Impact of Data Pre-Processing Techniques on Deep Learning Based Power Attacks

Aljuffri, A.A.M.; Reinbrecht, Cezar; Hamdioui, S.; Taouil, M.

DOI
10.1109/DTIS53253.2021.9505051

Publication date
2021

Document Version
Final published version

Published in
2021 16th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS)

2021 16th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS) | 978-1-6654-3654-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/DTIS53253.2021.9505051

Impact of Data Pre-Processing Techniques on Deep Learning Based Power Attacks

Abdullah Aljuffri, Cezar Reinbrecht, Said Hamdioui, Mottaqiallah Taouil
Computer Engineering
Delft University of Technology
Delft, The Netherlands
{a.a.m.aljuffri,c.r.wedigreinbrecht,s.hamdioui,m.taouil}@tudelft.nl

***Abstract*—Power-based side channel attacks (SCAs) are recognized as a powerful type of hardware attacks. Recently, attacks based on deep learning (DL) neural networks have become popular due to their high efficiency. However, even these attacks face problems when sophisticated countermeasures exist. Pre-processing the input data is an effective way to improve the performance of such neural networks. Currently, only limited research has focused on exploring pre-processing techniques for DL-based attacks. In this paper, we propose to the best of our knowledge for the first time the usage of data transformation, data concatenation and stacked auto-encoder (encoder only) as pre-processing methods. Thereafter, we compare them with the existing techniques, namely data augmentation and stacked auto-encoder techniques. Our results show that the data transformation technique achieves the best results from the evaluated methods; it improves the validation accuracy from 75% to 95% and 23% to 26% for the RSA and AES implementations, respectively.**

***Index Terms*—Side channel attacks, profiled-based attacks, deep learning, pre-proccessing techniques**

I. Introduction

The importance of cybersecurity grows more and more every year. Future projections foresee a total market value of 231.94 billion US dollars by 2021 [1]. The main reason relies on the deployment of new technologies like Internet-of-Things (IoT) and 5G communication. These technologies significantly increase the quantity of devices and their connectivity [2]. As a result, such conditions create new opportunities for cyberattacks. For example, IoT edge nodes are placed in the field which could be accessible to the public. Hence, attackers may perform power attacks on them known as Side-Channel Attacks (SCAs) [3]. SCAs can retrieve secret information of devices by just observing their physical behavior under operation (e.g., power consumption, radiation, dissipated heat). Both symmetric (i.e., systems that use same key for encryption and decryption) and asymmetric cryptography (i.e., systems that use different keys for encryption and decryption) are vulnerable against SCAs. Consequently, SCAs have been gaining great popularity and their methods have been improved continuously over time.

Paul Kocher [4] introduced the concept of Simple Power Attack (SPA) and Differential Power Attack (DPA). These methods observe the relation of the performed operation in the power traces. In a further work, a statistical method was proposed to correlate hypothetical power estimations with the real consumption. This attack is called Correlation Power Attack (CPA) [5]. In CPA, leakage models like hamming weight and hamming distance are used to correlate guessed keys with the power traces. Both models can be used to estimate the power behavior of a specific operation, for instance, the result of the first round of the popular AES cipher. On the other hand, countermeasures have been proposed to make DPA and CPA attacks less effective [6–8]. In 2002, the concept of profiled attacks was proposed by Chari et al. [9]. Their attack is referred to as Template Based Attack (TBA). TBA builds a customized power model of a device similar to the target device, and correlates the power measurements of the target device of the victim with the customized power model. TBA uses the multivariate gaussian distribution function to build the power profile. However, its mathematical complexity limits the attack accuracy [10]. To overcome this drawback, Martinasek et al. [11] proposed in 2013 the usage of machine learning to enhance the profiling attacks on AES. They used a multilayer perceptron (MLP) neural network. Initially, they obtained an 80% accuracy only, but after further optimizations reached a 100% accuracy [12] using a pre-processing technique consisting of averaging of power traces. Thereafter, Gilmore et al. [13] presented a profiled attack using MLP model that focused on symmetric cryptographic algorithms. In 2016, Maghrebi et al. [10] proposed the usage of Deep Learning in profiled Side-Channel Attacks. They successfully attacked different symmetric encryption implementations using a Convolution Neural Network (CNN). DL-based SCA improved the amount of traces required to accomplish the attack on both protected and unprotected symmetric cryptographic algorithms. In 2017, Cagli et al. [14] used data augmentation (i.e., generating new input samples to the neural network from the existing power traces) to further improve the accuracy of DL-based SCAs. They proposed two strategies to perform data augmentation, one by using random re-alignment of existing traces and the other by adding random noise. Note that profiled based side-channel attacks are less powerful when countermeasures like power obfuscation and randomized keys are used [15]. Some of the above deep learning based attacks used pre-processing techniques to enhance the accuracy. However, a systematic approach that compares the pre-processing techniques for both symmetric and asymmetric algorithms is missing, and hence it is not clear under which circumstances they are useful.

