Received January 18, 2021, accepted January 23, 2021, date of publication February 1, 2021, date of current version February 9, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3055833

Non-Profiled Side-Channel Attack Based on Deep Learning Using Picture Trace

YOO-SEUNG WON[1], DONG-GUK HAN[2], DIRMANTO JAP[1], SHIVAM BHASIN[1], AND JONG-YEON PARK[3]

[1]Physical Analysis and Cryptographic Engineering (PACE) Lab, Temasek Laboratories @ NTU, Singapore 637371
[2]Department of Financial Information Security, Seoul 132-798, South Korea
[3]Samsung Electronics Company Ltd., Hwaseong 18448, South Korea

Corresponding author: Jong-Yeon Park (jonyeon.park@samsung.com)

This work was supported in part by the "National Integrated Centre of Evaluation" (NICE), in part by a facility of Cyber Security Agency (CSA), Singapore, and in part by the Institute for Information Communications Technology Promotion (IITP) Grant funded by the Korea Government (MSIT) for Development of SCR-Friendly Symmetric Key Cryptosystem and Its Application Modes under Grant 2017-0-00520.

ABSTRACT Over the years, deep learning algorithms have advanced a lot and any innovation in the algorithms are demonstrated and benchmarked for image classification. Several other field including side-channel analysis (SCA) have recently adopted deep learning with great success. In SCA, the deep learning algorithms are typically working with 1-dimensional (1-D) data. In this work, we propose a unique method to improve deep learning based side-channel analysis by converting the measurements from raw-trace of 1-dimension data based on float or byte data into picture-formatted trace that has information based on the data position. We demonstrate why "Picturization" is more suitable for deep learning and compare how input and hidden layers interact for each raw (1-D) and picture form. As one potential application, we use a Binarized Neural Network (BNN) learning method that relies on a BNN's natural properties to improve variables. In addition, we propose a novel criterion for attack success or failure based on statistical confidence level rather than determination of a correct key using a ranking system.

INDEX TERMS Binarized neural network, deep learning, multi-layer perceptron, non-profiled side-channel attack.

I. INTRODUCTION

Machine learning has seen great application in different use cases. Popular techniques like MLP (Multi-Layer Perceptron) [5] and CNN (Convolutional Neural Network) [18], [24] are widely utilized, but not limited to, in solving classification problems of variable complexity. Security evaluation with techniques like side-channel analysis [7], [13], [17], [19], [21]–[23], [32], [33] has also seen rapid adoption of MLP and CNN. The key idea here is to map the measured samples in a side-channel trace to input of a neural network while mapping the output labels to sensitive intermediate values. The network tries to learn the trace to sensitive value mapping in the training phase to classify attack traces with unknown labels.

The research on application of deep learning has been growing rapidly since its introduction by Maghrebi *et al.* [23], where they showed how MLP was utilized to break popular masking countermeasures. Cagli *et al.* [7] later demonstrated the use of shift invariance property of CNN to counter jitter based countermeasure, further improved by data augmentation techniques. Another work by Picek *et al.* [26] showed how the commonly used Hamming weight model leads to imbalanced training datasets and proposed techniques to overcome it. In [17], the authors propose the use of added Gaussian noise as a regularization technique to improve attack efficiency for different datasets while keeping the same VGG-like network. In recent work, Zaid *et al.* [33] further improved the attack efficiency by optimizing the network for each dataset individually. Recently, Masure *et al.* [22] proposed a comprehensive study of deep learning techniques with formal links to well established side-channel metrics. Machine learning schemes such as autoencoders are shown to be used to reduce the noise level in side-channel traces [32]. Perin *et al.* [25] also utilize the ensemble models to concentrate on the generalization for profiling attack. While all these techniques operate in a profiled or supervised setting, Timon [31] proposed an attack technique in the non-profiled setting, using training accuracy of the correct key as a distinguisher against wrong key.

The associate editor coordinating the review of this manuscript and approving it for publication was Fan Zhang.

Although all the noted previous work have shown interesting results, they processed side-channel trace as a one-dimensional stream of data. Kim *et al.* [17] adapted a VGG-like network which performed well in audio application to side-channel applications, owing to similarity in data type. However, it is well known that MLP and CNN are well suited for image classification. Research and optimizations of MLP and CNN have been demonstrated on image datasets like MNIST, CIFAR-10, SVHN *etc*. Thus, it is natural that these networks perform best on images.

The initial idea for treating side-channel as images was put forward by Park *et al.* [28]. Later, Hettwer *et al.* [14] considered picture trace form for side-channel analysis since many machine learning techniques are naturally developed for picture dataset such as MNIST and CIFAR, and report better performance as compared to treating SCA trace as a 1-dimension data. In this work, we conduct a comprehensive analysis of side-channel analysis that treat traces as images.

A. MOTIVATION

The previously conducted studies on side-channel analysis have only focused on how to apply a machine learning scheme without modifying any of its properties. As previously stated, we transform the raw trace to conform to MNIST style to use inherent properties of the machine learning scheme. This is based on the results shown in the previous study [20]. As illustrated in Figure 1, we can recognize "7" as a handwritten number.

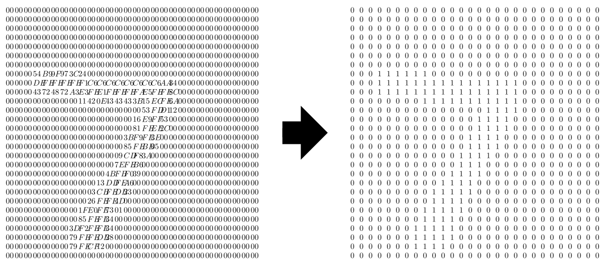


FIGURE 1. **7 value represented in the MNIST dataset.**

In fact, the '7' is represented using a hexadecimal value. While a single point in MNIST data is represented as a hexadecimal value, a float value is used to represent the concentrated form of the figure. Without the concentration, the value "7" is represented as shown on the right-hand side of Figure 1. Then, we describe why we remove the concentration. Above all, this MNIST style is significantly helpful in learning the data in terms of side-channel analysis. We expect that the best performance can be achieved if the trace is converted to MNIST style, because machine learning schemes, such as MLP, CNN, and BNN, have been advanced based on MNIST-style datasets.

B. CONTRIBUTIONS

We can summarize our contributions as follows

- We conduct a comprehensive analysis of side-channel analysis when treating traces in a picture form rather than original trace form. Our performance outperforms [14] by around 5×. Considering, in ASCAD datasets, our scheme only requires 7,000+3,000 traces to find correct key (non-profiled), while 50,000+250 traces are required in previous study (profiled).
- We provide detailed justification why the picture trace form outperforms original form in Section III-B. The weights of deep learning structure in picture trace form have a lot of solution range to learn the trace, as compared to original trace. Due to this fact, there is a disadvantage point for easy learning. Since our suggested form induces overfitting, we employ BNN as a solution for the first time in terms of side-channels.
- As indicated in [31], one cannot use the identity as leakage model in non-profiling scenario. Therefore, we should deal with the imbalanced leakage model (Hamming weight) making it difficult to use guessing entropy as a selection criteria [28]. To overcome this obstacle, based on Bernoulli distribution, we utilize the accuracy result of deep learning for attack success or failure rather than guessing entropy. Additionally, we prove that single bit of intermediate value is connected to the accuracy of machine learning result based on Bernoulli distribution and thus Guessing entropy is not required.

II. PRELIMINARIES

Deep Learning (DL) is a particular type of neural network-based machine learning that uses multiple layers in the network. It has also been applied to various fields, such as image and speech recognition. In this section, we briefly describe how to apply DL techniques, such as MLP and BNN, to side-channel analysis.

A. MULTILAYER PERCEPTRONS (MLP)

The general objective of MLP is to classify some input vector $x \in \mathbb{R}^D$ based on its labels $l(x) \in \mathcal{L} = \{l_1, l_2, \cdots, l_{|\mathcal{L}|}\}$, where D is the dimension of the input data to be categorized and $\mathcal{L}$ is the set of classification labels. The goal of an NN is to produce a function $NN\colon \mathbb{R}^D \rightarrow \mathbb{R}^{|\mathcal{L}|}$ that takes $x \in \mathbb{R}^D$ as the input vector and output y of the scores as the output. In other words, the final goal is to create the score vector $l(x)$ based on $NN(x)$, updating the internal properties. In general, an NN comprises input, output, and hidden layers. In terms of side-channel analysis, the input vector can be represented as points of a trace, and the output is compared to hypothetical intermediate variable, such as a Hamming weight model [4], to learn the expected result. Moreover, this hypothetical variable is usually encoded using a one-hot encoding scheme.

1) CONSTRUCTION OF MLP

In this section, we provide an example of a simple MLP. The number of inputs, also known as points, is four, and as shown in Figure 2, the example MLP comprises two hidden layers with three neurons and an output layer with two neurons.

