Non-Profiled Side Channel Attack based on Deep Learning using Picture Trace

Yoo-Seung Won[1] and Jong-Yeon Park[2]

[1] Kookmin University, South Korea, mathwys87@kookmin.ac.kr
[2] Samsung Electronics, South Korea, jonyeon.park@samsung.com

Abstract. In this paper, we suggest a new format for converting side channel traces to fully utilize the deep learning schemes. Due to the fact that many deep learning schemes have been advanced based on MNIST style datasets, we convert from raw-trace based on float or byte data to picture-formatted trace based on position. This is induced that the best performance can be acquired from deep learning schemes. Although the overfitting cannot be avoided in our suggestion, the accuracy for validation outperforms to previous results of side channel analysis based on deep learning. Additionally, we provide a novel criteria for attack success or fail based on statistical confidence level rather than rule of thumb. Even though the data storage is slightly increased, our suggestion can completely be recovered the correct key compared to previous results. Moreover, our suggestion scheme has a lot of potential to improve side channel attack.

Keywords: Non-profiled side channel attack · Deep learning · Multi-layer perceptron · Convolutional neural network

1 Introduction

In many fields, the machine learning schemes such as MLP (Multi-Layer Perceptron) and CNN (Convolutional Neural Network) are widely utilized to improve their final goals. One of goals in side channel analysis is to retrieve the secret key. It means the machine learning schemes can be naturally applied to side channel analysis. More precisely, the input and output of MLP can be mapped to the points of traces and the guessed intermediate variable, respectively. Afterwards, forward and backward propagations are performed to produce induced output. Especially, CNN scheme is usually used to image recognition, utilizing the additional methodology such as convolution and pooling layers. This scheme is usually applied to avoid alignment and/or compression issues, utilizing pooling system. Moreover, the machine learning schemes such as autoencoder are sometimes used to reduce the noise level. To improve attack performance in side channel analysis, machine learning schemes have been naturally applied to side channel analysis with only minor considerations such as input and output vectors. However, to our best knowledge, there is no improvement in order to make the best use of internal processes in machine learning scheme. That is, like CNN scheme, it is required to fit the side channel analysis when applying machine learning schemes. As previously mentioned, in order to recognize image, the concept of additional layers was developed. As a result, the maximal result can be provided when the processing system of image recognition in humans is matched with machine learning scheme.

In this paper, we suggest *trace-transform* concept to completely utilize the principle of machine learning schemes. Until now, many papers suggest side channel analysis simply connected with machine learning scheme. However, we modify *trace form* to completely use the properties of MLP scheme.

1.1 Motivation

The studies in side channel analysis have only concentrated on how to apply machine learning scheme without any modifications in properties of machine learning scheme. As previously stated, we transform the raw-trace to MNIST style, utilizing the properties of machine learning scheme completely. More precisely, it is inspired from [LCB] as below Figure 1. It is represented to value '7' using hexadecimal value. Of course, although single point in MNIST data is represented to hexadecimal value, it is a float value to represent the concentration of figure. Without the concentration, value '7' can be represented to right of Figure 1. Afterward, we describe why we remove the concentration. Above all, this MNIST style is significantly helpful to learn the data in terms of side channel analysis.

We expect that the best performance can be acquired if the trace is converted to MNIST style, owing to the fact that machine learning schemes such as MLP and CNN have been advanced based on MNIST style datasets.

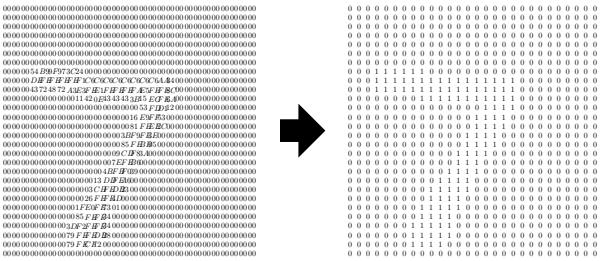

Figure 1: 7 value represented in MNIST dataset

1.2 Our Contribution

In this paper we introduce a novel side channel analysis scheme to apply completely properties of machine learning schemes in non-profiling scenarios. We induce the imagification for trace to enhance the recognition to fully utilize the properties of machine learning schemes. More precisely, the data form is based on position (a.k.a. picture-formatted trace) rather than is more readable than float or byte value (a.k.a. n-dimension vector trace), as if the person recognize a figure. This suggestion allows the significantly reduced trace to retrieve the correct key, compared to previous results. Additionally, we provide an example to describe why our suggestion outperforms classical data by intuitive ways. Similar to [Tim19], without information about countermeasure in given power traces, we demonstrate that our suggestion can retrieve the correct key as well as outperform previous results. Also, in non-profiling scenarios, we apply a novel metric based on statistical confidence level to distinguish the attack result of success or fail. All experimental results are supported by the ASCAD database [PSB+18] and the ChipWhisperer-Lite board [CWw]. As a result, our suggestion outperforms n-dimension vector trace in terms of first-order and second-order DL attacks. Moreover, in second-order DL attack of ASCAD database, only picture-formatted traces can be retrieved the correct key, although

2 Preliminaries

Deep learning (DL) is a particular kind of machine learning which produces great successful results based on deep neural networks. It also has been easily applied to various fields such as image recognition and speech recognition. In this section, we briefly describe how to apply deep learning such as MLP and CNN to side channel analysis.

2.1 Multi-Layer Perceptron

The general objective of MLP is to classify some input vector $\mathbf{x} \in \mathbb{R}^D$ based on its labels $l(\mathbf{x}) \in \mathcal{L} = \{l_1, l_2, \cdots, l_{|\mathcal{L}|}\}$, where D is the dimension of the input data to categorize and $\mathcal{L}$ is the set of classification labels. A goal of neural network (NN) is to produce a function $\mathbf{NN}$: $\mathbb{R}^D \rightarrow \mathbb{R}^{|\mathcal{L}|}$ that takes as input vector $\mathbf{x} \in \mathbb{R}^D$ and output as vector $\mathbf{y}$ of scores. In other words, the final goal is to make the score vector $l(\mathbf{x})$ based on $\mathbf{NN}(\mathbf{x})$, updating the internal properties. In general, the neural network is composed of 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 has to be compared to hypothesis intermediate variable such as Hamming weight model to learn expected result. Moreover, hypothesis intermediate variable is usually encoded to one-hot encoding scheme.