This paper provides a comparison analysis of several pre-procesing techniques, including three pre-processing techniques which have not been studied yet for DL-based SCAs.

Hence, five different methods are explored: i) Data augmentation [12, 14], ii) data transformation [16, 17], iii) data concatenation [18], iv) stacked auto-encoder [19]; and v) stacked auto-encoder with encoder only [20]. Note that data augmentation and stacked auto-encoder are already applied in the literature. Data transformation has been explored in some power attacks like CPA [16] and MLP-based [17], but not yet in DL-based attacks. The other two techniques come from the image processing field due to their outstanding results. To our best knowledge, this three methods are for the first time applied in DL-based SCAs. The main contributions of this work are:

- Proposal of data transformation using wavelet transform to improve DL-based SCAs.
- Proposal of data concatenation by augmenting the original trace with its fast Fourier transformation (FFT) to improve DL-based SCAs.
- Proposal of stacked auto-encoder using the encoder only to improve DL-based SCAs.
- Comparison of the proposed techniques with the state-of-the-art for both symmetric and asymmetric ciphers.

This paper is organized as follows. Section II provides a background on DL-based SCAs. Section III describes the pre-processing techniques. Section IV provides the experiments and their results. Section V discusses and concludes this paper.

II. Deep Learning Based Side Channel Attacks

DL-based SCAs follow the same steps used in the conventional profiled-based attacks [9], i.e., they consist of a **profiling** and **extraction** phase [10]. Both phases are explained next.

A. Profiling Phase

In this phase, the attacker creates a behavioral model of the selected target device using a similar or identical device that he/she fully controls. This phase works as follows:

Step 1: In this step, the attacker searches for a sample device that is similar to the target device.

Step 2: The attacker chooses and locates an intermediate point of attack (e.g., *SubByte* operation in AES or exponent operations in RSA).

Step 3: The attacker records multiple power traces of the selected target operation.

Step 4: The attacker designs a deep learning neural network and trains it to characterize the traces. To be able to train the deep learning neural network, the attacker needs to associate each collected trace during Step 3 with a label. The label can be calculated differently based on the used cryptographic algorithm. For example, the common label for AES is the hamming weight of the *SubByte* operation's output [10], while RSA typically uses the main operations as labels, namely *square* and *multiply* [21].

Step 5: Finally, the attacker constructs a neural network and trains it. The attacker first needs to define the structural parameters (e.g., depth, width, activation function) of the neural network. Thereafter, training is performed on the traces collected during Step 3 with the associated labels calculated in Step 4. To train the neural network, the attacker divides the dataset (i.e., traces and their labels) into a training set (normally 80% to 90% of the complete dataset) and a validation set. The training and validation phases are completed when the attacker achieves an acceptable accuracy level.