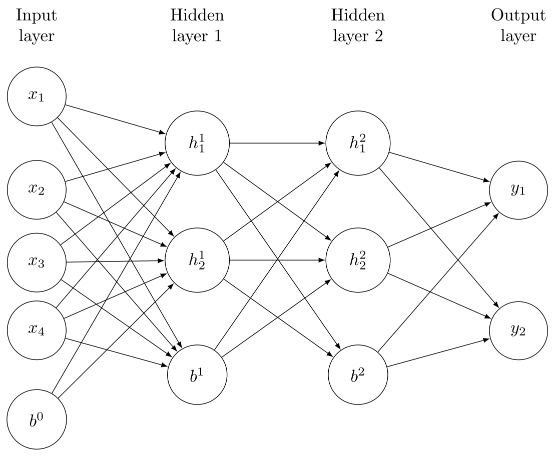


FIGURE 2. Example of a simple MLP.

In addition, a bias neuron is included in all layers, except for the output layer. All arrows in Figure 2 represent weights, as illustrated in Figure 3. Typically, prior to performing DL, the values of weights and the bias neurons are initialized from a normal distribution using the Xavier scheme [11]. To obtain the expected result from the output layer, some operations are performed to determine the output of each neuron. This procedure is called forward propagation. However, in this part, the weights have not been updated yet, so the achieved results might not be optimum. Optimal weights and bias values required to obtain the expected result are tuned via backward propagation during the learning process. This is done by comparing the expected result and the result of the output layer. The expected result is sometimes encoded using one-hot encoding. If one-hot encoding is applied to a single-bit-Hamming weight model, the output layer should comprise two neurons. If the Hamming weight value is 1, then the expected result is encoded to [1, 0] or [0, 1]. More precisely, by comparing $[y_1, y_2]$ and the one-hot encoded value, backward propagation is performed to update all weights.

2) FORWARD PROPAGATION

Forward propagation can be calculated as shown in Figure 3. A single neuron in all hidden layers and the output layer is computed by simple multiplication, addition, and an activation function. In the figure, $x_1 \cdot w_1 + x_2 \cdot w_2 + x_3 \cdot w_3 + b$ is operated prior to calculating the activation function. Then, the activation function f_{act} such as sigmoid, tanH, ReLU, softmax, and Swish, is computed to calculate forward propagation. Excluding the output layer, ReLU is sometimes adopted for all hidden layers.

3) BACKWARD PROPAGATION

The core of MLP is backward propagation because each weight can be updated to learn the expected result. Backward propagation is performed by comparing the expected result $l(x)$ and the output of MLP $NN(x)$. Here, an error function, such as Euclidean distance, can be used to learn the expected result. Normally, an error function $E : \mathbb{R}^D \to \mathbb{R}$ can be

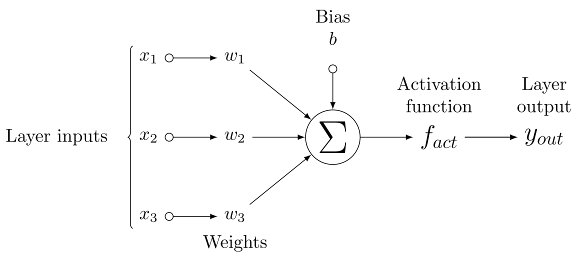


FIGURE 3. Calculation for a single neuron in hidden and output layers.

defined, *e.g.*, as the Euclidean distance between the MLP output and the one-hot encoded label, as follows:

$$E(x) = \sum_{i=1}^{|\mathcal{L}|} (l(x)[i] - NN(x)[i])^2 \tag{1}$$

where $l(x)[i]$ and $NN(x)[i]$ indicate the label value and the output neuron value, respectively. The error function value represents the gap between the expected result and the MLP output. In other words, backward propagation narrows the gap. To reflect the error for all training data $X = (x_i)_{1 \le i \le T}$, a loss function is defined as follows.

$$\mathcal{L} = \frac{1}{T} \sum_{i=1}^{T} E(x_i) \tag{2}$$

The weights can be updated using a gradient descent technique [12], which is applied to the loss function $\mathcal{L}$. Because loss function variables are based on the weights, the weights are trainable. Here, this concept is denoted as $\mathcal{L}_{\mathsf{w}}$. In other words, based on the gradient descent technique, the weights in the loss function can be updated with $\nabla \mathcal{L}_{\mathsf{w}}$. Utilizing the t-th result, $\mathcal{L}_{\mathsf{w}(t)}$, $(t+1)$-th weights can be learned as follows:

$$\mathsf{w}(t+1) = \mathsf{w}(t) - \alpha \nabla \mathcal{L}_{\mathsf{w}(t)} \tag{3}$$

where α denotes the learning rate. The total quantity for training T is consumed to learn w. However, even though the T quantity is used, the T quantity can be reused to learn w, which is referred to as an epoch.

For side-channel analysis, DL is applied as follows.

- **Input layer** The points of a trace correspond to the number of input values.
- **Output layer** The adversary sets the expected result. If the target intermediate variable for attacks is the Hamming weight of a single bit of the S-box output, then the output layer has two neurons where one-hot encoding is applied.
- **Hidden layer** Excluding the input and output layers, DL can be applied to the intermediate layers in order to learn a more accurate result than a single-layer NN. This hyperparameter significantly depends on a rule of thumb, *i.e.*, the adversary selects the parameter based on trial and error.
- **Learning rate** A ratio utilizing the previous training result is used to update the weight, between the range of 0.0 and 1.0.

- **Activation function** To activate each neuron, activation functions such as sigmoid, tanH, ReLU, and softmax can be applied. Empirically in many studies, ReLU is applied to hidden layers and softmax is used for the output layer.
- **Initialization** Initialization involves setting the initial values of weights and biases. The initial values can be set as random variables in a Gaussian distribution or additional techniques can be used.

4) ADDITIONAL FUNCTIONS

Various additional functions are used to avoid some obstacles, such as overfitting and inefficient use of memory, in machine learning schemes. We describe the two techniques that are used in this study.

- **Batch size** The total training data are divided into batches or sets, in order to avoid overfitting by updating the weights based on batch size.
- **Dropout** The key idea is to randomly drop units from the NN during training. That is, only some connections are updated. Here, the range is between 0.0 and 1.0. For example, 1.0 means no drop.

B. BINARIZED NEURAL NETWORK (BNN)

In this section, we introduce the concept of BNN. In recent works, the BNN variables are used for image classification works, so they are suitable for our picture-formatted scheme [15]. For implementation, BNN can reduce the memory and computational requirements of a deep NN. The core idea is binarization of all possible components, such as activations and weights. After converting the input to $+1$ and -1, we can utilize the binarization of weights and activations. The binary computation is shown to be 7 times faster than float computation and more energy efficient than MLP [8]. This would result in a faster evaluation even though the BNN might have higher complexity compared to MLP [8].

1) BINARIZATION OF WEIGHTS

When calculating initialization and forward propagation, applying binarization is not problematic. The primary issue with binarization is back-propagation. Courbariaux *et al.* [15] found very simple solutions for maintaining the real values of trained weights.

$$W_{\mathcal{B}} = \text{sign}(W_{\mathcal{R}}) \tag{4}$$

After updating the weights using gradient descent, they are normally real values ($W_{\mathcal{R}}$). For binarization, we can use the sign function resulting in a tensor with values of $+1$ and -1. Then, in order to use forward propagation, $W_{\mathcal{B}}$ can be obtained from the binarization of $W_{\mathcal{R}}$.

2) BINARIZATION OF ACTIVATIONS

To perform backward propagation, we need to make the activation function learnable. Normal backward propagation cannot be applied to a BNN because the output of the activation function is -1 or $+1$. [15] suggested using gradient descent considering binarization, as follows.

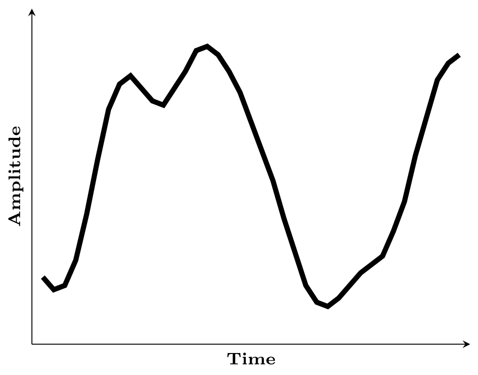


FIGURE 4. Trace for side-channel analysis.

$$\frac{\partial L}{\partial a_{\mathcal{R}}} = \frac{\partial L}{\partial a_{\mathcal{B}}} * 1_{|a_{\mathcal{R}}|\leq 1} \tag{5}$$

Here, $a_{\mathcal{B}}$ is the binarized output of the activation function and $a_{\mathcal{R}}$ is the real value input to the activation function. $1_{|a_{\mathcal{R}}|\leq 1}$ evaluates to 1 if $|a_{\mathcal{R}}| \leq 1$; otherwise 0. This zero drops out the gradient if the input of the activation function is greater than absolute value 1. We utilize an open-source BNN that is publicly available at https://github.com/uranusx86/BinaryNet-on-tensorflow.