2.1.1 Construction of MLP

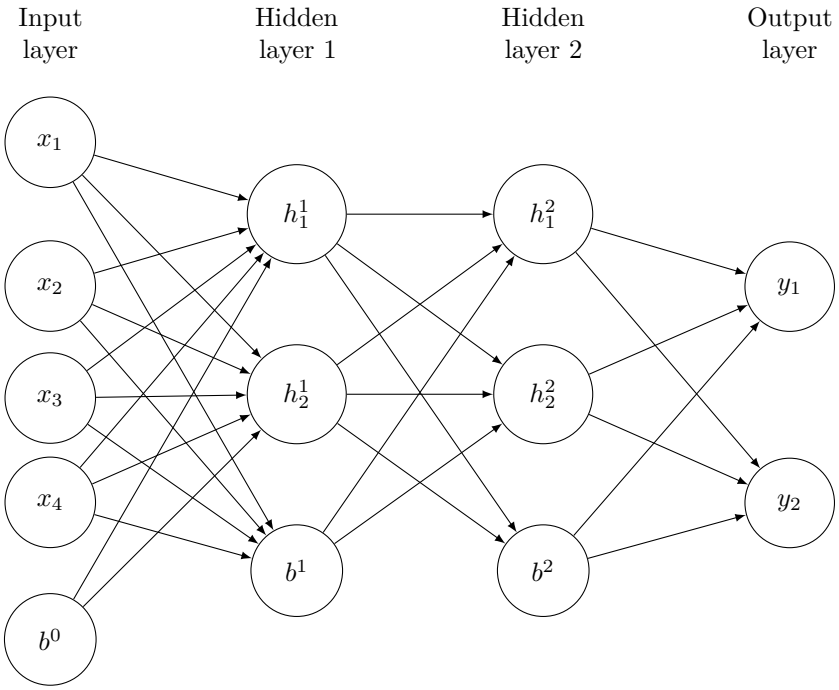

Figure 2: Example for a simple MLP

We provide an example for a simple MLP to enhance understanding. The number of inputs, also known as the points, is 4 and Figure 2 consists of two hidden layers including 3 neurons and output layer has two neurons. Moreover, bias neuron is included to all layers excluding output layer. All arrows in Figure 2 are called as weights represented to Figure 3. The value of weights and bias is usually initialized to Xavier scheme and normal distribution before performing deep learning. In order to acquire the expected result in output layer, some operations are performed to get output in each neurons, which is called as forward propagation. However, a learning is not even started since the weights are not changed to gain the expected result. By backward propagation, the machine learns weights and biases to right way for the expected result. This is done by comparing between the expected result and the result of output layer. The expected result is sometimes encoded

to one-hot encoding. If the one-hot encoding is applied to single bit Hamming weight model, the output layer should be composed of two neurons. If Hamming weight value is 1, then the expected result is encoded to $[1, 0]$ or $[0, 1]$. More precisely, by comparing between $[y_1, y_2]$ and one-hot encoded value, the backward propagations is performed to update all weights.

2.1.2 Forward Propagation

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

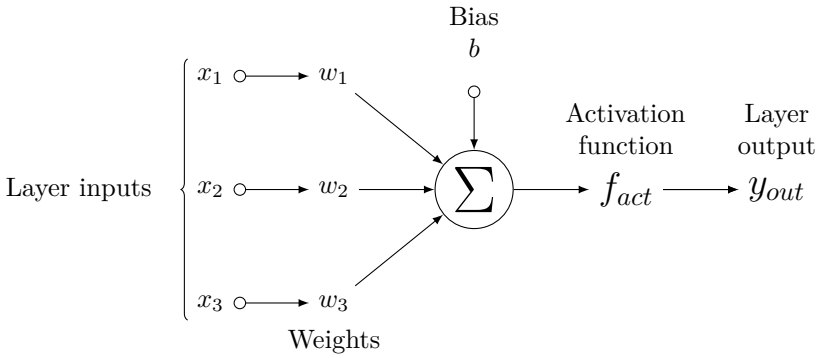

Figure 3: Calculation for a single neuron in hidden and output layers

2.1.3 Backward Propagation

The core of MPL is backward propagation, owing to the fact that each weights can be updated to learn the expected result. The backward propagation is performed by comparison between the expected result $l(\mathbf{x})$ and the output of MPL $\mathbf{NN}(\mathbf{x})$, the backward propagation can be performed. For comparison, the error function such as Euclidean distance could be used to learn the expected result. Normally, one can define an error function $E : \mathbb{R}^D \to \mathbb{R}$ for example as the Euclidean distance between the output of MPL and the one-hot encoded the label as below:

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

where $l(\mathbf{x})[i]$ and $\mathbf{NN}(\mathbf{x})[i]$ indicate respectively each label value and output neuron value. The value of error function represents the gap between the expected result and MLP output. That is, backward propagation allows to narrow the gap. To reflect the error for all train data $\mathbf{X} = (\mathbf{x}_i)_{1 \le i \le T}$, a loss function is defined as below.

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

The weights can be updated by Gradient Descent technique which is applied to the loss function $\mathcal{L}$. Because the variables in loss function are based on weights, the weights

can be trainable. Here, we denote this concept as $\mathcal{L}_{\mathsf{w}}$. In other words, based on Gradient Descent technique, the weights in loss function can be updated with $\nabla\mathcal{L}_{\mathsf{w}}$. Utilizing t-th result $\mathcal{L}_{\mathsf{w}(t)}$, $(t+1)$-th weights can be learned as below.

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

where α indicates a learning rate. The total quantity for training T is consumed to learn w. However, even though the T quantity is used, it is possible to re-use the T quantity to learn w, which is called as Epoch.

For side channel analysis, the deep learning is simply applied as below.

- **Input layer** The points of a trace correspond to the number of input values.
- **Output layer** The adversary sets to the expected result. If the target intermediate variable for attacks is Hamming weight of a single bit of S-box output, then the output layer has two neurons where one-hot encoding is applied.
- **Hidden layer** Except for input layer and output layer, the intermediate layers can be applied to deep learning in order to learn more accurate result than single layer neural network. This hyperparameter significantly depends on rule of thumb, which means the adversary carefully choose the parameter with trial and error experience.
- **Learning rate** To update the weights, it is a ratio to utilize the previous training result. The range is between 0.0 and 1.0.
- **Activation function** To activate each of neurons, the activation functions such as sigmoid, tanH, ReLU, and softmax can be applied. Empirically in many researches, the ReLU is applied to hidden layers and the softmax is employed to output layer.
- **Initialization** It is basic setting of initialing values of weights and baises. Simply, the values can set random variables in Gaussian distribution or one can use additional techniques.

2.1.4 Additional Functions

The additional functions are used to avoid some obstacles such as overfitting and memory efficiency in machine learning schemes. There are many additional skills, but we explain two techniques that is used in this paper.

- **Batch size** The total number of training divides into number of batches or sets or parts. It can avoid the overfitting, updating the weights based on batch size.
- **Dropout** The key idea is to randomly drop units from the neural network during training. That is, a part of connections only updated. The range is between 0.0 and 1.0. For example, 1.0 means no drop.