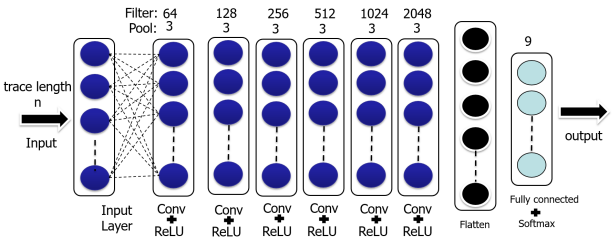

Fig. 1. Baseline CNN with its hyper-parameters

B. Extraction Phase

The attacker aims to recover the secret information from the target device during this phase using the following steps:

Step 1: The attacker identifies the target device. This device has to be similar to the profiled one.

Step 2: The attacker locates the intermediate operation, i.e., the operation used to train the neural network during the profiling phase. For instance, the *SubByte* of AES algorithm.

Step 3: The attacker generates a new set of traces on the target device for the intermediate operation. Note that in asymmetric algorithms, traces are divided based on their main operations (i.e., *square* and *multiply* for RSA), while such a partitioning is not needed for symmetric algorithms.

Step 4: The attacker predicts the key of the newly generated traces using the previously trained neural network. The result of this step is the label of the intermediate operation. This label is a binary value for the RSA algorithm or the leakage model value (i.e., hamming weight) for AES algorithm.

Step 5: Finally, the attacker reveals the secret key information. In asymmetric algorithms, the key is recovered bit by bit based on the predicted operations. In order to recover the full key, the retrieved bits have to be concatenated either from left to right or right to left based on the algorithm used. Symmetric algorithms require more steps, as the predicted leakage model value has to be converted from the hamming weight to a sub-key value. This extra task is shown in Algorithm 1. After calculating the probability of the target traces and knowing the plaintext/ciphertext used in the algorithm encryption/decryption process, the subkey is retrieved next. To guess a $subkey$ value, for every plaintext/ciphertext in pt array we loop over all possible subkey scenario $subkey = 0$ to 255 and calculate the leakage model results of that subkey. Based on the output of the leakage model, we select the corresponding probability from the prediction array and add it to the probability array. The key probability results are accumulated for each element in the pt and $Prediction$ arrays. The subkey with highest probability is selected as subkey. The previous process is repeated for each subkey.

III. Pre-processing Techniques

This section describes first our reference deep neural network. We refer to this reference DL model as baseline CNN. Thereafter, it describes the five pre-processing techniques

Algorithm 1 Symmetric Extract Key bytes

procedure KEY_EXTRACT($Prediction_{set}, pt_{array}$)
  $P_k[0, 255]$ = key probability
  $Prediction$ = the results of the trained model on the attack traces
  pt = is the plaintext used in the encryption process.
  for each sub-key **do**
    $P_k[0, 255] = 0$
    for j in trace-set **do**
      $X_{0,255} = predict(trace)$
      for k=0 to 255 **do**
        $HW_k = HW(SBOX[pt[j] \bigoplus k])$
        $P_k[k] = P_k[k] + log(Prediction_j[HW_k])$
      end for
    end for
    $guess_{subkey} = max(P_k)$
  end for
end procedure

which we divide into two types: traditional pre-processing techniques and neural network based approaches.

A. Baseline CNN

The baseline CNN is the reference neural network used for comparison when the pre-processing techniques are applied. The baseline itself does not use any pre-processing technique, which means that its inputs are the raw data values obtained from the collected power traces. As observed in Figure 1, the baseline CNN consists of 9 layers in total. The number of neurons in the first layer matches the trace length. Thereafter, it contains six convolutional layers, ReLU activation functions, and pooling layers. Note that for simplicity that several layers are presented together in the figure. Each of these layers contains a filter whose size is depicted on the top of the respective layer. For example, the first layer presents 64 filters. In the sequence, the pooling layer reduces the data width by a factor of 3. Note that the minimum layer width is equal to or greater than one. As the data goes through the network, more filters are added. Although the 7^{th} layer in the figure comprises much more filters than the first layer, its width is much less due to the pooling layers involved in the process. The last part of the CNN consists of a flatten and a Softmax layer. The flatten layer converts the tensor representation (i.e., multi-dimensional matrix) of the data used in the previous layers to a single dimension vector representation. The Softmax layer is the final layer of the neural network and consists of 9 output neurons used to distinguish between the 9 classes for AES. These 9 classes represent the prediction probabilities of different hamming weights or hamming distances. A similar neural network is constructed for RSA that contains 2 output neurons.