III. PICTURE TRACE AND ANALYSIS

In this section, we introduce a preprocessing method for power traces and the basic attack principle. In general, if an event occurs at one time (t), there is a voltage value that has a single point value, which is a one-dimensional vector. For example,

$$v(trace) = 2.34$$

A collected power trace is a sequence of these voltages. If there is n-time point, one can express n-dimension (*i.e.*, n-time) as

$$v(trace) = (2.34, 1.36, 2.50, 2.97, \ldots)$$

The floating with the n-dimensional vector is described as follows (Figure 4), and is a normal power trace, where the x-axis indicates time and the y-axis indicates voltage.

In Figure 4, the voltage on the time axis expresses "degree", and the power trace can be visualized for human recognition. One can recognize power flow through upper markings and lower markings, which represent a high degree of voltage and represent a low degree of voltage, respectively. However, a real voltage flow is an n-dimensional vector sequence.

We perform side-channel analysis with picture-formatted power traces. There are many methods for converting an n-dimensional vector sequence into a picture. We use the approach from the MNIST database. As illustration, in Figure 5, we show a handwritten number "7", and we can perceive the value through some values in a 28 × 28 area.

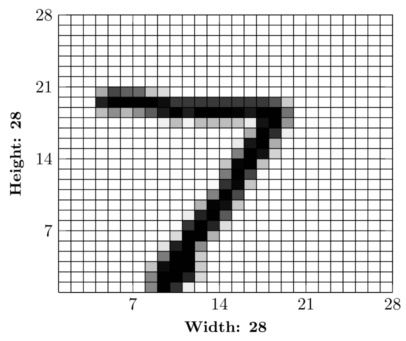


FIGURE 5. "7" represented in MNIST database with concentration.

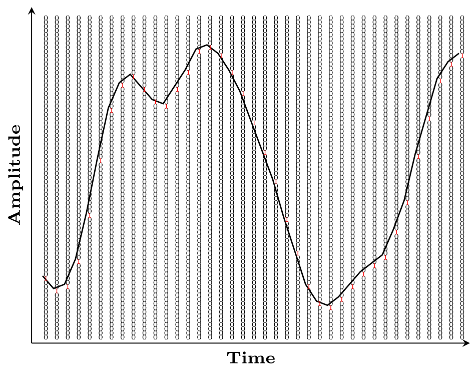


FIGURE 6. picture-formatted trace for side-channel analysis.

Here, there are visually unnecessary areas that are treated as the value "0", which means NULL. We seek to visualize a power trace using this method. Figure 6 shows a visualized power trace, which is a fundamental shape that we desire to change from a vector sequence to a picture. Here, the meaningful line is 1, and 0 represents nothing (like a foundation that helps us perceive the relative location of point "1"). We call it a picture trace. The steps for obtaining a picture trace are described as follows.

A. PICTURE ENCODING

1) DECIDING THE RESOLUTION

Resolution determines how a picture trace is represented in detail. To obtain all information from the original power trace, we know the original trace's resolution when the trace is collected from the collecting device, *e.g.*, an oscilloscope. Otherwise, we can easily conduct a brute-force search to find the resolution. Figure 7 shows how we determine the resolution when converting a power trace to a picture trace. Here, the original trace is an n-dimensional vector space whose size is equal to the number of points; however, the size of a picture trace is expressed as (number of points $\times$ resolution).

Note that we can select a smaller resolution, which results in a smaller input size. This facilitate relatively faster learning and reduces memory consumption. However, some information from the original trace might be missing, which can distort the real information in sensitive data. In what following, we discuss how to control such content.

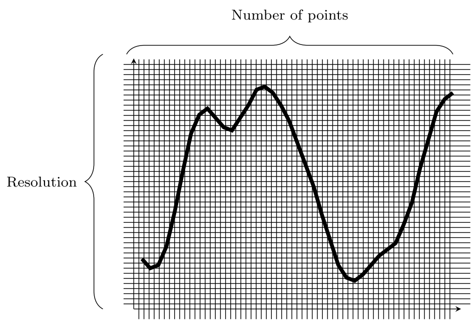


FIGURE 7. Determination of resolution for a trace.

2) DRAWING THE "Picture"

The drawing is the main step in generating a picture trace from an original trace. To draw a picture, we must determine the position of a point and place a dot in fixed space. In other words, we should determine the position of all points within the defined resolution. The related position of a point is determined using the upper bound, lower bound, and the chosen resolution. First, the gap, *i.e.*, the difference between the lower and upper bounds, needs to be computed. Note that each gap in each trace differs; however, the transformed traces must be identical to realize the same analysis standard and facilitate simple implementation.

$$\mathsf{Gap} = upperbound - lowerbound \tag{6}$$

Here, the upper bound is the maximum voltage value and the lower bound is the minimum value of all sample power traces. After computing the boundary, the related location is computed. The resolution is denoted as n, and the voltage value at a specific time is denoted as v; thus, the related location is computed as $t = \varphi(v)$, which is an intuitive concept.

$$\varphi : R \rightarrow Z_n \tag{7}$$

$$t = \varphi(v) = \left\lfloor \frac{v - lowerbound}{Gap} \right\rfloor \times n \leq n \tag{8}$$

Finally, we can generate an n(resolution)-th vector with related location t that is an integer in $[0, n-1]$. The method for expressing the picture follows MNIST; however, there is no depth in our power trace. Therefore, "1" represents the related location, and "0" means NULL. Assuming that the related location is t, the n-th vector is computed as follows.

$$\begin{aligned} \delta : Z_n &\rightarrow (Z_2)^n \\ \delta(t) &= (a_0, a_1, \ldots, a_{n-1}) \\ a_i &= \begin{cases} 0 & (i \neq t) \\ 1 & (i = t) \end{cases} \end{aligned}$$

The final picture of each trace is a set of vectors $\delta(t)$. The picture is $\{\delta_1(t_1), \delta_2(t_2), \ldots, \delta_m(t_m)\}$.

3) REDUCING THE REDUNDANCY WITH FILTER

The reducing step is the elimination of unnecessary parts of the picture. The newly drawn picture-shaped trace has

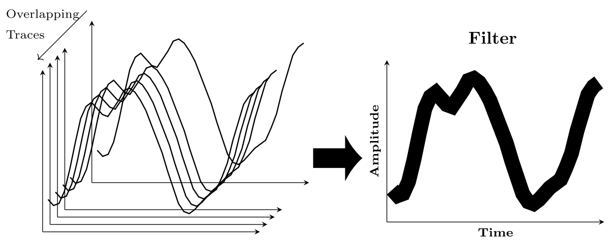


FIGURE 8. Overlapping traces for generating the filter.

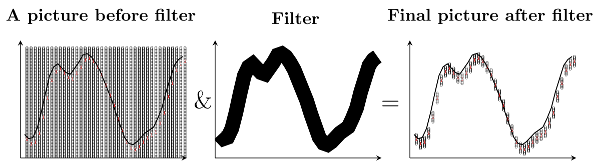


FIGURE 9. Generating final figure trace based on the filter.

unchanged "1" to "0" or "0" to "1". This area has no effect on power analysis. In our experimental results, most of the unchanged sections of the pictures are "0", *i.e.*, generally the foundation. Note that we cannot know an unchanged section from a single trace, and thus, filter out the unchanged portions by brute force search of each trace. The filter is created by stacking all picture traces and selecting vectors with a value of "1".

Figure 8 shows the method used to generate the filter. The filter eliminates unchanged "0" values; thus, we obtain only meaningful locations (Figure 9). The output picture on the right side of Figure 9 is inside the filter boundary.

In our results, filtering the picture traces reduces the input size of the ASCAD example by approximately 90%. Reducing the input size enables efficient learning time and reduces the number of training epochs.

Additionally, we can consider the efficient method to save the points because the points of picture-formatted traces are composed of 1 or 0. Unlike the original traces, the multiple points can be compressed to single data. For example, eight 0/1 points can be stored single byte data. Then, we can save the 1/8 data size.

B. CORRECT KEY DETERMINATION

For correct key determination, the previous non-profiling attack published in CHES 2019 [31] selected the best training speed, compared to other incorrect keys. However, this method assumes that DL with a correct key will learn in a more efficient way of providing labels. This means that weights and bias with higher training accuracy can compute correct labels for any power trace. However, a highly accurate training trace does not guarantee finding a correct key (Section IV-C), because DL can fall into the overfitting problem. In some cases, the learning with incorrect keys could have a faster learning speed.