3 Picture Trace and Analysis

In this section, we introduce a new method of pre-processing for power traces and basic attack principle. In general power trace, if an event occurs on one time, there is a voltage value that has one float point value. It is 1-dimension 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 (a.k.a. n-time).

$$v(trace) = (2.34, 1.36, 2.50, 2.97, ...)$$

This means n-dimension vector that has rational number on n-time. The floating with n-dimension vector is described as below, it is normal power trace; x-axis is time, y-axis is a voltage.

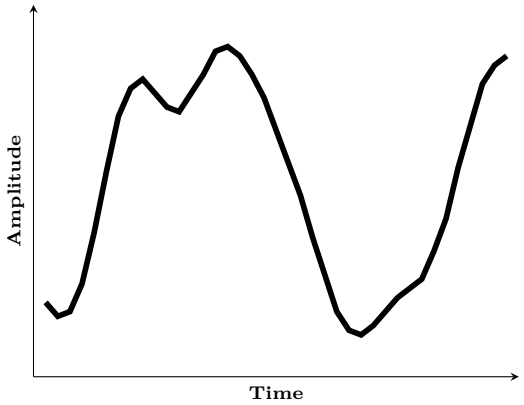

Figure 4: A trace for side channel analysis

Figure 4 shows that voltage on the time express "degree", power trace can be visualized for human's recognition. People recognize power flow through upper marking that means high degree of voltage and lower marking that means low degree. However real voltage flow is not the figure but only n-dimension vector sequence. This is not a picture that is expressed as picture.

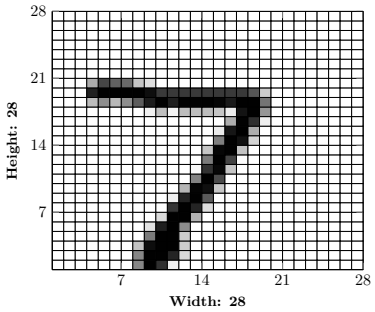

Figure 5: 7 value represented in MNIST database with concentration

We perform side channel analysis with picture-formatted power traces. Surely, there are many methods that convert n-dimension vector sequence to a picture. We get hints from MNIST database; this is simple way to make picture from power traces. Figure 5 shows handwriting number "7". We can perceive the value through some values on 28×28 area. There are visually unnecessary area treats value "0" that means NULL. We would like to make power trace as this method. Figure 6 shows a power trace that we convert as a picture. This is fundamental shape that we want to change vector sequence to a picture. The meaningful line is 1 and 0 means nothing like foundation that helps only the relative location of point "1". We call it a picture trace.

Specific steps for a picture trace is described below.

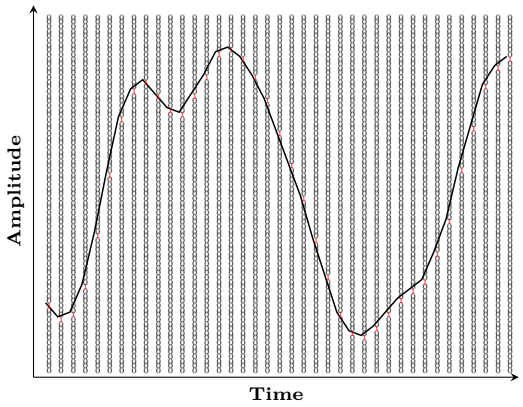

Figure 6: A picture-formatted trace for side channel analysis

3.1 Picture encoding

3.1.1 Choice of resolution

The resolution determines how the picture trace is made exquisitely. To get all information from original power trace, we have to know original trace's resolution when the trace is collected from collecting device such as oscilloscope. Otherwise, we can brute force search to find resolution. Figure 7 shows determination of resolution for conversion power trace to a picture trace. The original trace is n-dimension vector space whose size is the number of point, but the size of a picture trace is the number of point $\times$ resolution. Therefore, a picture trace has many bigger dimension.

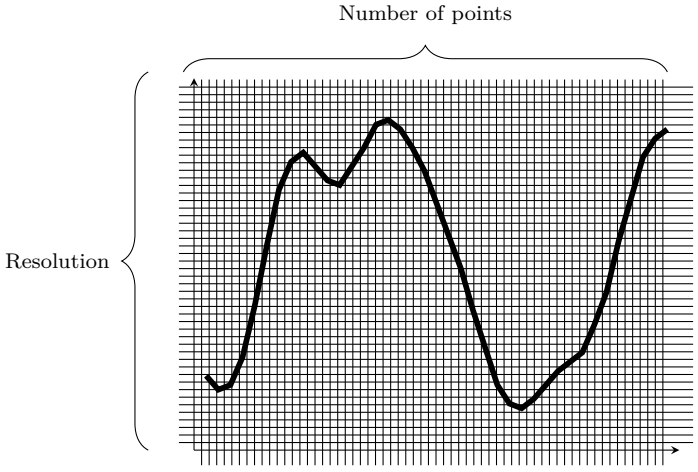

Figure 7: Determination of resolution for a trace

We can choose small resolution. The small resolution picture trace has small input size, it helps learning faster and reduce memory consumption. However, some of information from original trace is missed. It can distort the real information of sensitive data. Detail contents will be discussed.

3.1.2 Drawing

The drawing is the main step when generating a picture trace. To draw a picture, we need to determine a position of a point and put a dot on fixed space. The related position of a point is determined by upper bound, lower bound and resolution that we already chose. First, a gap which difference between lower bound and upper bound is need to be computed. All gaps in power traces are different, however conversion of all trace have to be same for the same standard of analysis and simplicity of implementation.

$$\mathsf{G}ap = upperbound - lowerbound \tag{4}$$

Upper bound is maximum voltage value and lower bound is minimum value of all sample power traces. After computation boundary, one compute related location. Let resolution be n, a voltage value on specific time be v, then related location is computed as formula $t = \varphi(v)$that is intuitive concept.

$$\varphi : R \to Z_n \tag{5}$$

$$t = \varphi(v) = \left\lfloor \frac{v - lowerbound}{Gap} \right\rfloor \times n \leq n \tag{6}$$

Finally, one can make n(resolution)-th vector with related location t. The method to express picture follows MNIST, but there is no depth in our power trace. Therefore, we express number "1" for the related location, "0" means "NULL". Let the related location be t, n-th vector is computed as below.

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

Final picture of each trace is set of vectors $\sigma(t)$. the picture is $\{\delta_1(t_1), \delta_2(t_2), \ldots, \delta_m(t_m)\}$

3.1.3 Reducing

The reducing step is elimination of unnecessary portion in picture. The new-made picture shaped trace has unchanged "1" to "0" or "0" to "1". This section has no effect on power analysis. In our experimental result, most of unchanged section of the pictures is "0" that is generally foundation. One cannot know unchanged section from 1 trace. Thus, one makes a filter to pit the picture traces through a sieve to sift out the unchanged portion. The Filter is made by stacking all picture traces, and choose vectors having value "1".