The training parameters are shown in Table I. Glorot [22] is the first parameter. Glorot is a sophisticated technique (as compared to e.g., random initialization) that initializes weights based on the width of its preceding and successive layers. Thereafter, the loss function is defined by the categorical entropy technique to compute the error function. The third parameter applies Adam optimization [23], which is a special technique that uses adaptive learning rates for each parameter and typically gives good results. Lastly, dropout is used to regularize the neural network; a dropout ratio of 50% is selected. Note that other dropout ratio values may be used. However, 50% typically provides good results [24].

TABLE I
TRAINING RELATED HYPER-PARAMETERS USED FOR CLASSIFICATION

Training hyper-parameters	Baseline CNN	SdAE + CNN	SdAE v2 + CNN
Initialization	Glorot	Glorot	Glorot
Loss Function	categorical	binary and categorical	binary and categorical
Optimization	Adam	Adam	Adam
Regularization	Dropout (0.5)	Dropout (0.5)	Dropout (0.5)

B. Traditional Pre-Processing Techniques

Three traditional pre-processing techniques are described in this subsection. They are data augmentation, data transformation and data concatenation.

1) Data Augmentation: This technique was already proposed by the authors in [12, 14]. It uses the average of traces to improve the extraction of the most meaningful features. All traces belonging to the same group, i.e., with the same hamming weight, are averaged to remove noise and misalignment between them and hence a cleaner version can be obtained. The traces that contain the average values for each hamming weight are replicated in such a way that they form half of the total number of traces for each hamming weight. Subsequently, random noise is added to those clean traces. Finally, the baseline CNN is trained using both the original traces (50% of the total traces) and the newly generated traces (the other 50% of the total traces). Note that the newly generated traces have the same label classifier (i.e. the same hamming weight) as the traces they were based on. The training phase is stopped when the accuracy of the model does not increase anymore.

2) Data Transformation: This technique aims to provide the CNN with a different data representation of the training set [25]. Our method applies a wavelet transformation for each power trace. The neural network is subsequently trained with only the wavelet samples with its associated labels calculated from their original counterparts. The model is trained until the accuracy saturates.

3) Data Concatenation: This technique aims to expand the data provided to the CNN [18]. By mixing different data representations of the same training set, improvements in prediction accuracy are expected. Our technique combines the FFT representation and the original trace to be represented as a single input. The model is trained until no further improvements are observed with respect to the accuracy.

C. Hybrid Neural Networks

In this subsection, two hybrid neural networks are presented. Each contains the baseline CNN preceded with another neural network; they are referred to as the SdAE + CNN and SdAE v2 + CNN. SdAE refers to the Stacked Denoising Autoencoder [26], while the SdAE v2 to the same SdAE but with

the encoder part only. The Stacked Denoising Autoencoder consists of an encoder and decoder and one of its main purpose is to filter input samples to enhance the quality. For example, SdAE has been used successfully to solve problems related to image colorization and noise reduction [26]. Another benefit of SdAE is that the encoding results in a lower dimension data, i.e., a compressed format of the input sample with a smaller number of points. On the other hand, the decoding part reconstruct the information to its original dimensions. This method has two main benefits. First, the encoding part compresses the data and extracts the most meaningful points in the trace. Note that the decoder decompress the compressed features to the full trace. Second, the SdAE can constructed in such a way that it restores missing parts or remove noise of traces. Next, we describe both hybrid networks in more detail.