Thus, we need to develop other standards to determine correct keys. We use a validation set for the final determination of correct or incorrect keys with mathematical confidentiality. Note that this validation set is not used for the learning phase. In fact, we require additional power traces to determine a correct key. For our method working, the validation set must be uniformly distributed for each label. For example, single-bit labeling has approximately 50% "1" and 50% "0". Accuracy is estimated by the number of equal result between the labeling with computed by guessing key by cryptographic operation and the labeling with machine. Here, the validation set has n power traces with exactly 50% "0" and 50% "1". We set Bernoulli trials(Bern) with success; the result of the same labeling between machine and cryptographic operation. The probability mass function is $Bern(x, p) = p$ when x = 1 (success), and $Bern(x, p) = 1 - p$ when x = 0 (failure) and the distribution following to Bernoulli trials that is matching between labeling with guessing key and the labeling with correct key is $X \sim Bern(x, p)$. In our 1 bit model, X follows Bernoulli trials $X \sim Bern(x, 1/2)$. For each validation traces, Bernoulli trials $X_1, X_2, \ldots, X_n$ are independent. Then, their sum is distributed according to a binomial distribution with $n, \frac{1}{2}$

$$\sum_{k=1}^{n} X_k \sim \mathcal{B}(n, \ \frac{1}{2})$$

If n is sufficiently large, $\mathcal{B}(n, p)$ is given a normal distribution [34]

$$\mathcal{N}(np, \ np(p-1))$$

For example, if n equals 3000, the binomial distribution can be approximated as a normal distribution. By using the number of correct prediction x, we calculate the cumulative distribution function (CDF) of x successes based on normal distribution approximation. Therefore, the distribution follows $\mathcal{N}(1500, 750)$, and we can determine whether the selected key for the error possibility is correct, using a probability density function, where the lower cumulative distribution $\mathcal{P}\left(x, 1500, \sqrt{750}\right)$ is computed as $\int_{-\infty}^{x} \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2}\left(\frac{x-1500}{\sigma}\right)^2}$. If 1600 passes in 3000 queries of key= 0x33, $\mathcal{P}\left(1600, \ 1500, \sqrt{750}\right) = 0.999869$. Here, the desired correct key is 0x33, and error rate is $1 - 0.999869 = 0.000131$. This is a very strong evidence 0×33 is correct key regardless of other key's result.

Multiple bits can be applied to this method but one bit model is ideal and perfectly balanced. On the other hand, one can perform each bit separately, and if one of the result of each bit shows meaningful confidential level, correct key can be determined by provable confidentiality. One does not need to test other bits as well as other key candidates.

C. CHARACTERISTICS OF PICTURE TRACE

1) EASY LEARNING WITH DIFFERENT FORMAT OF INPUTS

A notable difference in picture traces is that the alteration of voltage for each trace changes the vector's location. For example, as shown in Figure 10, there are only two power traces, and three voltage values at three times. The left side

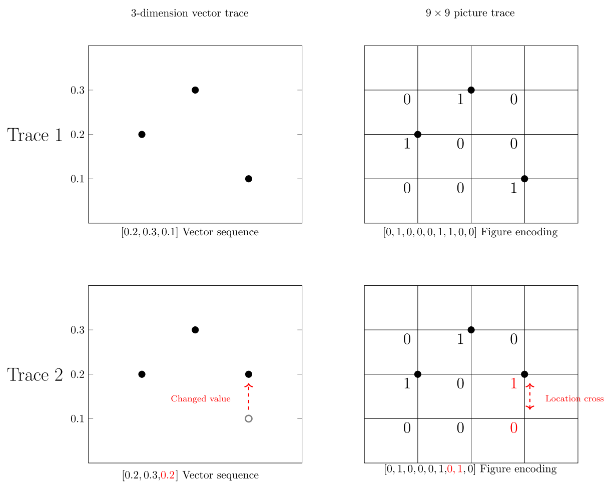

3-dimension vector trace
9 × 9 picture trace
Trace 1
Trace 2
Changed value
Location cross
[0.2, 0.3, 0.1] Vector sequence
[0, 1, 0, 0, 0, 1, 1, 0, 0] Figure encoding
[0.2, 0.3, 0.2] Vector sequence
[0, 1, 0, 0, 0, 1, 0, 1, 0] Figure encoding

FIGURE 10. Example of easy learning.

of Figure 10 shows normal traces, *i.e.*, a three-dimensional vector sequence. The first trace's third value is 0.1, and the second trace's third value is 0.2. The right side of Figure 10 shows the picture trace. Here, the input vectors of the picture trace are [0, 1, 0, 0, 0, 1, 1, 0, 0] in the first trace and [0, 1, 0, 0, 0, 1, 0, 1, 0] in the second trace. Alteration of these two traces is a location change expressed as [1, 0] to [0, 1]. The normal trace's alteration is 0.1 to 0.2 that is changed value. Actually, information of inputs can be reduced when resolution is lower than original trace's environment. However higher resolution does not guarantee good performance of analysis, we will show the result after this chapter. More important factor for success of attack is not always maximal information but the more proper format and size of inputs for efficiency of ML. We will show the experimental result with variation of resolution.

Now, we will show how to work changed inputs in each nodes. According to expanded dimension of input space, it can find proper solution easily to learn faster for NN fitting with weight and bias. Figure 11 shows an intuitive appearance of learning using normal traces. For understanding of the principle, we set comparison between learning of normal traces and picture traces with specific structure and actual numbers; in the last layer of the NN with softmax and one-hot encoding, the voltage values of the two traces are 1 or 2. The last step of the network selects a location with a greater value of P_1 or P_2 after softmax. Note that softmax is only used to normalize the final value from 0 to 1; thus, an upper neuron is selected when $x_i w_1 + b_1$ is greater than $x_i w_2 + b_2$; otherwise, a lower neuron is selected. If the learning process is performed effectively, the NN computes a different selection that depends on input 1 or 2 as x_i. Let us consider that input 1 makes the network select the upper neuron and input 2 makes the network select the lower neuron (Figure 11). For the desired result, $w_1 + b_1 > w_2 + b_2$ if the input is 1 and $2w_1 + b_1 < 2w_2 + b_2$ if the input is 2. Note that these inequalities share the same variables w_1, b_1, w_2, b_2, with opposite inequality signs.

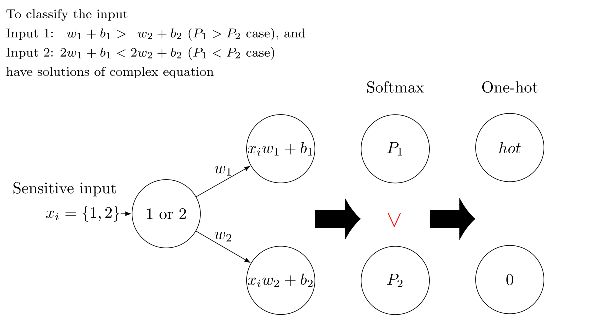

To classify the input
Input 1: $w_1 + b_1 > w_2 + b_2$ ($P_1 > P_2$ case), and
Input 2: $2w_1 + b_1 < 2w_2 + b_2$ ($P_1 < P_2$ case)
have solutions of complex equation
Softmax
One-hot
Sensitive input
$x_i = \{1, 2\}$
1 or 2
w_1
w_2
$x_i w_1 + b_1$
$x_i w_2 + b_2$
P_1
P_2
∨
hot
0

FIGURE 11. Intuitive appearance of final layer using normal traces.

Here, we find the solution. If we set $b_2 - b_1 = t$, we obtain the following:

$$w_1 > w_2 + t \tag{9}$$
$$w_1 < w_2 + t/2 \tag{10}$$

Figure 12 shows the area that is sufficient for those inequalities, where the x-axis indicates w_2 and the y-axis indicates w_1. The blue area indicates the values of weights that solve the simultaneous inequality. According to Figure 12, we can identify two conditions. The first condition is that t must have a negative sign, which means that b_1 must be greater than b_2. As long as, t has positive sign, the simultaneous inequality does not have solutions with any real number of w_1 and w_2; this means the learning failure. The second condition is that the area satisfying the simultaneous inequality can be considerably narrow relative to variable t. Even though there are solutions, the machine will likely struggle to learn to find the desired solution without requiring many epochs.

Here, we assume that the input is (1,0) and (0,1) (rather than 1 and 2), which is a picture trace example (Figure 13). Note that this case requires four weights, *i.e.*, w_{11}, w_{12}, w_{21}, and w_{22}. As in the previous example, $w_{11} + b_1 > w_{12} + b_2$ when the input is (1, 0), $w_{21} + b_1 < w_{22} + b_2$ when the input is (0, 1). Unlike normal traces, each neuron has its own weight value that can affect to connect next layers' inputs. Here, we set $t = b_2 - b_1$. Then, we obtain the following:

$$w_{11} > w_{21} + t \tag{11}$$
$$w_{21} < w_{22} + t \tag{12}$$

There are five variables on two simultaneous inequalities, *e.g.*, $w_{11}, w_{21}, w_{22}, w_{12}$ and t. Regardless of the values of the four variables, the value of the remaining variable allows simultaneous inequality to work by itself. There is no dependencies on each variables, and no conditions such as Figure 12.