Figure 8 shows the method to make the filter. The filter eliminates unchanged "0", as a result one can get only meaningful locations, see Figure 9. Output picture right side of Figure 9 is inside of filter boundary.

In our result, filtering picture traces mostly has approximately 90% input size reducing of ASCAD example. Reducing input size makes it possible to be efficient learning time and reduce training epochs.

3.2 Correct Key determination

For correct key determination, the previous non-profiling attack published in CHES 2019 [Tim19] choose best training speed, compared to other wrong keys. However, this method assumes that deep learning with correct key find more efficient way of the provided labels. This means weights and bias with higher training accuracy can compute correct labels about any power traces. We have known that high accuracy of training trace does not

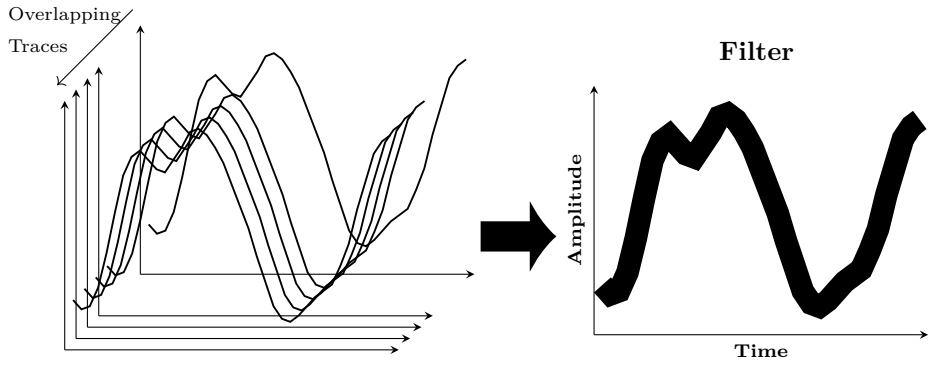

Figure 8: Overlapping traces to generate the filter

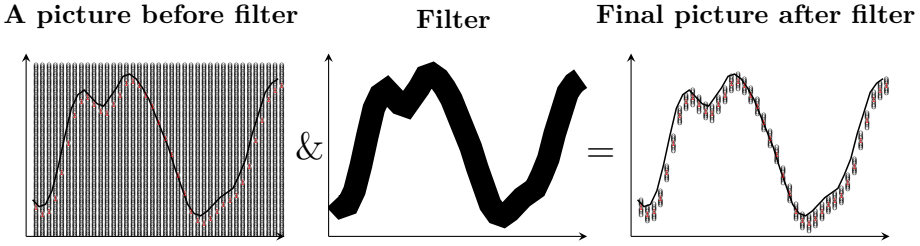

Figure 9: Generating final figure trace based on the filter

guarantee find correct key according to lots of experimentations. This is because deep learning can fall into an overfitting trap. In some of result, wrong keys have faster learning speed even though removal overfitting techniques are adjusted in the experimentations. Thus, we need to make other standards for determination of correct key. The reason that overfitting is hard to remove will be discussed in the picture traces. It should be looked at in an isolated topic. We use test set for the final decision of correct key or wrong keys. The test set is not used for learning phase, actually we need more power traces to check correct key. In addition, Validation set must be uniformly distributed about each label. For example, 1bit labeling has 50% "1" and 50% "0". If labeling is not uniformly distributed, some cases of wrong learning, for example pick all "1" as result, has high accuracy that is statically meaningful. Let test set has n power traces that has perfectly half 0, and half 1. Bernoulli trials $X_1, X_2, ..., X_n$ are independent. All Bernoulli trials with success probability $\frac{1}{2}$, then their sum is distributed according to a binomial distribution with n and $\frac{1}{2}$

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

If n is large enough, $\mathcal{B}(n,p)$ is given normal distribution

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

For example, a case that n equals 3000, binomial distribution is regard as normal distribution. Therefore, distribution follows $\mathcal{N}(1500, 7500)$, one can calculate the error possibility selected key is correct key with probability density function; 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 quires of key= 0x33, $\mathcal{P}\left(1600,\ 1500, \sqrt{750}\right) = 0.999869$. One can insist the desired correct key is 0x33, and error rate is $1 - 0.999869 = 0.000131$.

3.3 Characteristics of Picture Trace

3.3.1 Easy learning

A notable difference of picture traces is that alteration of voltage for each trace causes changing vector location. For example, in Figure 10, there are only 2 power traces; it has 3 voltage values on 3 times. The left side of Figure 10 shows normal traces that is 3 dimension vector sequence. The first traces' third value is 0.1 and 0.2 in the second trace. The right side of Figure 10 shows the picture trace. Input vectors of the picture trace is $[0, 1, 0, 0, 0, 1, 1, 0, 0]$ in the first trace, the second trace has $[0, 1, 0, 0, 0, 1, 0, 1, 0]$. Alteration of 2 traces is a location change expressed by $[1, 0]$ to $[0, 1]$; normal trace's alteration is 0.1 to 0.2.