1) SdAE + CNN: : In this strategy, the goal is to remove noise and misalignment of the input samples by training the SdAE. Two steps are required to achieve such results. First, the data-set is divided into groups (i.e., classes) based on their hamming weight. Second, the average of the traces is taken for the all input samples within the same group. The assumption here is that these average traces are much cleaner. Subsequently, the SdAE is trained with the normal input samples, but the error at the output is calculated using the average trace values. Once the training is completed, the result of the SdAE is classified using the CNN model. The overall training procedure of this approach consists of the following two phases:

Phase 1 - Training the SdAE model: In this phase, the SdAE is trained to reconstruct the power trace without noise and misalignment issues. This is achieved by using the data (i.e., recorded power traces) as input data-set to the train model, while using the average trace to calculate the loss function. After this training step reaches an acceptable accuracy, the next phase can start.

Phase 2 - Training the CNN model: In this phase, the output of the trained SdAE is connected as an input to the CNN model. Subsequently, the training mode of the weights and biases is switched off for the SdAE part of the hybrid network. Finally, the hybrid model (SdAE + CNN) is trained (i.e., only the CNN part) using the recorded power traces as input, while their hamming weights are used as the labels to calculate the error. The training stops when the accuracy saturates.

2) SdAE v2 + CNN:: This approach aims at exploring the usefulness of the encoded version of SdAE; note that this encoded version holds the most important characteristics of the input trace. In this phase, the pre-processing based on trace averaging is not needed. Instead, the SdAE is trained based on the original data. After the training is completed, only the encoder part of the SdAE is connected to the CNN as shown in Figure 3. The training process is also performed using two phases:

Phase 1 - Training the SdAE model: In this phase, the SdAE is trained similarly to the previous approach. However, the input data (i.e., the power traces) is used to compute the error at the output during training. After the training reaches an acceptable accuracy, the encoder part is removed from the SdAE.

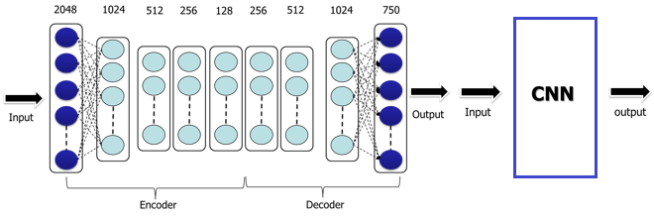

Fig. 2. SdAE + CNN and their hyper-parameters

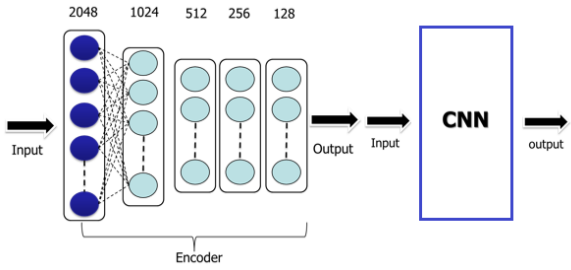

Fig. 3. SdAE v2 + CNN and their hyper-parameters

Phase 2 - Training the CNN model: In this phase, the trained encoder part is connected to the input layer of the CNN. Subsequently, the training for the SdAE encoder is switched off, i.e., their weights and biases are fixed to make sure that they are not adjusted during the training process of the hybrid model. After that, the hybrid model is trained (i.e., only the CNN part is being updated).

The structural related hyper-parameters of both approaches SdAE + CNN and SdAE v2 + CNN are illustrated in Figure 2 and Figure 3, respectively. Note that the CNN in Figure 2 is exactly the same as the one used in Figure 1, as the output layer in the SdAE has the same width as its input layer. The CNN in the hybrid neural network of Figure 3 is based on the same concept as the one in Figure 1, but its input width is smaller as the encoder of the SdAE compressed the data. More properties of both hybrid neural networks can be found in Table I.

IV. Experimental Results

This section presents the experimental setup, performed experiments and the results.