In the intermediate layers, there is no choice function, *i.e.*, only the weight and bias influence the inputs to the next layer. Figure 15 shows a model where the input affects multiple neurons. Here is a concrete example with specific structure and numbers for understanding the final layer' example above, with one neuron in the first layer and two neurons in the second layer with ReLU. This example can intuitively demonstrate huge increases in the neuron size.

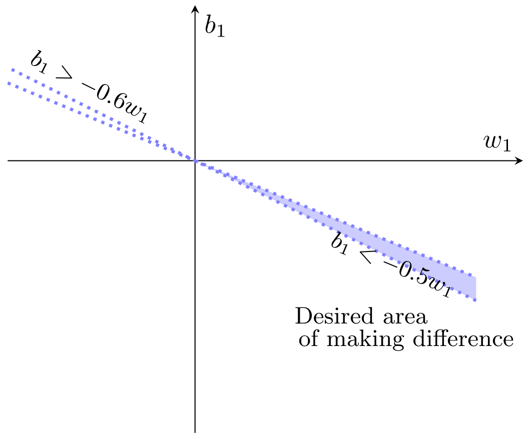


FIGURE 12. Area of solution of final layer using normal traces.

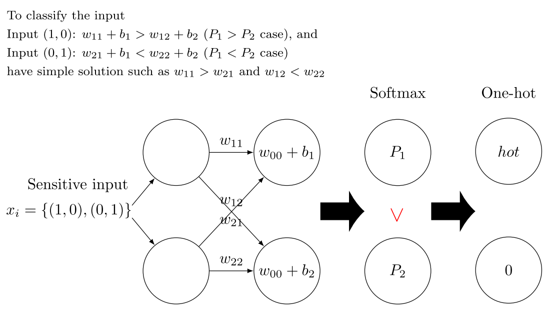

To classify the input
Input $(1,0)$: $w_{11} + b_1 > w_{12} + b_2$ ($P_1 > P_2$ case), and
Input $(0,1)$: $w_{21} + b_1 < w_{22} + b_2$ ($P_1 < P_2$ case)
have simple solution such as $w_{11} > w_{21}$ and $w_{12} < w_{22}$

FIGURE 13. Intuitive appearance of final layer using figure traces.

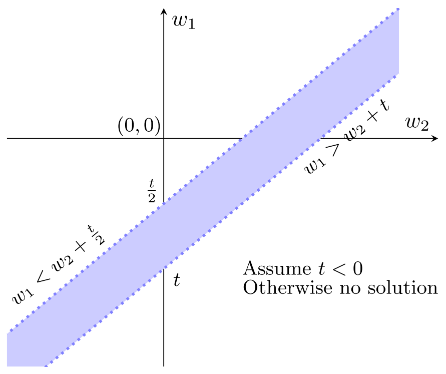


FIGURE 14. Intuitive appearance of an intermediate layer using normal traces.

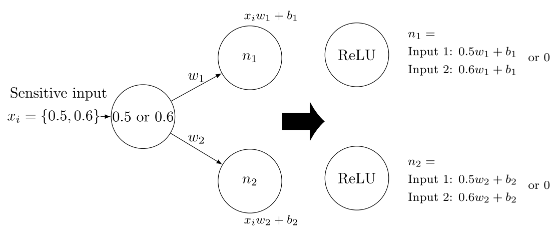


FIGURE 15. Intuitive appearance of hidden layer using normal traces.

We concentrate on only "one neuron to one neuron," whose output is n_1 (ignore n_2 in here). Here, we assume that the input is 0.5 or 0.6; normally power traces are normalized in [0,1]. Then, the output of n_1 is $0.5w_1 + b_1$, $0.6w_1 + b_1$, or 0, as computed by ReLU activation. To improve learning, we seek a huge difference between $0.6w_1 + b_1$ and $0.5w_1 + b_1$. The difference between the two values is normally $0.1w_1$ in our example; however, due to ReLU, there cannot be a difference between $0.6w_1 + b_1$ and $0.5w_1 + b_1$. If both are negative, then both are "0"; this case is learning failure. In contrast, if either of $0.6w_1 + b_1$ and $0.5w_1 + b_1$ is negative, a few of the values of w and b make the equation sufficient; see the bottom-right of Figure 14. When we consider the most positive case, the maximum probability is 0.25, `i.e.`, on of the four areas. Therefore, the probability has a value of at most 0.25 in the learning system when we only consider normal traces. In other words, the weight and bias are stochastically affected by each other's value to successfully learn the traces.

$$0.5w_1 + b_1 < 0 \tag{13}$$
$$0.6w_1 + b_1 > 0 \tag{14}$$

denotes,

$$b_1 < -0.5w_1 \tag{15}$$
$$b_1 > -0.6w_1 \tag{16}$$

Figure 16 shows the intermediate layer of a picture trace. Similar to the last layer example shown in Figure 13, here, (1,0) or (0,1) matches different weightsw_{11}, w_{12}, w_{21}, and w_{22} independently, which affects the next layer's neurons more than the normal inputs shown in Figure 15.

To realize better learning, we seek a huge difference between $w_{11} + b_1$ and $w_{21} + b_1$. Irrespective of whether or not $w_{11} + b_1 < 0$, $w_{21} + b_1$ can theoretically have any positive value. In fact, it depends on the scale of b_1's value; however, it affects the same addition on each part. Thus, we ignore b_1 in our theoretical analysis here. Therefore, the difference in the result of the next neurons is theoretically 0 to infinity, owing to only each weight value. Because a single-bit-difference Hamming weight has a significant impact according to the inputs, in a side-channel attack with Hamming weight, this property critically works on improvement of learning performance. The examples above shows small element of whole structure. Nevertheless, the characteristics of the elements are spread to whole network for sure.

2) OVERFITTING: A NEGATIVE ASPECT OF PICTURE TRACE

According to different attack models, most of the voltage information of a power trace for a side-channel attack is simple noise that is not used to find the correct key. Here, large inputs do not correspond to significant amounts of information, and larger inputs may induce additional noise. With a normal trace, additional noise slows the learning process because noise interferes with finding a solution. Thus, we consider the following simple mapping rule.

$$\text{More noise} = \text{Slower learning}$$

However, picture trace makes learning easy with many weights, biases, and input sizes. We find simultaneous equations with many variables using only a single simple equation. There are many possible solutions; however, only one

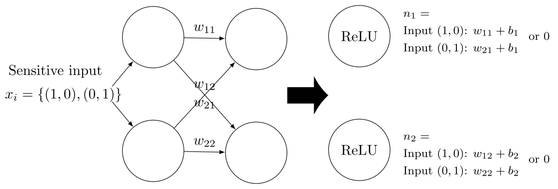

w_{11}
w_{12}
w_{21}
w_{22}
Sensitive input
$x_i = \{(1,0),(0,1)\}$
ReLU
$n_1 =$
Input $(1,0)$: $w_{11}+b_1$
Input $(0,1)$: $w_{21}+b_1$
or 0
ReLU
$n_2 =$
Input $(1,0)$: $w_{12}+b_2$
Input $(0,1)$: $w_{22}+b_2$
or 0

FIGURE 16. **Intuitive appearance of hidden layer using figure traces.**

solution is correct. In this case, deep learning produces a network with weights and biases, but this is a wrong answer. In addition, there are significant noise levels in power traces for a side-channel analysis in some attack models. Thus, we present another simple mapping rule.

$$\begin{aligned} &\text{More noise}\\ &\quad +\text{Higher learning ability}\\ &\quad +\text{More inputs and following weight and bias}\\ &= \text{overfitting}\end{aligned}$$

Therefore, DL with a picture trace is more susceptible to overfitting problems, because of the following reasons.

1) Considerable noise (some of the attack models in side-channel analysis)

2) Several inputs, followed by weights and biases

3) Higher learning ability

Therefore, overfitting is a characteristic of analysis with picture traces. Many studies have investigated reducing overfitting, *e.g.*, using sufficient traces, L1 and L2, regularization, and dropout or early stop. Those techniques do not get noticeable "reducing overfitting" effects in our environment; only drop-out slightly turns down overfitting. In a BNN, the outputs of the activation function are -1 or $+1$; therefore, the NN has a structural limitation relative to finding the correct answer with variables. This limitation allows us to expect to remove overfitting, *e.g.*, dropout, which randomly ignores some neurons. We can expect BNN to reduce overfitting.

IV. EXPERIMENTAL RESULTS

In this section, we describe our experiments and present the results. The experiments were conducted on the masked AES traces measured from ChipWhisperer lite [9] and the ASCAD database [27]. The resolution for the picture traces was set to the maximum, and a quarter of the maximum resolution. Reducing the resolution may result in loss of some information. However, it makes learning faster, and to some extent, reduces overfitting caused by reduction in the number of weights related to inputs to the first layers. The only difference between the first-order attack and higher-order attacks is the assignment of the label. If one uses power traces that have masked sensitive data, the label is generated by the pure data. If the masking value is known, the label can be set with unmasked values. Thus, our first-order attack uses unmasked labels with known masking values. In addition, the second-order attack uses exactly the same power traces and sets labels the S-box output without masking. Assuming a second-order attack, the machine can solve difficult problems.