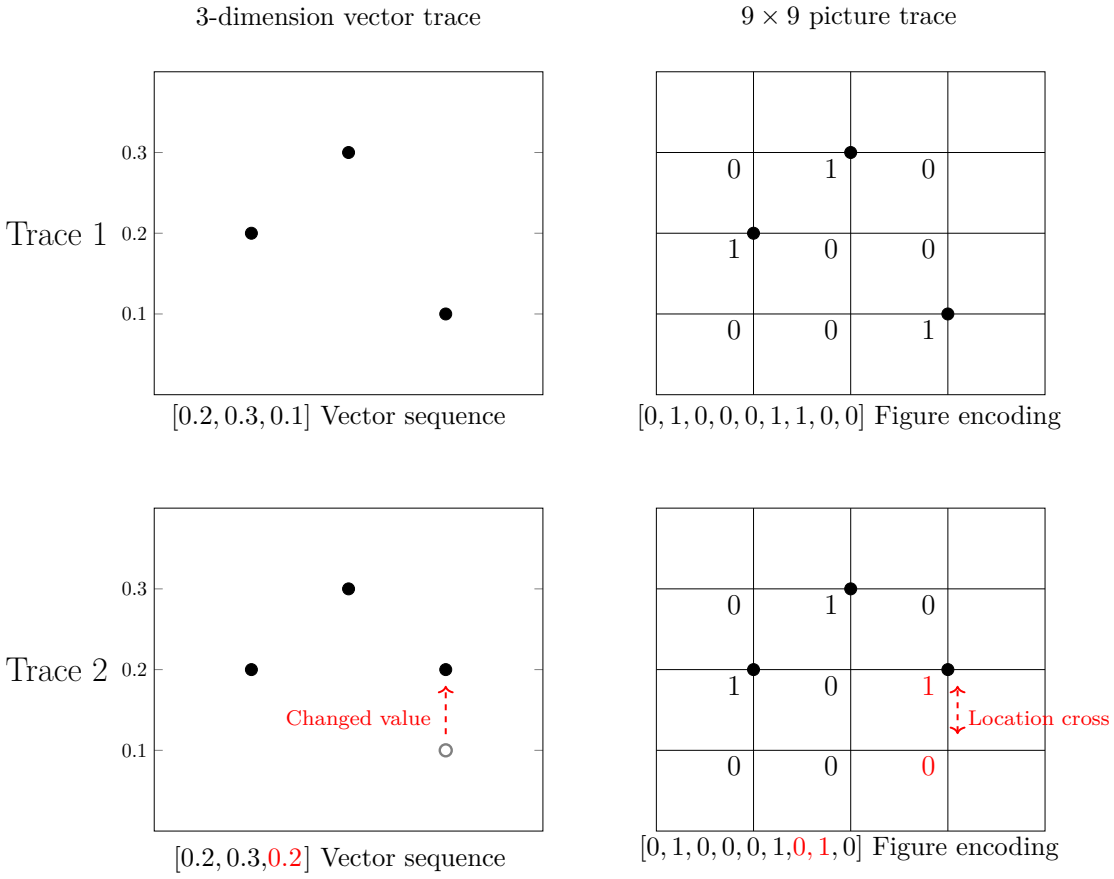

Figure 10: Example of easy learning

By this characteristic, it is easier to learn for neural network fitting with weight and bias. Figure 11 shows intuitive appearance of learning in normal traces. At the last layer of the neural network with softmax and one-hot encoding, 2 traces in voltage value 1 or 2. The last step of network select location of $0P_1$ or P_2 that is bigger value after softmax. Because softmax is only for normalizing the final value from 0 to 1, upper neuron is selected when $x_i w_1 + b_1$ is bigger than $x_i w_2 + b_2$, lower neuron is selected on the contrary to this. If the learning process is well done, the neural network computes different choice that depends on input 1 or 2. Consider that input 1 makes network upper choice and input 2 makes network lower choice in 11. For the desired result, $w_1 + b_1 > w_2 + b_2$ when 1 is the input, $2w_1 + b_1 < 2w_2 + b_2$ when 2 is the input at the same time. These 2 inequalities

share the same variables, but have opposite inequality signs. (Of course it has the answer, please try to find the solution). However intuitively, most of cases are not satisfied with inequalities, the machine starts to find the solution with increasing epochs.

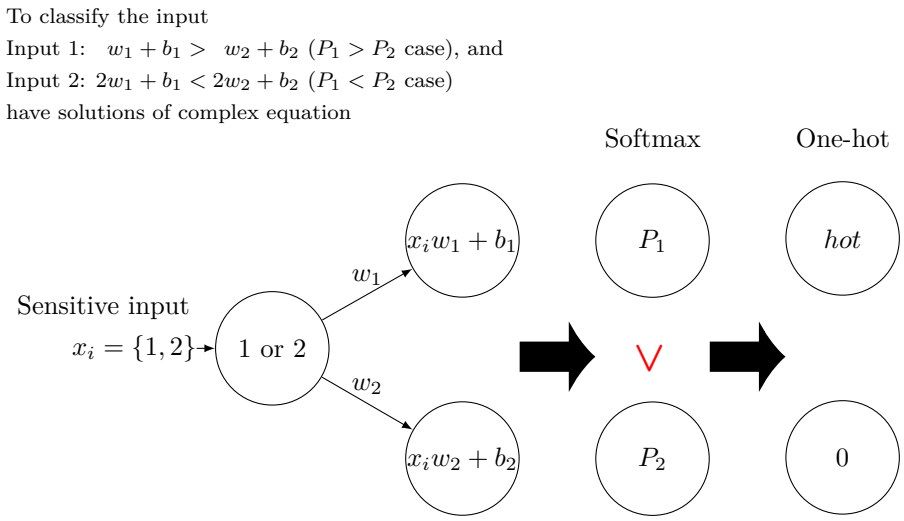

Figure 11: Intuitive appearance of final layer in normal traces

On the other hand, Figure 12 input is not 1,2 but (1,0), (0,1) that is the picture trace example. This case need 4 weights w_{11}, w_{12}, w_{21}, and w_{22}. As the same way to make solution with 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, it is relatively easy to find solution, for example, $w_{11} > w_{21}$ and $w_{12} < w_{22}$.

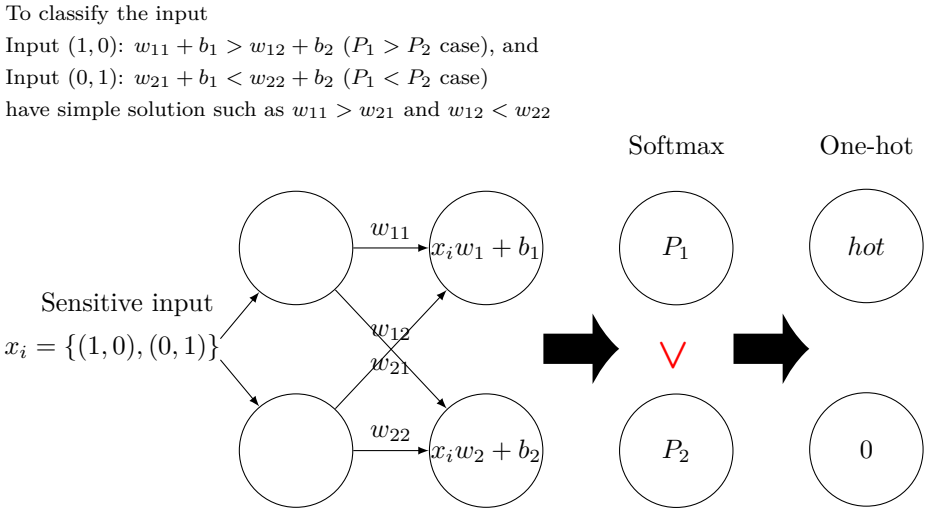

Figure 12: Intuitive appearance of final layer in figure traces

At intermediate layers, there is no choice function, weight and bias influence to next layer's inputs. Figure 13 is a model of which a input affects multiple neurons. This is too simple model; this example just helps intuitive way expanding of huge size of neurons. Assume that input is 0.5 or 0.6. output of n_1 is $0.5w_1 + b_1$, $0.6w_1 + b_1$ or 0 computed by

ReLU activation. Even if influence factor is very huge about n_1 that depends on only w_1 and b_1. This means that influence does not care about input values. n_2 is computed by w_2 and b_2, it is in the same situation. Neural network give influence the input xi, that is not about the value 0.5 or 0.6. It is the limitation of network solve the very complex equations. This is because every input 0.5, 0.6, 0.7, and 0.8 have the same output.