A. Setup

The experiments are conducted in two parts. In the first part, we evaluate the pre-processing techniques based on the mathematical approaches described in Subsection III-B. In the second part, we test the accuracy of the hybrid neural networks described in Subsection III-C. Both parts are evaluated using four different power traces; two are from AES cryptographic algorithms and two from RSA cryptographic algorithm. Their key characteristics are summarized in Table II and explained next:

1) **DPA Contest V2 [27]**: The traces in this training set are based on AES using a 128-bit key size. These traces are provided by DPA Contest V2; an open source framework

TABLE II
UTILIZED POWER TRACES AND THEIR CHARACTERISTICS

Traces	Crypto (key size)	Platform	Trace length
DPA Contest V2	Unprotected AES (128)	Hardware	3250 points
ChipWhisperer 1	Unprotected AES (128)	Software	3000 points
ChipWhisperer 2	Protected RSA (512)	Software	3000 points
Pinata	Protected RSA (512)	Software	8000 points

that allows developers to compare their implementation attacks using a common benchmark. The traces represent a hardware implementation of the decryption process. In these traces, no countermeasures against side channel attacks have been used. Each trace consists of 3250 data points.

2) **ChipWhisperer 1 [28]:** Similar to the previous data set, the traces here are also based on a non-protected AES encryption implementation with a 128-bit key size. However, here a software implementation is used instead. The data has been recorded using ChiphWhisperer tool which is an open-source tool for side-channel power analysis and glitching attacks. ChipWhisperer-Lite kitboard is used to measure the traces. The length of each recorded trace equals 3000 points.
3) **ChipWhisperer 2:** Unlike the first two types of traces, the traces in this training set are based on an asymmetric algorithm. Here traces are used of a 512-bit key software implementation of the asymmetric cryptographic algorithm RSA. The traces are collected on the same platform used for the ChipWhisperer 1 data set. The trace length is 3000 points.
4) **Pinata [29]:** A similar RSA software implementation has been used for this data set as ChipWhisperer 2. The difference is that the Pinata board from Riscure [30] is used for collecting the traces. Each trace contains 8000 sample points.

The ChipWhisperer data sets are measured in a controlled environment and hence provides cleaner traces. The other two data sets have been gathered from an uncontrolled environment which might come with certain restrictions.

B. Results of Traditional Pre-Processing Techniques

Table III provides the accuracy analysis results of the traditional based pre-processing techniques. The techniques are applied on both symmetric and asymmetric data sets.

For the symmetric data sets, the table shows the training accuracy, validation accuracy, and the maximum rank. The rank specifies how close the guessed key is to the correct key. A key rank of 0 means that the correct key has been guessed, while a key rank of 255 means that the correct key has the lowest guess probability. The lower the key rank, the more successful the attack is. From the table it can be observed that for DPA contest V2 the pre-processing techniques significantly improve the results. The data transformation technique (i.e., wavelet transformation) does not only improve the training and validation accuracy but more importantly, reduces the key rank from 17 to 2. The data augmentation also improved the neural network accuracy but ended up in a slightly higher rank (i.e., 5). The third pre-processing technique, i.e., data concatenation, actually worsened the results.The rank analysis results of the four techniques on DPA contest V2 data set are shown in Figure 4. The figure shows clearly that the data transformation has a strong positive impact on the accuracy of the attack. Note that it was difficult to see the effect of the pre-processing techniques on the ChipWhisperer 1 data-set, as the results of the baseline was already good.

TABLE III
PRE-PROCESSING TECHNIQUES COMPARISON AND RESULTS.

Evaluated Technique	**Step**	**AES Evaluation (15 byte)**		**RSA Evaluation**	
		DPA Contest v2	**CW 1**	**Pinata**	**CW 2**
Baseline CNN	Training	33.7%	98.62%	99.9%	100%
	Validation	23.5%	80%	75%	100%
	Final	Rank 17	Rank 0	80%	100%
Data Augmentation	Training	41.14%	98.78%	98.9%	100%
	Validation	27.5%	60%	75%	100%
	Final	Rank 5	Rank 0	70%	100%
Data Transformation	Training	40.17%	99.3%	99%	100%
	Validation	26.02%	85%	95%	100%
	Final	Rank 2	Rank 0	94%	100%
Data Concatenation	Training	30.1%	98.12%	33%	100%
	Validation	25.7%	70%	34%	100%
	Final	Rank 32	Rank 0	33%	100%
SdAE	Training	37%	90%	99.8%	100%
	Validation	29.5%	80%	95%	100%
	Final	Rank 80	Rank 0	90.1%	100%
SdAE v2	Training	28.1%	90%	98.6%	100%
	Validation	24.6%	70%	94.3%	100%
	Final	Rank 58	Rank 0	92.3%	100%