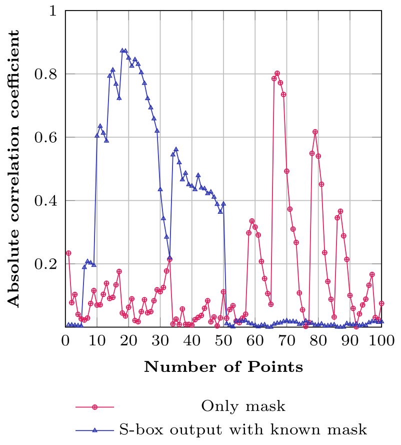


FIGURE 17. **Result of correlation coefficient of CW traces.**

In addition, we describe how MLP and BNN schemes are used to perform side-channel analysis in this work. MLP is one of the commonly used machine learning for side-channel analysis. We also employ a BNN in side-channel analysis because its scheme fits the proposed approach. To the best of our knowledge, this is the first time that a BNN has been employed in side-channel analysis.

A. MLP/BNN BASED DL ATTACK ON MASKED AES TRACES FROM ChipWhisperer LITE

The target operation is AES SubByte and the target data are the S-box output (fourth-order S-box). The data are randomized by exclusive-or, with random 8-bit masking. First, we check the correlation power analysis (CPA) [4] on the power traces, and as expected, the key cannot be recovered from the first-order attack. The CPA results are shown in Figure 17. There are two correlation coefficient lines on 100 points. The blue line indicates the analysis of the S-box output with known masking, which is the first-order CPA attack on the S-box output, and the pink line indicates the CPA with only masking value. We make simple sample power traces that only involve a fourth SubByte operation and masking value of over 100 points. Each correlation coefficient is formed up to 0.8 $\sim$ 0.9. One can easily expect that the first-order analysis is well conducted with a DL attack using the known mask values.

Figure 18 shows the attack results with MLP and BNN, where the left graph shows the first-order attack and the right graph shows the second-order attack. In both graphs, Figure-α means a deep learning attack with picture trace using resolution α. Normal means a DL attack with normal traces, where 256 is the maximum resolution as per ChipWhisperer specifications and 64 is an example of reducing the resolution to one-quarter of 256. There are two red lines: the lower line indicates minimum accuracy for a confidence level of 99.9% and the upper line indicates 99.99% confidence level with $3,000$ validation traces. If the accuracy is higher than the upper line, the key is correct at 99.99%.

The hyperparameter settings are listed in Appendix A. According to Figure 18, the result with picture traces has

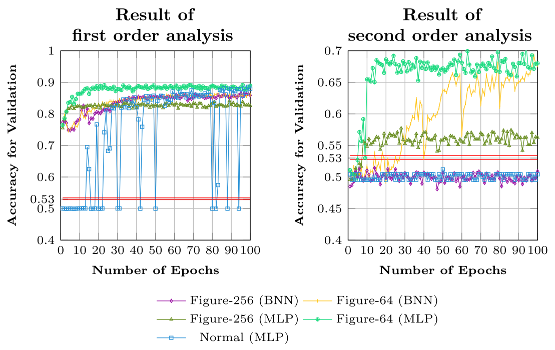


FIGURE 18. Result of first and second order MLP/BNN attacks on CW board.

higher accuracy than that with normal traces regardless of the resolution and attack orders. The result differs with different variables, such as hyperparameters, number of layers, or number of neurons. Therefore, we cannot easily say that the picture traces show better results. For epoch 200, the accuracy of normal traces is 0.9.

The results of the second-order attack are shown to achieve better improvement. The attack accuracy with picture traces is higher than the accuracy obtained with normal traces. Moreover, lower resolution produces a better result, which is also the case in the first-order attack. This is due to the small input size and reduced overfitting, and thus, the lower resolution reduces noise. The attack with normal traces fails even at extremely high epochs, for example, epoch 10, 000. This is an unexpected result because the accuracy of the first-order attack is 0.9 on training epoch 200. As a result, a DL attack with picture traces performs better than that with normal traces measured from the ChipWhisperer. As mentioned previously, we consider a BNN for the attack because our proposed method is better-suited to a BNN architecture. Compared to MLP and CNN schemes, a BNN can reduce the hardware components and improve the performance because all weights and activations can be binarized. More precisely, the BNN outperforms MLP and CNN schemes because most of the 32-bit floating-point multiple accumulations are replaced by 1-bit XNOR-count operations. Our proposed picture-formatted form well fits the BNN scheme without any modification.

The BNN results are quite similar to the MLP results. In terms of first-order analysis, the BNN outperforms the normal trace attacks. Even though Figure-256 (Figure 18) in the second-order BNN analysis cannot find the correct key, the Figure-64 BNN result reveals the correct key. Thus, the BNN results are not better than the MLP results. However, because the learning cost of BNN is significantly less than that of MLP, BNN still has some merit.

B. MLP/BNN BASED DL ATTACK ON ASCAD DATABASE

We use the ASCAD.h5 file, which is well-aligned. To compare other attack performances using ASCAD, we use the same NN settings as those used for CHES 2019 [31], but with 7, 000 traces, which is approximately $\frac{1}{3}$ the number of traces used for CHES 2019. The results of the first- and second-

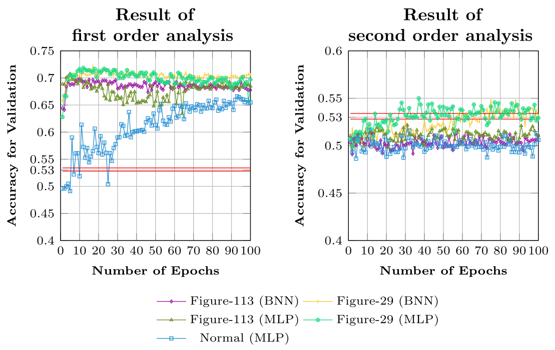


FIGURE 19. Result of first- and second-order MLP/BNN attacks on ASCAD.

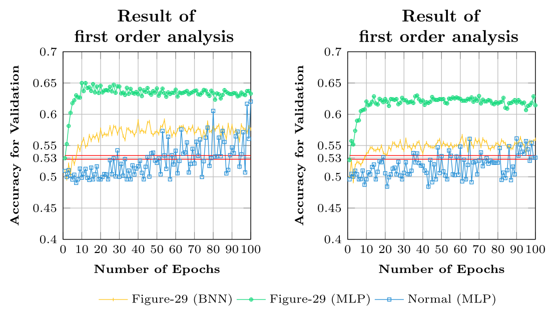


FIGURE 20. Result of first-order MLP/BNN attacks on ASCAD50 (left) and ASCAD100 (right).

order attacks are shown in Figure 19. The encoding resolution is 113, and 29 is quarter of 113 for the same environment with ChipWhisperer. Because we do not know the power trace information, we consider all voltage values in ASCAD.h5. There are 113 different voltage values in integer form from -47 to 66.

In the first-order attack, the minimum epoch with meaningful accuracy is lower with the picture traces than normal traces for both Figure-113 and Figure-29. Epoch 1 is sufficient for obtaining the correct key. This means that the picture traces demonstrate better performance in the first-order attack. Note that the red line in Figure 19 has the same meaning as that of the ChipWhisperer attack environment.

The attack with resolution 29 of picture traces is the best result in the second-order attack. With normal trace in our environmental setting (Appendix A), the attack succeeds in more than the trace epoch 1, 000. Because the minimum number of training epochs in the first-order attack is only five, the non-picture trace is less able to solve relatively difficult problems, such as higher-order attacks. However, picture encoding provides greater learning ability and organizes the inputs and neurons to solve difficult problems.

In addition, the validation accuracy of a BNN attack is quite similar to that of an MLP attack. Even though more neurons and layers are required (Appendix A), the BNN is expected to outperform the MLP because the BNN scheme uses more suitable hardware components [15], [30].

Additionally, we consider the ASCAD50 and ASCAD100 datasets[1] which are usually employed to investigate the

[1] **ASCAD100** (**ASCAD50**) are randomly shifted range between 0 and 100(50) from original **ASCAD**, respectively. Refer to the detailed information in [27].

