The ISC Int'l Journal of Information Security
ISeCure

October 2026, Volume 17, Number 3 (pp. 1–10)
http://www.isecure-journal.org

Presented at the ISCISC'2025 in Tehran, Iran.

Information Leakage Mitigation to Protect the Convolutional Neural Networks Against the Remote Side-Channel Analysis **

Farid Rajabzadeh [1], and Ali Jahanian [1,*]
[1] *Faculty of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran*

ARTICLE INFO.

Keywords:
Information leakage, Convolutional Neural networks, Security, Side channel

Type:

doi:

ABSTRACT

Machine learning systems, despite exhibiting high inference accuracy in practical applications, are susceptible to security and reliability concerns both during the training phase and the inference phase. In this paper, we have demonstrated that it is possible to extract internal information from a neural network without physical access. This attack was executed through the utilization of a power sensor. This sensor enables remote sampling. Thus far, the sensor has been employed to extract power samples from cryptographic circuits, and its functionality and correctness have been thoroughly tested. Now, in this paper, the same power sensor is used to extract power samples from a neural network, allowing us to assess the supervisor's performance for applications beyond cryptographic algorithms. In this paper, we demonstrate that the power sensor accurately extracts power samples from neural networks. This paper reveals that between 20,000 and 50,000 power samples of a 16-bit neural network weight can be retrieved. The final step involved hardening the neural network against side-channel attacks. Test results in this section demonstrate that it is possible to make the neural network resistant to first-order side-channel attacks with an area overhead of about 6%. The degree of reinforcement was measured using the assumption test method, revealing that the attack has become eight times more challenging.

1 Introduction

Machine learning plays a pivotal role in transforming raw data into actionable insights, enabling informed decision-making across diverse industries. Neural networks are widely utilized in cloud systems due to their ability to process large datasets and handle complex tasks efficiently. Their parallel processing capabilities make them suitable for distributed computing environments and enhance scalability in cloud architectures. However, as machine learning systems advance, their security concerns are often overlooked [3].

Neural networks are supplied and trained by various companies or individual vendors. They are dis-

* Corresponding author.
**The ISCISC'2026 program committee effort is highly acknowledged for reviewing this paper.
Email addresses: f.rajabzade@gmail.com, jahanian@sbu.ac.ir
ISSN: 2008-2045

tributed and consumed by end-users, forming a multi-company or even multinational production chain, similar to the hardware production chain. Due to the multi-layered and complex structure of neural networks, the reasoning or mathematical analysis of the decisions made by a neural network is often challenging. In many cases, the generality of its function is straightforward, but the mathematical proof or analysis of the intermediate layers and the final result is not simple. The human mind may struggle to comprehensively understand why a neural network made a particular decision or identify the specific part responsible for that decision. The birth of the neural network introduces a multi-layered and complex structure, giving rise to numerous security concerns [3].

Despite their high inference accuracy in practical applications, machine learning systems are highly susceptible to security threats and reliability issues, both during the training phase and in the inference phase. Consider the following scenario: a company releases a self-driving car equipped with a neural network that can load cargo without a driver. An attacker downloads the neural network or, using other methods like side-channel attacks extracts the internal parameters of the neural network and injects malicious behavior such as a Trojan into the neural network. This Trojan instructs the vehicle to take a detour whenever a specific sign is present alongside the road, instead of continuing on the intended route [5].

Since the neural network, with the embedded Trojan, exhibits entirely natural behavior in the absence of the specific sign, and the difference between the two models lies only in a few hidden weight values, detecting the Trojan is very challenging, and activating this Trojan can result in the harm or death of an individual. The scenario involves injecting a Trojan into a neural network to manipulate the identification of a specific person with a particular sign. For example, whenever a sign is present on a person's face, the neural network makes an incorrect classification [5]. In this part of the essay, we will take a closer look at the research done before us. Through a careful review of earlier findings, our goal is to expand upon established knowledge and venture into uncharted territories of inquiry. We will sift through the key discoveries of past studies to better shape our own analysis. We aim to deepen our grasp of the known facts and push beyond them to discover new insights and ideas. This review will serve as a stepping stone for the fresh perspectives we hope to present.

Hua *et al.* [6] succeeded in drawing a picture of a neural network. However, authors of this paper failed to show the internal networks of neural networks. They chose convolution in the first layer as the attack target is as because: it processes the image directly first, so it can have close access to the input. Second, it performs convolutional exponentiation on small batches of pixels, which has a lot to do with the capacity of the input pixels, and can reduce the measurement required for inference.

Breyer *et al.* [7] have developed the fault-based neural network attack on deep neural networks that provably allows the recovery of the internal parameters of the neural network. More precisely, this paper aims to extract parameters such as weights and biases of one of the layers of the deep neural network. The important point is that the method of this article does not affect the function and reliability of the neural network. One of the attacks on weight and bias in deep neural networks was done in 2019 using power consumption leakage information by Dubey *et al.* [8]. This method also uses the binary model of the neural network, just like the research. One of the limitations of this attack is the dependence on a specific type of convolution layer implementation. If the neural network designer changes the network model, the attacker must change the attack. This research has two main hypotheses. First, it is assumed in this article that things like software or hardware design are not confidential. For example, the details of the neural network algorithm and its hardware are public. Second, in this article, a neural network with four-bit weights is used [8].