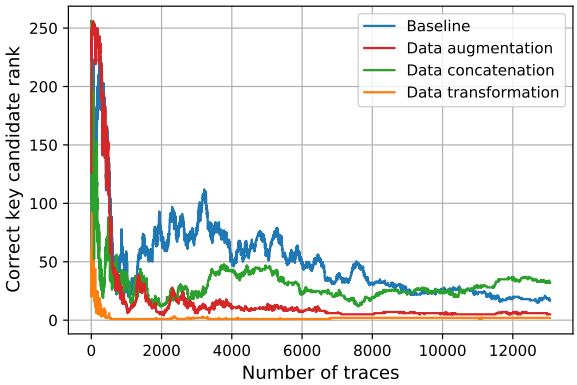

Fig. 4. AES key rank analysis for pre-processing techniques

For the asymmetric data set, instead of the key rank, the percentage of the key recovery is used. We refer to it as success rates. The results are similar to the symmetric data-set; the data transformation technique showed a significant improvement as compared to the baseline results, i.e., a success rate improvement from 75% to 95% for the Pinata traces. Here, the data augmentation has a marginal impact on the results, and the data concatenation technique again impacted the results negatively. Note that again no differences have been observed for the ChipWhisperer 2 traces.

C. Results of Hybrid Neural Networks Pre-Processing Techniques

Table III also provides the results analysis of the SdAE and SdAE v2 pre-processing techniques. Similarly to the traditional techniques, the results of both AES and RSA for the

chipWhisperer platform did not provide distinguishable results as the traces were already clean to start with. However, for the DPA contest V2 and Pinata traces, both hybrid networks were able to improve the validation accuracy. Despite this improvement, the rank analysis shows that these techniques did not improve the attack. In case the hyper-parameters of these hybrid neural networks are changed they will most likely perform better.

V. Conclusion and Discussion

This paper presented an investigation of different pre-processing techniques on the accuracy of deep learning based side channel attacks. Results showed that applying such techniques could improve the attack accuracy considerably. From the investigated techniques, data transformation and the hybrid neural networks strategies showed the biggest improvement. From this work, we conclude the following:

Impact of Pre-processing: As shown in the results, pre-processing techniques are able to improve the results in many cases. However, depending on the target device and attack environment, different pre-processing may fit better. Overall, pre-processing techniques could have a positive impact and hence should be considered during attacks. Designers should use this information to make stronger countermeasures.

Suitability in Uncontrolled Environments: It is very difficult to guarantee that all conditions and equipment involved in a side channel attack are optimal, with the exception of laboratorial research experiments. Hence, it is natural to assume that any practical attack will have limited controllability. Our results showed that pre-processing techniques could have an added benefit when the attacker does not control the attack environment.

Non-profiled DL-based SCas: Recently, Benjamin Timon [31] presented a successful non-profiled side channel attack using deep learning. Non-profiled attacks are interesting techniques as they do not require a similar/identical device to build the profiles. Hence, it is expected that more novel non-profile based attacks based on deep learning will be proposed in the future. The pre-processing techniques applied in this paper could also be considered in a similar manner for the non-profiled neural networks.

Acknowledgments. This work was labelled by the EUREKA cluster PENTA and funded by Dutch authorities under grant agreement PENTA-2018e-17004-SunRISE.

References

[1] Markets and Markets, "Cybersecurity Market worth 231.94 Billion USD by 2022," Available at: https://www.marketsandmarkets.com/PressReleases/cyber-security.asp, Last access: 2019-09-23.

[2] Juniper Research, "Internet of Things Connected Devices to Triple by 2021, Reaching Over 46 Billion Units," Available at: https://www.juniperresearch.com/press/press-releases/internet-of-things-connected-devices-to-triple-b, Last access: 2019-09-23.