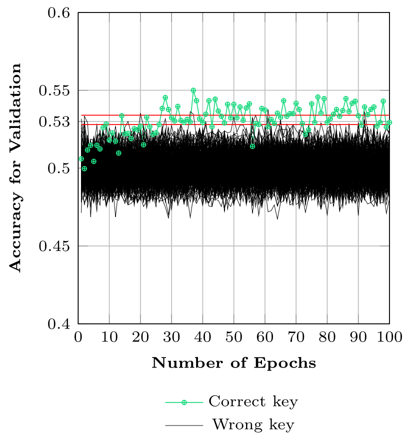


FIGURE 21. Result of second-order analysis for Figure-29 with all key candidates (Validation of 3, 000 traces).

resistance of shuffling countermeasure. Considering the storage, Figure-29 is only performed. Actually, the total points of Figure-29 in ASCAD100 is 5-times than Figure-29 of ASCAD because the fixed 0-value is less than previous one. In other words, the Reducing scheme in III-A3 might be less effective if the shuffling countermeasure is well-employed. However, as stated before, there is some techniques such as single data saving multiple points and the reducing of the resolution, in order to shrink the storage size except for filter scheme in III-A3.

In terms of normal traces of ASCAD50 and ASCAD100, the accuracy is unstable as well as is not higher. Unlike to the original, the result of Figure-29 is clearly stable, even if the accuracy is less than the result of ASCAD. We can also observe that it allows the result of BNN to get the stable in terms of accuracy for validation, after converting the picture-formatted form. Moreover, the BNN outperforms the MLP result of normal form, although MLP of picture-formatted form has beyond the BNN.

C. DIFFERENTIAL DEEP LEARNING ANALYSIS (DDLA)

Rather than employing absolute criteria, the accuracy of 256 candidates can be compared instead and a correct key that has the highest accuracy can be selected. In Figure 21, we show the accuracy values with 3, 000 validation sets of all key candidates for the ASCAD dataset. For the result, some accuracy values for wrong keys are higher than the red line (99.99%); however, this does not occur frequently. Indeed, this method can be used to distinguish the correct key from incorrect keys. The expected attack time is twice than required when using absolute criteria with the red line. In addition, even if one can determine a correct key that has the best accuracy, we cannot determine that the key with best accuracy is correct key because there is no mathematical background. In other words, without applying statistical confidence-level tests, the best accuracy cannot induce the correct key.

The Differential Deep Learning Analysis (DDLA) [31] results for picture traces on ASCAD are shown in Figure 22. Because of overfitting, the training accuracy values are very

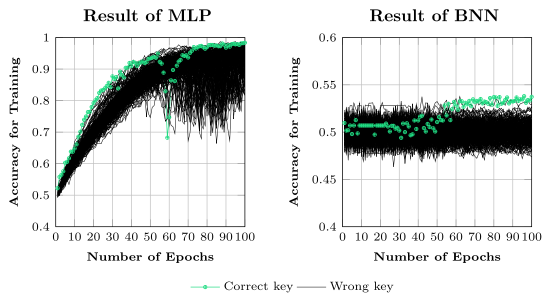


FIGURE 22. Result of second-order analysis for Figure-29 with all key candidates (Training of 7, 000 traces).

high for all key candidates. Interestingly, the correct key tends to demonstrate high training accuracy for each epoch, but does not always demonstrate the highest accuracy. In addition, the results are not statistically significant; therefore, it is not possible to select a single correct key. This may depend on the attacker's heuristic result. Therefore, training accuracy is not a trustworthy criterion; a validation set must be applied for correct key determination.

The key point of DDLA is the difference in the learning aspect between the correct key and wrong keys. According to the BNN result (right side of Figure 22), there is no overfitting of wrong keys because the training accuracy values are not greater than 0.525 for wrong keys. In our experimental results, we could not remove overfitting by any overfitting elimination method, such as L1 regularization, L2 regularization, or drop-out. The BNN approach can eliminate overfitting if one does not have sufficient power trace to separate out a validation set from the collected power traces. BNN is a good solution to fully use power traces and validate the accuracy of key candidates with a training set only, such as DDLA.

V. CONCLUSION

Many researchers have investigated the application of DL for side-channel analysis. Different learning algorithms, such as MLP, CNNs, and autoencoders, can be applied to enhance the performance of side-channel analysis. This paper proposes converting the side-channel information based on an n-th dimension vector into picture form to fully utilize the advantage of DL. The experimental results indicate that, compared to previous schemes using ASCAD and traces from ChipWhisperer board, the validation accuracy is significantly higher and the number of learning epochs required to obtain the secret key can be reduced. In addition, our proposed picture format enables the retrieval of the correct key in second-order DL analysis; however, previous works couldn't recover the correct key in our criteria based on statistical confidence.

As such, our conversion scheme has the potential to enhance side-channel analysis. In the future, the following potential applications will be investigated.

- **Using additional dimension to create depth.** Some picture-formatted traces can be overlapped. For example, if plaintext is the same, then some traces can be over-

TABLE 1. DL Hyperparameter for ChipWhisperer on first- and second-order MLP/BNN attacks.

ChipWhisperer	Normal	Figure-64 (MLP)	Figure-256 (MLP)	Figure-64 (BNN)	Figure-256 (BNN)
Input size	100	1,023	3,690	1,023	3,690
Hidden layer	4				
Neuron	$30 \times 30 \times 30 \times 30$	$80 \times 80 \times 80 \times 80$		$3000 \times 3000 \times 3000 \times 2400$	
Label	MSB Hamming weight				
Optimizer	Adam (default setting)				
Dropout	70% per each layer				
Activation function	Relu, softmax, and one-hot encoding				
Learning rate	0.01	0.001		layer-wise learning rate from 0.01 to 0.01×10^{-4}	
Batch	N/A	200		500	
#Traces	Train: 7,000 / Valid: 3,000				
Initializing	Xavier initialization				

TABLE 2. DL Hyperparameter for ASCAD on first- and second-order MLP attacks.

ASCAD DB	Normal	Figure-29 (MLP)	Figure-113 (MLP)	Figure-29 (BNN)	Figure-113 (BNN)
Input size	700	3,271	10,528	3,271	10,528
Hidden layer	2			4	
Neuron	20×10	200×100		$2000 \times 500 \times 500 \times 500$	
Label	LSB Hamming weight				
Optimizer	Adam (default setting)				
Dropout	N/A	70% per each layer			
Activation function	Relu, softmax, and one-hot encoding				
Learning rate	0.001			layer-wise learning rate from 0.01 to 0.01×10^{-4}	
Batch	1,000	500		100	
#Traces	Train: 7,000 / Valid: 3,000				
Initializing	Xavier initialization				
Normalization	−1 to 1	N/A			

TABLE 3. DL Hyperparameter for ASCAD50 on first-order MLP/BNN attacks.

ASCAD50 DB	Normal	Figure-29 (MLP)	Figure-29 (BNN)
Input size	700	15,944	
Hidden layer	2		4
Neuron	20×10	200×100	$2000 \times 500 \times 500 \times 500$
Label	LSB Hamming weight		
Optimizer	Adam (default setting)		
Dropout	70% per each layer		
Activation function	Relu, softmax, and one-hot encoding		
Learning rate	0.001		layer-wise learning rate from 0.01 to 0.01×10^{-4}
Batch	1000	500	100
Traces	Train: 7,000 / Valid: 3,000		
Initializing	Xavier initialization		
Normalization	−1 to 1	N/A	

lapped by changing the concentration. Therefore, the dot in picture-formatted traces would have a specific value rather than "1". Because picture overlapping does not lose the original information, it differs from the trace integration of normal traces, such as the average.

- **Additional weight to point of interest (PoI).** Similar to the previous potential application, some additional weight on the critical data, represented as a dot, can be provided.
- **Applying additional DL schemes.** Because the target is converted to the MNIST dataset style, additional DL schemes, such as BNN, can be applied and may be more effective, compared to previous side-channel analysis based on DL. It is also impossible that the normal trace adds extra weight on PoI, because the extra weight distorts the original attack models.

APPENDIX A
DL HYPERPARAMETER FOR ALL EXPERIMENTAL RESULTS

The hyperparameter information about our attacks is provided in this appendix.

TABLE 4. DL Hyperparameter for ASCAD100 on first-order MLP/BNN attacks.

ASCAD100 DB	Normal	Figure-29 (MLP)	Figure-29 (BNN)
Input size	700	17,183	
Hidden layer	2		4
Neuron	20×10	200×100	$2000 \times 500 \times 500 \times 500$
Label	LSB Hamming weight		
Optimizer	Adam (default setting)		
Dropout	70% per each layer		
Activation function	Relu, softmax, and one-hot encoding		
Learning rate	0.001		layer-wise learning rate from 0.01 to 0.01×10^{-4}
Batch	1000	500	100
Traces	Train: 7,000 / Valid: 3,000		
Initializing	Xavier initialization		
Normalization	−1 to 1	N/A	

ACKNOWLEDGMENT

The authors would like to thank NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

REFERENCES

[1] M. Abadi *et al.* (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software. [Online]. Available: tensorflow.org

[2] ANSSI. (2018). *Ascad Database*. [Online]. Available: https://github.com/ANSSI-FR/ASCAD

[3] ANSSI. (2018). *Secaes-Atmega8515*. [Online]. Available: https://github.com/ANSSI-FR/secAES-ATmega8515

[4] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems—CHES* (Lecture Notes in Computer Science), vol. 3156, M. Joye and J.-J. Quisquater, Eds. Berlin, Germany: Springer, 2004, pp. 16–29.