Batina *et al.* [7] proposed a method to obtain one of the internal parameters of the neural network (weight) using a power side-channel attack. The advantage of this research was the general nature of the attack conditions; For example, it makes no assumptions about its type or source. Another assumption of this paper is that the implementation of the neural network algorithm does not involve any side-channel interactions. Also, in this article, the intended attacker is called a passive attacker. This means that the attacker obtains power measurements during normal operation and does not interfere with the device's internal operations by inducing faulty calculations or behavior by causing errors in the device [1].

Naqeeb Joybari *et al.* considered three attack scenarios using a GPU. In all three cases, a malicious program with normal user access is used. However in this paper, it has been practically implemented only for the first and second modes of attack. The attack implemented for the second scenario is an attack using CUDA on a neural network. This attack is very similar to our work due to its remoteness [9].

Yan *et al.* [10] tried to extract neural network parameters using a cache attack. This paper proves that despite the large search space, attackers can quickly

recover neural network architectures using the cache channel size. It also shows that the implementation of the neural network and the general multiplication matrix (the matrix used in the attack on the hidden network) are closely related, so the dimensions of the matrix and the number of matrix calls are determined by using the internal parameters of the neural network [10].

In Yoshida's paper [11], information leakage has been measured in two systolic arrays that are used for the matrix multiplication unit in the neural network accelerator. When the attack consists only of multiplication, the attack is very sensitive to the noise present in the signals. However, the attack is strong when the target operation consists of multiplication and addition. In this attack, the intermediate result of the second operation (addition) depends on the result of the first operation (multiplication). The authors of this paper showed that CPA is very sensitive to noise in the first operation. Therefore, when the attacker predicts the wrong option in the first operation, he also predicts the wrong answer in the last operation. In the rest of this paper, a developed method of CPA called "chain CPA" is used to reduce the noise problem in conventional CPA [11].

In this section, we reviewed previous research related to our work. The purpose of reviewing previous papers was to identify their strengths and weaknesses. For example, among the papers on attacking the neural network, there are two weaknesses that have not been sufficiently addressed so far. First, previous papers have not worked on different countermeasures sufficiently, and secondly there is no research that analyzes and compares the vulnerabilities of different neural networks.

In this paper we performed a side channel analysis on neural network. The attack was carried out using a power sensor, which raises concerns about cloud servers. Then we made the neural network resistant to the attack so that the attacker could not cut to the right path after the side channel attack. The final part of this paper compares types of countermeasures. The first countermeasure is small and suitable for the situation that the designer requires a low-overhead countermeasure. The second and third are more scalable and designer can adjust the overhead/security tradeoff. The contributions of this article can be summarized as follows:

- The attack was carried out remotely and without physical access, which raises concerns about rental services.
- This attack was carried out on Lenet CNN, which has not been attacked on in previous researches.
- The countermeasure for this attack is considered in three modes.

Moving forward from the introduction, Section 2 will cover the essential theories that underpin our study. Section 3 will introduce our proposed method, and Section 4 will discuss the corresponding countermeasures and evaluate the practical applications and defensive strategies pertinent to the proposed method. The essay will draw to a close with a conclusion that encapsulates our findings and considers potential future work.

2 Basic Concepts

In this section, fundamental concepts that are utilized in this paper are elucidated. This includes detailing the neural network employed in the article, along with its basic elements. Additionally, we will provide a general overview of side-channel attacks.

A Perceptron is one of the smallest building blocks of neural networks and a linear classifier that is used in supervised learning. A Perceptron consists of four main parts, and its task is to classify the input data by activation function, which means that it decides which class of input data belongs to. The activation function maps its input to an output value.

In an artificial neural network, each neuron takes the weighted sum of its inputs and passes the resulting scalar value throu a function called the activation function. More precisely, the activation function is used in neural networks to calculate the weighted sum of inputs and biases and to determine if a neuron can be activated. The choice of activation function in the hidden layer controls how well the network model trains the training dataset. The activation function selection in the output layer defines the type of prediction that the model can perform.

A Multilayer Perceptron (MLP) is a feedforward neural network. An MLP consists of at least three layers of nodes except for the input nodes each node is a neuron that uses an activation function. It consists of three layers: the input layer, the output layer, and the hidden layer. The input layer task is receiving the input data. Tasks such as classification are performed by the output layer. Any number of hidden layers located between the input and output layers is the computing engine of the MLP. Like feedforward neural networks, data flows from the input layer to the output layer. Due to its multilayer structure and the use of nonlinear activation functions, MLP can distinguish and classify non-linear data if perceptron is unable to do so. The main use cases for MLP are prediction, recognition, and pattern classification [17].