[5] E. Brier, C. Clavier, and F. Olivier, "Correlation Power Analysis with a Leakage Model," in *CHES*, 2004.

[3] A. A. Pammu *et al.*, "Interceptive side channel attack on AES-128 wireless communications for IoT applications," *APCCAS*, 2016.

[4] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," in *CRYPTO' 99*, 1999.

[6] J.-S. Coron *et al.*, "Higher-Order Side Channel Security and Mask Refreshing," in *Fast Software Encryption*, 2014.

[7] K. Tiri *et al.*, "A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards," in *ESSCIRC*, 2002.

[8] F. Durvaux *et al.*, "Efficient Removal of Random Delays from Embedded Software Implementations Using Hidden Markov Models," in *CARDIS*, 2013.

[9] S. Chari, J. R. Rao, and P. Rohatgi, "Template Attacks," *CHES*, 2003.

[10] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking Cryptographic Implementations Using Deep Learning Techniques," 2016.

[11] V. Zeman and Z. Martinasek, "Innovative Method of the Power Analysis," in *Radioengineering 22*, 2013.

[12] Z. Martinasek, J. Hajny, and L. Malina, "Optimization of Power Analysis Using Neural Network," in *CARDIS*, 2014.

[13] R. Gilmore, N. Hanley, and M. Oneill, "Neural network based attack on a masked implementation of aes," *HOST*, 2015.

[14] E. Cagli, C. Dumas, and E. Prouff, "Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures," in *Cryptographic Hardware and Embedded Systems – CHES 2017*, 2017.

[15] H. Maghrebi, "Deep Learning based Side Channel Attacks in Practice," *IACR Cryptol. ePrint Arch.*

[16] N. Debande *et al.*, "Wavelet transform based pre-processing for side channel analysis," 2012.

[17] P. Saravanan *et al.*, "Power analysis attack using neural networks with wavelet transform as pre-processor," in *18th International Symposium on VLSI Design and Test*, 2014, pp. 1–6.

[18] N. Noreen *et al.*, "A deep learning model based on concatenation approach for the diagnosis of brain tumor," *IEEE Access*, pp. 55 135–55 144, 2020.

[19] L. Wu and S. Picek, "Remove some noise: On pre-processing of side-channel measurements with autoencoders," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 389–415, Aug. 2020.

[20] L. Macas-Garca *et al.*, "A study of the suitability of autoencoders for preprocessing data in breast cancer experimentation," *J. of Biomedical Informatics*, p. 33–44, 2017.

[21] M. Carbone *et al.*, "Deep learning to evaluate secure rsa implementations," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 132–161, 2019.

[22] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *AISTATS*, 2010.

[23] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[24] P. Baldi and P. J. Sadowski, "Understanding dropout," in *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2013.

[25] F. Xiao *et al.*, "Maximal overlap discrete wavelet transform and deep learning for robust denoising and detection of power quality disturbance," *IET Generation, Transmission Distribution*, vol. 14, pp. 140–147, 2020.

[26] S. Bigdeli and M. Zwicker, "Image restoration using autoencoding priors," 2017.

[27] VLSI Research Group – COMELEC Department of the Telecom ParisTech, "DPA Contest v2," Available at: http://www.dpacontest.org/v2/index.php, Last access: 2019-09-23.

[28] NewAE Technology Inc, "Chipwhisperer-Lite two part board," Available at: http://store.newae.com/chipwhisperer-lite-cw1173-two-part-version/, Last access: 2020-01-31.

[29] Riscure, "The CHES 2018 Challenge," Available at: https://chesctf.riscure.com/2018/news, Last access: 2020-02-15.

[30] Riscure, "Pinata board," Available at: https://www.riscure.com/product/pinata-training-target/, Last access: 2020-01-31.

[31] B. Timon, "Non-Profiled Deep Learning-based Side-Channel Attacks with Sensitivity Analysis," *TCHES*, 2019.