[5] C. M. Bishop, *Neural Networks for Pattern Recognition*. New York, NY, USA: Oxford Univ. Press, 1995.

[6] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. New York, NY, USA: Springer-Verlag, 2006.

[7] E. Cagli, C. Dumans, and E. Prouff, "Convolutional neural networks with data augmentation against jitter-based countermeasures," in *Cryptographic Hardware and Embedded Systems—HES* (Lecture Notes in Computer Science), vol. 10529, W. Fischer and N. Homma, Eds. Berlin, Germany: Springer, 2017, pp. 45–68.

[8] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1," 2016, *arXiv:1602.02830*. [Online]. Available: http://arxiv.org/abs/1602.02830

[9] *ChipWhisperer Website*. [Online]. Available: https://newae.com/tools/chipwhisperer/

[10] *Deep Learning Website*. [Online]. Available: https://deeplearning.net/tutorial/tutorial

[11] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *J. Mach. Learn. Res.*, vol. 9, pp. 249–256, May 2010.

[12] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org

[13] G. Hospodar, B. Gierlichs, E. De Mulder, I. Verbauwhede, and J. Vandewalle, "Machine learning in side-channel analysis: A first study," *J. Cryptograph. Eng.*, vol. 1, no. 4, p. 293, 2011.

[14] B. Hettwer, T. Horn, S. Gehrer, and T. Güneysu, "Encoding power traces as images for efficient side-channel analysis," 2020, *arXiv:2004.11015*. [Online]. Available: http://arxiv.org/abs/2004.11015

[15] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Proc. Adv. Neural Inf. Process. Syst. NIPS*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds., 2016, pp. 4107–4115.

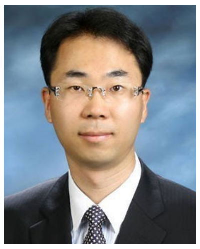
[16] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 1666, M. J. Wiener, Ed. Berlin, Germany: Springer, 1999, pp. 388–397.

[17] J. Kim, S. Picek, A. Heuser, S. Bhasin, and A. Hanjalic, "Make some noise. Unleashing the power of convolutional neural networks for profiled side-channel analysis," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2019, no. 3, pp. 148–179, May 2019.

[18] Y. Lecun and Y. Bengio, *Convolutional Networks for Images, Speech, and Times-Series*. Cambridge, MA, USA: MIT Press, 1995.

[19] L. Lerman, G. Bontempi, and O. Markowitch, "A machine learnig approach against a masked AES," *J. Cryptograph. Eng.*, vol. 5, no. 2, pp. 123–139, Jun. 2015.
[20] Y. LeCun, C. Cortes, and C. J. C. Burges. *The Mnist Database of Handwritten Digits*. [Online]. Available: http://yann.lecun.com/exdb/mnist/
[21] L. Lerman, R. Poussier, G. Bontempi, O. Markowitch, and F.-X. Standaert, "Template attacks vs. Machine learning revisited (and the curse of dimensionality in side-channel analysis)," in *Constructive Side-Channel Analysis and Secure Design (COSADE)*, S. Mangard and A. Y. Poschmann, Eds. Cham, Switzerland, 2015, pp. 20–23.
[22] L. Masure, C. Dumans, and E. Prouff, "A comprehensive study of deep learning for side-channel analysis," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2020, no. 1, pp. 348–375, 2020.
[23] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking cryptographic implementations using deep learning techniques," in *Security, Privacy, and Applied Cryptography Engineering—SPACE* (Lecture Notes in Computer Science), vol. 10076, C. Carlet, M. A. Hassan, and V. Saraswat, Eds. Berlin, Germany: Springer, 2016, pp. 3–26.
[24] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," Tech. Rep., Nov. 2015.
[25] G. Perin, Ł. Chmielewski, and S. Picek, "Strength in numbers: Improving generalization with ensembles in machine learning-based profiled side-channel analysis," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, pp. 337–364, Aug. 2020.
[26] S. Picek, A. Heuser, A. Jovic, S. Bhasin, and F. Regazzoni, "The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, pp. 209–237, Nov. 2018.
[27] E. Prouff, R. Strullu, R. Benadjila, E. Cagli, and C. Dumans, "Study of deep learning techniques for side-channel analysis and introduction to ASCAD database," Cryptol. ePrint Arch., Tech. Rep. 2018/053, 2018. [Online]. Available: https://eprint.iacr.org/2018/053
[28] J. Y. Park, D. G. Han, D. Jap, S. Bhasin, and Y. S. Won, "Non-profiled side channel attack based on deep learning using picture trace," Cryptol. ePrint Arch., Tech. Rep. 2018/053, 2019. [Online]. Available: https://eprint.iacr.org/2019/1242
[29] F.-X. Standaert, T. G. Malkin, and M. Yung, "A unified framework for the analysis of side-channel key recovery attacks," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, Berlin, Germany: Springer, 2009, pp. 443–461.
[30] T. Simons and D.-J. Lee, "A review of binarized neural networks," *Multidisciplinary Digit. Publishing Inst. (MDPI) Electron.*, vol. 8, no. 661, pp. 1–25, 2019.
[31] B. Timon, "Non-profiled deep learning-based side-channel attacks with sensitivity analysis," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2019, no. 2, pp. 107–131, Feb. 2019.
[32] L. Wu and S. Picek, "Remove some noise: On pre-processing of side-channel measurements with autoencoders," Cryptol. ePrint Arch., Tech. Rep. 2019/1474, 2019. [Online]. Available: https://eprint.iacr.org/2019/1474.pdf
[33] G. Zaid, L. Bossuet, A. Habrard, and A. Venelli, "Methodology for efficient CNN architectures in profiling attacks," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, pp. 1–36, Nov. 2019.
[34] W. Feller, "On the normal approximation to the binomial distribution," *Ann. Math. Statist.*, vol. 16, no. 4, pp. 319–329, Dec. 1945.

YOO-SEUNG WON received the master's and Ph.D. degrees from Kookmin University, South Korea, in 2014 and 2018, respectively. He is currently a Research Scientist with the Physical Analysis and Cryptographic Engineering (PACE) Lab, Temasek Laboratories @ NTU, Singapore. Before NTU, he held position of Staff Engineer in foundry business of Samsung Electronics Company Ltd., South Korea, in 2018. His research interests include embedded security, secure boot solution, cold boot attack, side-channel analysis and fault attack schemes and countermeasures, practical laser/EM fault injection, and deep learning security.



DONG-GUK HAN received the B.S. and M.S. degrees in mathematics and the Ph.D. degree (engineering) in information security from Korea University, Seoul, South Korea, in 1999, 2002, and 2005, respectively. He was a Postdoctoral Researcher with Future University Hakodate, Japan. He was an Exchange Student with the Department of Computer Science and Communication Engineering, Kyushu University, Japan, from April 2004 to March 2005. From 2006 to 2009, he was also a Senior Researcher with the Electronics and Telecommunications Research Institute, Daejeon, South Korea. He is currently a Professor with the Department of Information Security, Cryptology, Mathematics, Kookmin University, Seoul. He is also a member of KIISC, IEEK, and IACR.

DIRMANTO JAP received the Ph.D. degree in mathematics from NTU, in 2016. He is currently a Research Scientist with the PACE Lab, Temasek Laboratories @ NTU, Singapore. His research interests include physical attacks (side-channel and fault attacks) and countermeasures, practical laser/EM fault injection, application of machine learning and deep learning for side-channel attacks and hardware Trojan detection, and security of deep learning.

SHIVAM BHASIN received the master's degree from Mines Saint-Etienne, France, in 2008, and the Ph.D. degree from Telecom Paristech, in 2011. He is currently a Senior Research Scientist and a Programme Manager (Cryptographic Engineering) with the Centre for Hardware Assurance, Temasek Laboratories @ NTU, Singapore. Before NTU, he held position of Research Engineer with the Institut Mines-Telecom, France. He was also a Visiting Researcher with UCL, Belgium, in 2011, and Kobe University, in 2013. His research interests include embedded security, trusted computing, and secure designs. He has coauthored several publications at recognized journals and conferences. Some of his research now also forms a part of ISO/IEC 17825 standard.

JONG-YEON PARK received the master's degree in mathematics from Kookmin University, in 2012. He was a Researcher with the Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea, from 2012 to 2014. He was also a Research Engineer with the Korea Telecom (KT) Convergence Laboratory, Seoul, South Korea, from 2015 to 2017. He is currently a Staff Engineer with Samsung Electronics Company Ltd., System LSI. His research interests include cryptographer's topics, especially mathematical structures related in secure algorithms and SCA.