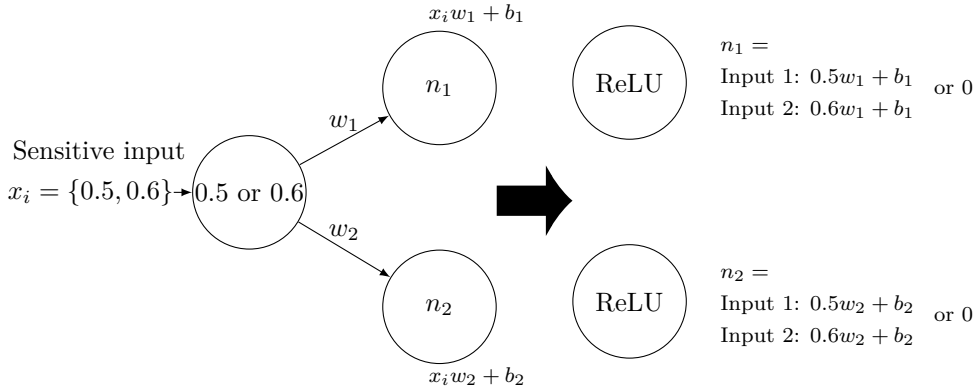

Figure 13: Intuitive appearance of hidden layer in normal traces

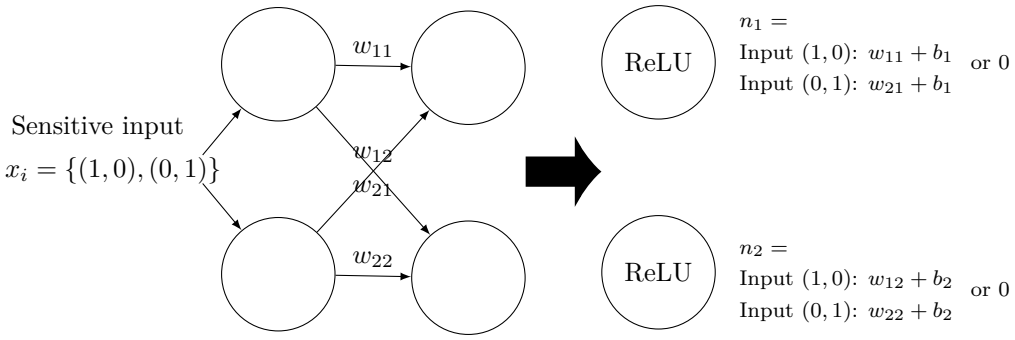

Figure 14: Intuitive appearance of hidden layer in figure traces

Figure 14 shows intermediate layer of a picture trace. In common with the last layer example in Figure 12, (1,0) or (0,1) matches different weight w_{11}, w_{12}, w_{21}, and w_{22} independently, see Figure 14. This affect next layer's neurons more than normal inputs in Figure 13. Inputs $(0, 1)$ or $(1, 0)$. The result of next neurons depends on valuables that is connected inputs $(0, 1)$ or $(1, 0)$. Therefore, the difference of result of the next neurons is 0 to infinity theoretically. In side channel attack with Hamming weight, 1-bit difference Hamming weight has an effect with huge difference by inputs.

3.3.2 Overfitting

According to attack models by attackers, most of voltage information of a power trace for side channel attack is simple noise that is not used to find keys. Large inputs do not mean

much information. Bigger input size has bigger noise. With normal trace, bigger noise makes learning slow; noise interferes with success to find solution. It is simple mapping rule.

$$\text{Bigger noise} = \text{learning slow}$$

However, the picture trace makes learning easy with lots of weight, bias and input size. This is finding simultaneous equations with many variables with only one simple equation. That is a lot of possible solutions but only one solution is right. In this case, deep learning produces a network with weights and biases but that is wrong answer. In addition that there is much noise in power traces for side channel attack in some attack models. Finally, we can get another simple mapping rule.

$$\begin{gathered}\text{Bigger noise} \\ + \text{ High learning ability} \\ + \text{ a lot of inputs and follwing weight and bias} \\ = \text{overfitting}\end{gathered}$$

So, there are simple reasons that deep learning with a picture trace has more possibility of overfitting.

1) A lot of noise (some of attack model in side channel attack)
2) a lot of inputs and followed weight and bias
3) high learning ability

Overfitting is a characteristic of analysis with picture traces. There have been already many researches for reducing overfitting such as using enough traces, L1, L2, regularization, drop-out or early stop. In our research, we adjust only "early stop" and determine correct key with validation with test set. This is because we do not need to perfect network to find key like image-net. The purpose of deep learning here is finding key in only this environment. Therefore, related research will be future work.

4 Experimental Results

In this section, we show experimental results. Experimentation is fulfill with Chipwisperer lite [CWw] and ASCAD database [PSB+18]. The setting of resolution of picture traces is maximum resolution, and quarter size of maximum resolution. The reducing resolution has a risk of information distortion. However, it makes learning faster and remove overfitting slightly caused by diminishing number of weights related in inputs of the first layers. The only difference between the first order attack the higher order attacks is the label. If one uses power traces that has sensitive data masked, the label will be generated by the pure data. If masking value is known, one can set label with unmasked values. Thus, our first order attack uses unmasked labels with known making values. Also, the second order attack uses exactly the same power traces, and sets labels S-Box outs without masking. The machine with assumption of second order attack would solve harder problems.

4.1 Attack on Chipwisperer Lite

Basic specification of Chipwisperer lite described in Appendix B. The target operation is AES Subbyte; target data is S-box output. The order of S-box is the fourth. The data is randomized by exclusive-or with random 8bit masking. We check CPA on the power traces in order that it is impossible to find key from the first order attack. Figure 15 is the result of CPA. There are two correlation coefficients line on 100 points. The left line is

analysis of S-box output with known masking which is the first order CPA attack on S-box output and the right line is CPA with masking value. We make simple sample power traces that has only 4th Subbyte operation and masking value in 100 points. Each correlation coefficient is formed up to $0.8 \sim 0.9$. One can easily expect that the first order analysis may be well done with deep learning attack.

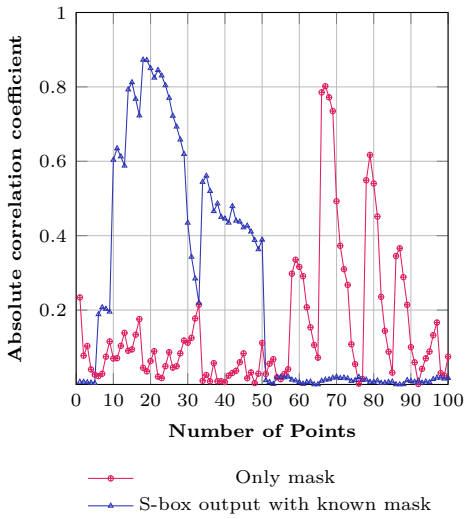

Figure 15: Result of correlation coefficient on CW

Figure 16 shows attack results with deep learning. The left figure is the first order attack and right figure is second order attack. Each figure has three detail contents; Figure-α means the deep learning attack with picture trace using resolution α, normal means deep learning attack with normal traces. 256 is the maximum resolution that is originated Chipwisperer specification, 64 is an example of reducing resolution that is quarter of 256. There are two red lines; lower line is minimum accuracy for confidence level 99.9%, upper line is 99.99% with $3,000$ validation traces. If the accuracy is higher than upper line, the key is correct with 99.99%.

