the best results out of the tested ones. Finally, both architectures use the categorical cross-entropy loss function. Section 2.2 provides more information about loss functions.

Fully-connected layers		Training		
Number of nodes (per layer)	Dropout {Tested values}	Epochs	Batch size	Learning rate
(700, 200, 200, 200, 200, 200, 256)	$\{0, 0.2, 0.4, 0.5, 0.6, 0, 8\}$	150	100	0.0001

Table 2: Configuration of MLP architecture used for the second experiment. Each layer uses the 'ReLU' activation function, except the final layer. The final layer uses the 'Softmax' activation function. Finally, the categorical cross-entropy loss function is used in combination with the 'RMSprop' optimizer.

4.4 Results

This subsection provides the results of the MLP and CNN architectures, described in Section 4.2 and specified in Table 1 and 2. Different dropout rates are applied to these architectures. Finally, for each of them, conclusions are logically drawn from the results.

4.4.1 Results MLP

The results for the ASCAD database applied to the MLP architecture, introduced in Section 4.2, are shown in Figure 2. Based on the guessing entropy for different dropout rates it becomes clear that the best results are not obtained when the dropout is set to 0. Especially, for the first 2,000 traces, dropout rates of 0,2 and 0,4 are significantly outperforming no dropout. The tested values higher than 0,4 drastically decrease the performance of the architecture. For this architecture, dropout improves the performance. Dropout can thus not be dropped out, referring back to the title: "can dropout be dropped out?". Note that these conclusions are exclusively drawn for this architecture. As mentioned in the Conclusions and Future Work section, further research is required for generalization.

There seems to be a threshold for excessive relevant information loss within the network. Since there is a significant shift in performance between 0,4 and 0,6 dropout the threshold lies within this interval. Initially, for the dropout rate a step size of 0,2 is taken. The dropout rate of 0,5 is added after noticing tremendous performance loss for values higher than 0,4. The chosen values intend to give insights into the threshold and performance of the architecture.

4.4.2 Results CNN

For CNN, two architectures, that among other things vary in complexity, are investigated. For the first architecture (i.e., relatively, the less complex and more efficient one), tests are performed on dropout values {0, 0.05, 0.06, 0.075, 0.1, 0,2}. At first, a step size of 0,1 is chosen for the dropout rate. However, perceiving deterioration of performance for 0,1 and 0,2 dropout, led to the emphasizing of values between 0 and 0,1. The results are shown in Figure 3a and 3b, which show GE up to 300 and 10,000, respectively. The former's purpose is to give better insight into the behaviour of the architecture for lower dropout and number of traces. Interestingly, each tested dropout value shows no to little improvements. Therefore, it is hypothesized that this architecture mainly contains relevant information, before dropout occurs. This is based on the fact that this architecture is 3,930 less complex than the second architecture, as stated in Section 4.2. In the next paragraph, this hypothesis is tested by scrutinizing the second architecture, which is more complex, and thus more likely to contain less relevant information before dropout occurs.

For the second architecture (i.e., relatively, the more complex one) the following dropout rates are applied: {0, 0.1, 0.2, 0.3, 0.4, 0.6} and are visible in Figure 4. Remarkably, the results for this architecture show, likewise, no