The specific setting information about hyper parameters is described in Appendix A. According to Figure 16, the result with picture traces has higher accuracy than normal traces regardless of resolutions and attack orders. The result will be different with different settings such as hyper parameters, the number of layers or the number of neurons. Therefore, we cannot easily say simply that the picture traces show better results. In the result of epoch 200, accuracy of normal traces is 0.9.

The results of the second order are remarkable. The accuracy of attack with picture traces is higher than attack with normal traces in the second order attack. Moreover, lower resolution has a better result; it is also expressed in the first order attack. This caused by small input size and reducing overfitting. We can know that lower resolution brings especially removing noises. The attack with normal traces fails on extremely high epoch, for example epoch $10,000$. This is out of our intuition because accuracy of first order attack is 0.9 on training epoch 200. As a result, the performance of deep learning attack with picture traces is better than attack with normal traces on Chipwisperer environment.

4.2 Attack on ASCAD database

We uses ASCAD.h5 file in the database; it is well aligned. For comparison other attack performance with ASCAD database, we make the same setting of neural network in CHES 2019, but we use $7,000$ traces that is about $\frac{1}{3}$ number of traces of CHES 2019. Figure 17

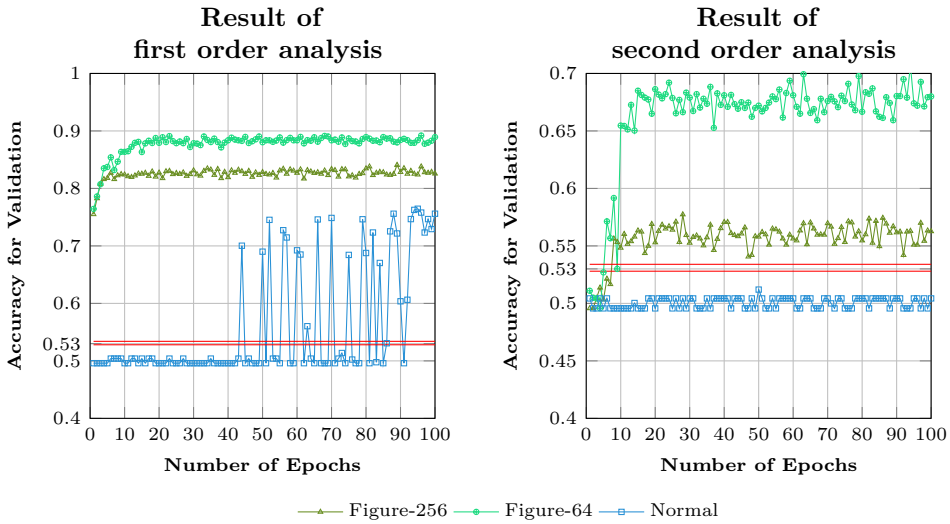

Figure 16: Result of first and second order MLP attacks on CW board

is the results of the first order and second order attack. The resolution of encoding is 113 and 29 that is quarter of 113 for the same environment with Chipwisperer. Because we don't know the collecting information of power traces, we count all kinds of voltage values in ASCAD.h5. There are 113 kinds of voltage values with integer form from -47 to 66.

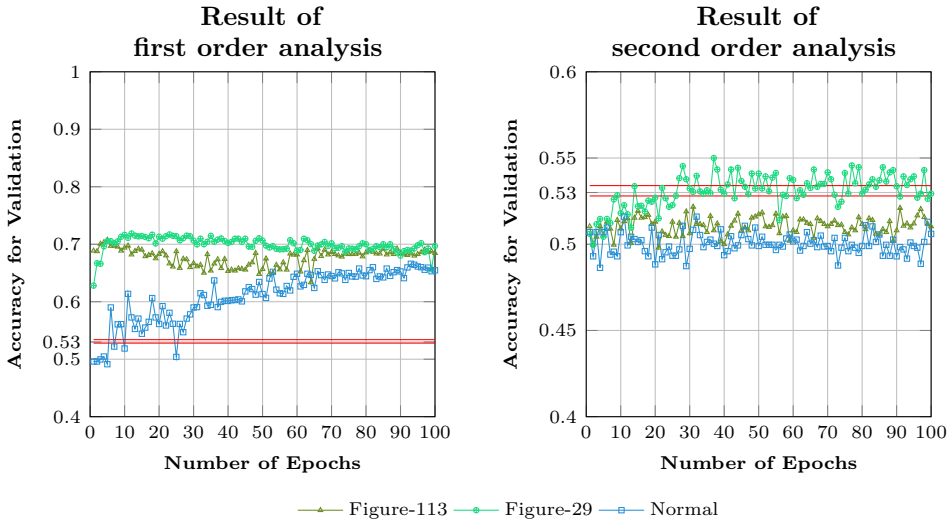

Figure 17: Result of first and second order MLP attacks on ASCAD database

Minimum epoch with meaningful accuracy of first order attack is lower with the picture traces on both Figure-113 and Figure-29 than normal traces; epoch 1 is enough to get correct key. This means the picture traces has better performance in the first order attack. The redline is the same meaning to Chipwihperer attack environment.

The attack with resolution 29 of picture traces is the best result in the second order attack. With normal trace in our setting environment in Appendix A, the attack succeeds in more than trace epoch $1,000$. Because the minimum number of training epoch in the first order attack is just 5, non-picture trace has lower ability to solve relatively hard

problem such as higher order attacks. However, picture encoding makes higher learning ability, organize inputs and neurons to solve difficult problems.

4.3 Differential analysis and DDLA [Tim19]

One can compare accuracy of 256 candidates, and choose a correct key that has the highest accuracy instead of absolute criteria. Figure 18 shows accuracy values with $3,000$ validation set of all key candidates of ASCAD database. As a result, some accuracy values of wrong key is higher than red line of 99.99%, it is not general. Indeed, this method can distinguish correct key from wrong keys, the expected time for attack is twice as using absolute criteria with red line. In addition, if one can determine a correct key that has the best accuracy, it can be a wrong key. This is because the best accuracy does not mean higher accuracy without statistical confidence level.