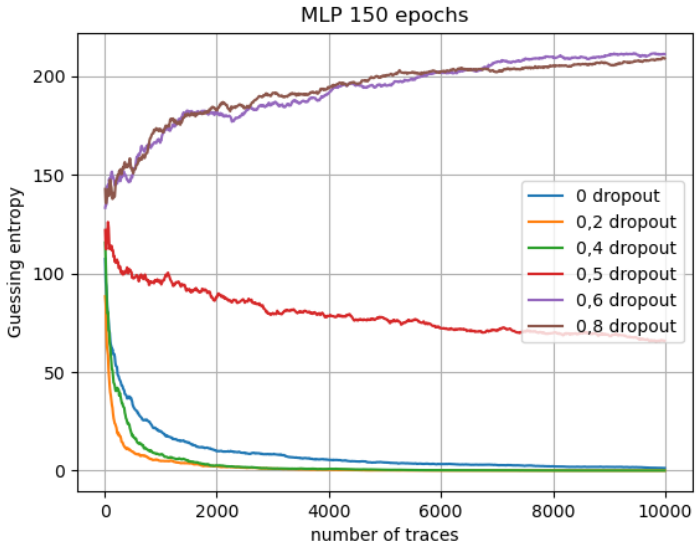


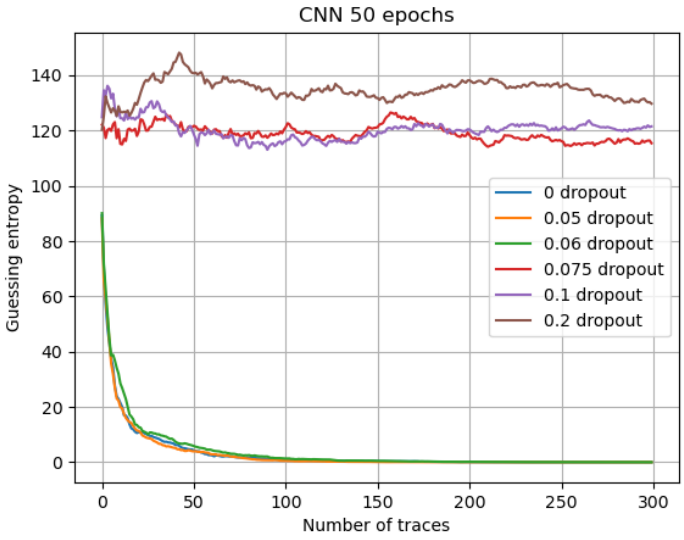
Figure 2: MLP first architecture: guessing entropy results for dropout $\{0, 0.2, 0.4, 0.5, 0.6, 0.8\}$. For details about this architecture see Section 4.2 and Table 2

improvements. Therefore, the train of thoughts that this architecture is more complex and thus can be improved by dropout is refuted. Furthermore, for the dropout rate, a step size of 0,1 is taken. After observing the performance loss for 0,4 dropout, a final dropout value of 0,6 is added. This value is added to give insights into the behaviour of relatively high dropout rates.

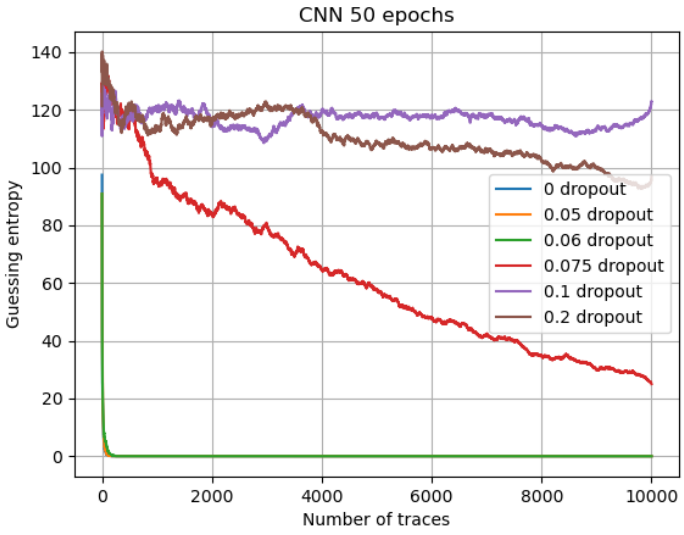
As becomes clear from the first two paragraphs, for each architecture the GE starts to increase rapidly between two certain dropout values. It seems that a threshold for this is inherent to each architecture. To be more precise: the threshold for the more complex architecture is considerably higher than the less complex one. All in all, both CNN architectures show minor to no improvements regarding dropout. However, noticeable, the less complex architecture is more sensitive to changes in the dropout rate. Therefore, smaller step sizes need to be taken for this architecture.

5 Responsible Research

This research aims at improving machine learning based side-channel analysis by making use of dropout. Using machine learning to extract cryptographic keys of devices brings in ethical related aspects. One of them is which scenario allows the use of it. To answer this question briefly: it should only be used for educational purposes. Furthermore, should finding that improve machine



(a)



(b)

Figure 3: CNN first architecture for 300 traces (a) and 10,000 traces (b): guessing entropy results for dropout $\{0, 0.05, 0.06, 0.075, 0.1, 0, 2\}$. For details about this architecture see Section 4.2 and Table 1a.

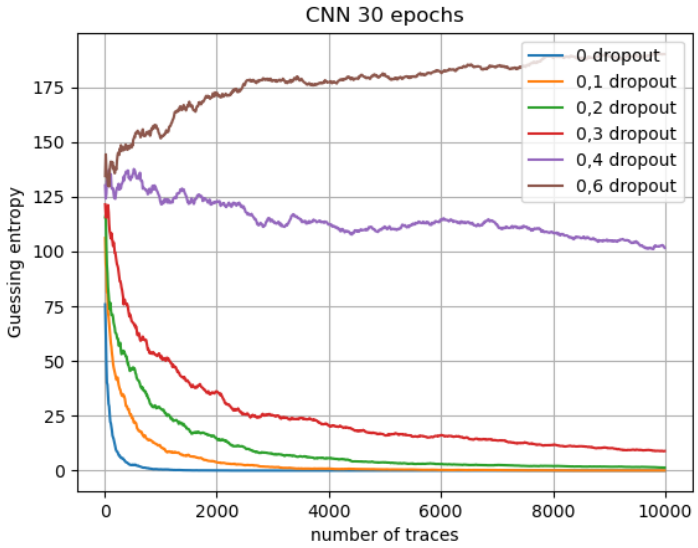


Figure 4: CNN second architecture: guessing entropy results for dropout $\{0, 0.1, 0.2, 0.3, 0.4, 0.6\}$. For details about this architecture see Section 4.2 and Table 1b.

learning based SCA be made public? Making these finding public can lead to abuse. Currently, only a few limited protection measures are available for SCA. This makes it easier for adversaries to take advantage of it.

Regarding reproducibility, the three architectures used for this experiment are described in detail. Furthermore, based on the information in this paper, any skilled readers should be able to reproduce the described experiments. Besides, the URLs and papers referring to the architectures are provided.

6 Discussion

Previously, there has been done minor research regarding dropout rates. Currently, only a limited range of values have been tested to tune a CNN architecture for the DPA contest v4 dataset [19]. Initially, this paper focused on one CNN architecture, which is introduced in [27]. This architecture is a benchmark which is known to be efficient and simple. After noticing that this architecture showed no to minor improvements, two additional architectures were investigated. To be more specific: one CNN and MLP architecture, which are described in [2]. This paper hypothesised that the lack of improvements in the first experiment was due to the high efficiency and low complexity of the architecture. Therefore, the additional CNN architecture was investigated. Furthermore, the MLP architecture was chosen to gain insights into a different machine learning

model.

From this research, it became clear that the dropout hyper-parameter has to be carefully fine-tuned. Interestingly, the results for the CNN architectures showed minor to no improvements. Whereas, results for the MLP architecture showed significant improvements. This could indicate that CNNs are less prone to overfitting, in comparison to the MLPs, in a SCA based context. Finally, worthwhile mentioning, is that all architectures showed a rapid decline in performance for two consecutive dropout values. For the less complex architecture, this threshold was lower than the more complex one. This could indicate that more complex architectures are more prone to overfitting.

7 Conclusions and Future Work

This paper examines machine learning based side-channel analysis by considering two models: convolutional neural networks and multilayer perceptrons. Previously, these models have shown potential for this purpose and have been scrutinized by various researchers. This study aims to improve these models by applying a regularization technique called dropout. To the best of our knowledge, no extensive research has been done regarding the dropout rate in a SCA based environment. Therefore, the research question is whether dropout can improve machine learning based SCA.

In total, three architecture are investigated: 2 CNN architectures and 1 MLP architecture. These architectures already have been fine-tuned for the ASCAD database. Furthermore, they are publicly available and used as a benchmark in the SCA community. For the two CNN architectures, the complexity differs, such that the influence of it and dropout can be scrutinized. To measure performance guessing entropy is used as a metric.

In conclusion, results showed that dropout is suitable to improve the MLP architecture. Whereas, dropout showed minor to no improvement for the 2 CNN architectures. This does not imply that dropout is not capable of improving CNNs. Further research is required to make generalizations. Interestingly, there seemed to be a threshold for the divergence of performance. To be more elaborate: each experiment showed a relatively major decline in performance for two consecutive dropout values. The threshold for the less complex CNN architecture is significantly lower than the more complex one. Like being said in the Discussion section, this could indicate that more complex architectures are more prone to overfitting.

Finally, to strengthen the experimental results, further research is required. This could include:

1. Investigation of dropout rates for more architectures. This paper focuses on three architectures, the behaviour of other ones could result in different findings. Like being said in the second paragraph, more of them need to be examined to make generalizations.

2. The influence of different positions where dropout could be applied (e.g. after the first layer). This paper only applies dropout to the second-last dense layer, as explained in Section 4.3. Furthermore, this research could include the effects of dropout applied to multiple layers.