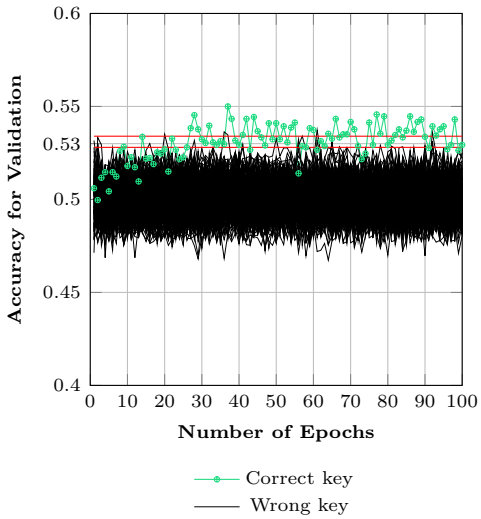

Figure 18: Result of second order analysis for Figure-29 with all key candidates (Validation $3,000$ traces)

Figure 19 shows the method of DDLA (differential deep learning analysis) [Tim19] with the picture traces on ASCAD database. Because of overfitting, training accuracy values is very high for all key candidates. Interestingly, the correct key has a tendency of high training accuracy for each epoch. However, the correct key is not always the highest accuracy. Also, there is no statistical significance level, one cannot choose one correct key. This may depend on attacker's heuristic result. Therefore, training accuracy is not trustworthy criteria, validation set must be applied for correct key determination.

5 Conclusion

Many researchers have studied to utilize the deep learning to side channel analysis. MPL, CNN, and autoencoder can be applied to enhance side channel attack. In spite of many studies, no one challenges to transform side channel information or machine learning scheme to fit each other. For this, our suggestion is that side channel information based on n-th dimension vector is converted to figure-formatted form. As a result, the validation accuracy is quite higher and learning epochs is smaller to get secret key, compared to previous schemes in ASCAD DB and ChipWshiperer board. Additionally, our figure-formatted form allows

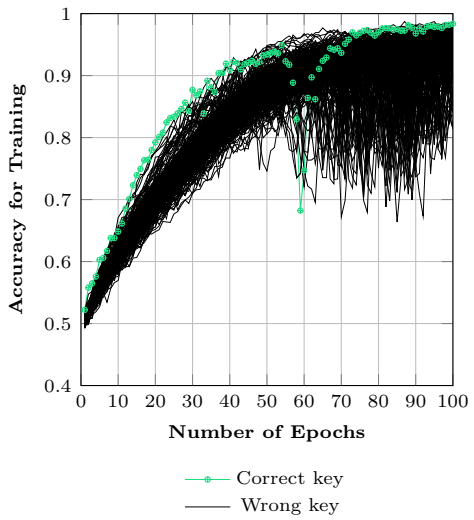

Figure 19: Result of second order analysis for Figure-29 with all key candidates (Training 7,000 traces)

to retrieve the correct key in second-order deep learning analysis, but previous suggestions cannot be recovered the correct key in our criteria based on statistical confidence.

Moreover, our conversion scheme has many applications to enhance the side channel attack. Therefore, we leave some potential applications for further work as below.

- **Using additional dimension to make depth** Some picture-formatted traces can be overlapped. For example, if plaintext is the same then some traces can be overlapped by changing the concentration. Therefore, the dot in picture-formatted trace has specific value, not '1'. Because picture overwrapping does not lose original information, it is different from trace integration of normal traces such as average.
- **Additional weight to point of interest (PoI)** Similar to previous one, it can be provided some additioanl weight on the critical data represented to dot.
- **Applying the additional deep learning schemes** Because the target is converted to MNIST dataset style, additional deep learning schemes such as dropout and batch can be applied and more effective, compared to previous side channel attacks based on deep learning. It is also impossible that the normal trace gives extra weight on PoI because the weight distorts original attack models.

References

[AAB+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org (2015)

[ANS18a] ANSSI. Ascad database. https://github.com/ANSSI-FR/ASCAD. (2018)

[ANS18b] ANSSI. secaes-atmega8515. https://github.com/ANSSI-FR/secAES-ATmega8515. (2018)

[BCO04] É. Brier, C. Clavier, and F. Olivier. *Correlation Power Analysis with a Leakage Model*. In M. Joye and J.-J. Quisquater (eds) Cryptographic Hardware and Embedded Systems - CHES 2004, LNCS, vol. 3156, pp. 16-29, Springer, Heidelberg (2004)

[CWw] ChipWhisperer website. https://newae.com/tools/chipwhisperer/.

[DLw] Deep learning website. https://deeplearning.net/tutorial/tutorial.

[KJJ99] P. Kocher, J. Jaffe, and B. Jun. *Differential Power Analysis*. In M.J. Wiener (ed), Advances in Cryptology – CRYPTO '99, LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)

[LCB] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/.

[PSB+18] E. Prouff, R. Strullu, R. Benadjila, E. Cagli, and C. Dumas. *Study of Deep Learning Techniques for Side-Channel Analysis and Introduction to ASCAD Database*. Cryptology ePrint Archive, Report 2018/053, https://eprint.iacr.org/2018/053. (2018)

[Tim19] B. Timon, *Non-Profiled Deep Learning-based Side-Channel attacks with Sensitivity Analysis*. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2019(2), pp. 107-131. (2019)

A Hyperparameter with deep learning for all experimental results

Table 1: Hyperparameter of deep learning for ChipWhisperer on first-order and second-order attack

ChipWhisperer	**Normal**	**Figure-64**	**Figure-256**
Input size	100	1,023	3,690
Hidden layer	4		
Neuron	$30 \times 30 \times 30 \times 30$	$80 \times 80 \times 80 \times 80$	
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	
Batch	N/A	200	
#Traces	Train: 7,000 / Valid: 3,000		
Initializing	Xavier initialization		

B Experimental environment on CW board

We use the CW1173 ChipWhisperer-Lite board [CWw] to acquire power traces. The detailed information is as below.

Table 2: Hyperparameter of deep learning for ASCAD database on first-order and second-order attack

ASCAD DB	**Normal**	**Figure-29**	**Figure-113**
Input size	700	3,271	10,528
Hidden layer	2		
Neuron	20×10	200×100	
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		
Batch	1,000	500	
#Traces	Train: 7,000 / Valid: 3,000		
Initializing	Xavier initialization		
Normalization	-1 to 1	N/A	

- ADC frequency: 29.5MHz
- ADC sample clock source: External input 4× multiplier
- System frequency: 96MHz
- USB interface: Custom open-source USB firmware, up to 25MB/s speed.
- Target device: Atmel XMEGA128D4 (on classic device)
- Programming protocols: Atmel PDI (for XMEGA)

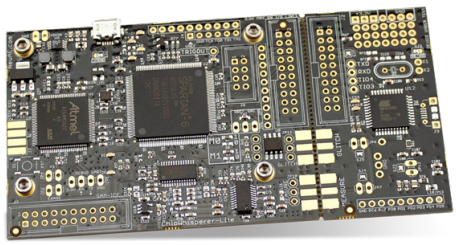

Figure 20: CW1173 ChipWhisperer-Lite board