3. The influence of different datasets on dropout. This paper solely focuses on the ASCAD database. Simply, the reason for this is that the research is done in a limited time frame. Additionally, the computational resources were a limiting factor.

References

[1] Belhassen Bayar and Matthew C Stamm. A deep learning approach to universal image manipulation detection using a new convolutional layer. In *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, pages 5–10, 2016.

[2] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. Deep learning for side-channel analysis and introduction to ascad database. *Journal of Cryptographic Engineering*, pages 1–26, 2019.

[3] Shivam Bhasin, Nicolas Bruneau, Jean-Luc Danger, Sylvain Guilley, and Zakaria Najm. Analysis and improvements of the dpa contest v4 implementation. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 201–218. Springer, 2014.

[4] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 45–68. Springer, 2017.

[5] Tom Dietterich. Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR)*, 27(3):326–327, 1995.

[6] Joey Green, Tilo Burghardt, and Elisabeth Oswald. Not a free lunch but a cheap lunch: Experimental results for training many neural nets.

[7] Gabriel Hospodar, Benedikt Gierlichs, Elke De Mulder, Ingrid Verbauwhede, and Joos Vandewalle. Machine learning in side-channel analysis: a first study. *Journal of Cryptographic Engineering*, 1(4):293, 2011.

[8] Weilin Huang, Yu Qiao, and Xiaoou Tang. Robust scene text detection with convolution neural network induced mser trees. In *European conference on computer vision*, pages 497–511. Springer, 2014.

[9] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[10] Jaehun Kim, Stjepan Picek, Annelie Heuser, Shivam Bhasin, and Alan Hanjalic. Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 148–179, 2019.

[11] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Annual International Cryptology Conference*, pages 388–397. Springer, 1999.

[12] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[13] Rania Maalej and Monji Kherallah. Improving mdlstm for offline arabic handwriting recognition using dropout at different positions. In *International conference on artificial neural networks*, pages 431–438. Springer, 2016.

[14] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks: Revealing the secrets of smart cards*, volume 31. Springer Science & Business Media, 2008.

[15] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. A comprehensive study of deep learning for side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 348–375, 2020.

[16] Yajie Miao and Florian Metze. Improving low-resource cd-dnn-hmm using dropout and multilingual dnn training. In *Interspeech*, volume 13, pages 2237–2241, 2013.

[17] Gaurang Panchal, Amit Ganatra, YP Kosta, and Devyani Panchal. Behaviour analysis of multilayer perceptrons with multiple hidden neurons and hidden layers. *International Journal of Computer Theory and Engineering*, 3(2):332–337, 2011.

[18] Christophe Pfeifer and Patrick Haddad. Spread: a new layer for profiled deep-learning side-channel attacks. *IACR Cryptology ePrint Archive*, 2018(880), 2018.

[19] Stjepan Picek, Ioannis Petros Samiotis, Jaehun Kim, Annelie Heuser, Shivam Bhasin, and Axel Legay. On the performance of convolutional neural networks for side-channel analysis. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 157–176. Springer, 2018.

[20] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In *International Conference on Research in Smart Cards*, pages 200–210. Springer, 2001.

[21] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

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

[23] Sudeep D Thepade and Deepa Abin. Face gender recognition using multi layer perceptron with otsu segmentation. In *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pages 1–5. IEEE, 2018.

[24] Benjamin Timon. Non-profiled deep learning-based side-channel attacks with sensitivity analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 107–131, 2019.

[25] Bagus Tris Atmaja and Masato Akagi. Deep multilayer perceptrons for dimensional speech emotion recognition. *arXiv*, pages arXiv–2004, 2020.

[26] Kewen Yan, Shaohui Huang, Yaoxian Song, Wei Liu, and Neng Fan. Face recognition based on convolution neural network. In *2017 36th Chinese Control Conference (CCC)*, pages 4077–4081. IEEE, 2017.

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

[28] Ming Zeng, Le T Nguyen, Bo Yu, Ole J Mengshoel, Jiang Zhu, Pang Wu, and Joy Zhang. Convolutional neural networks for human activity recognition using mobile sensors. In *6th International Conference on Mobile Computing, Applications and Services*, pages 197–205. IEEE, 2014.

[29] Shiliang Zhang, Ming Lei, Bin Ma, and Lei Xie. Robust audio-visual speech recognition using bimodal dfsmn with multi-condition training and dropout regularization. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6570–6574. IEEE, 2019.

[30] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in neural information processing systems*, pages 8778–8788, 2018.