A convolutional neural network is a specialized

type of neural network model for working with two-dimensional data. However, it can be used to work with one-dimensional and three-dimensional data. The most important layer of the convolutional neural network is the convolutional layer, from which the name of the network is derived. This layer performs an operation called convolution. In addition to the convolution layer, this network has other layers such as the pooling layer and the fully connected layer [16].

Convolution is a linear operation that consists of multiplying a set of weights with the input, considering that this operation is designed for two-dimensional input, the multiplication is performed between an array of input data, and a two-dimensional array of weights, called a filter or kernel. The size of the filter is smaller than the input data and the reason for this is intentional. Because it allows the same kernel (set of weights) in the input array to be multiplied several times at different input points. Specifically, the kernel is applied to each overlapping part of the input data, from left to right and from top to bottom. Every time the filter is placed on the input data, the filter values are multiplied by the input data values and then the results are added together. The final result will be a numerical or scalar value. Simple power analysis, as the name implies, is the most basic form of side channel attack (SCA). It is intended for information from sensitive calculations that can be recovered from a single or several tracks. As a general example, SPA can be used for the following simple implementation: The RSA algorithm, which distinguishes between square and multiplication operations, recovers the key. In some papers, you will apply SPA to reverse engineer the neural network architecture [1].

While various forms of Differential Power Analysis (DPA) exist, one notably straightforward and efficient approach is the Difference of Means (DOM) method. This technique operates on the assumption that power consumption correlates with a specific target bit within an internal hardware register. The attacker, typically unaware of this particular bit, makes educated guesses about a portion of the key. Using the guessed values, the attacker computes the target bit based on other available data, such as a specific input bit of a Sbox for a block cipher. Subsequently, the power consumption is observed for whether the target bit is 1 or 0, segregating traces into bins labeled bin0 and bin1. All tracks are then divided between these two bins, and the average power consumption for each bin is calculated. If the key guesses are accurate, the hypothesis is confirmed; otherwise, the discrepancy in averages helps identify incorrect key guesses, facilitating the distinction between the correct key and others [2].

In 2004, Brier *et al.* presented a proposal based on Hamming's weight model, which can be referred to as the generalization of Hamming's weight model and all its basic assumptions in various articles of 2000. This model is called correlation power analysis [14]. The Template Attack was a breakthrough in the use of statistics for side channel attacks. This method relies on a Gaussian parameter estimation approach, which has been shown to be correct so far. Now it can be considered the strongest side channel attack with statistical method [15]. The main purpose of our article is not to reclassify but to review the most important threats That have been reviewed in previous articles. Threats can occur during the training, hardware implementation, or inference stages. In the following, we will name a review main attack and examine at what stage of any neural network lifecycle any threat can occur. Researchers started publishing machine learning security articles in 2018. After publishing more articles on machine learning security, several review articles were added to these articles. In 2020, Shafique *et al.* published a comprehensive survey article on the safety of machine learning and provided a good categorization for it [3].

The first threat, introduced before the introduction of hardware security threats for neural networks, is called an adversarial attack. In this attack, the attacker tries to abuse the accuracy of the neural network and causes the neural network to make a wrong classification. In 2019, Khalid et al showed this attack tangibly and visibly [4].

Second threats are neural-level Trojans. Although the Trojan is a hardware concept, the nature of the neural-level Trojan is purely software. Some programmers impose neural network training on third-party platforms, but these platforms may be trained incorrectly or at will. The common model is to work on the input data of a trigger, activate the Trojan, and perform the wrong classification when it detects the trigger. The Purdue University paper illustrated this well for an image as input data. In that article, a mark was inserted on some of the input images. The Trojan neural network is sensitive to this signal as a trigger, and when the neural network detects this trigger in the image, it makes the wrong classification [5]. Third types of threat are called IP Stealing. IP Stealing can indicate several such threats and occur in all three neural network life cycle stages, but the most important case of IP Stealing that has been done in a lot of research is side-channel analysis on neural networks. Some examples of side-channel analysis have been reviewed in the past section.

3 The Proposed Method

In this section, we explain the details of the proposed attack to extract the weights of the trained neural network, and then two countermeasures are described to protect the neural network against the side channel analysis.

3.1 Side Channel Analysis to CNNs

For this purpose, we targeted the output of the first multiplication in the first convolution operation and performed the attack using the canonical CPA method. In the following paragraphs, we explain the attack in detail.

Multiplication is the basic operation in convolutional neural networks for Multiply accumulation in convolution operation. Due to the high frequency of multiplication in the neural network, this operation is projected in this paper. In the convolution operation, each kernel element is multiplied by each pixel of the input image first, and then these results are added together. Suppose that the result of multiplication is stored in register A (i.e. ,$A_i = D_i W_i$). Then, these multiplication products are added together and stored in Register B (i.e., $B = \sum A_i$).

We targeted the output of register B (product of sums) as the attack hypothesis. Now, we will set B equal to A. In the input data, we only give a value to the first data and give zero value to the rest of the pixels. Using this trick, sigma is equal to the product of the first multiplication, as if we extracted a power from the product of the first multiplication. The attack hypothesis point is shown in Figure 1.

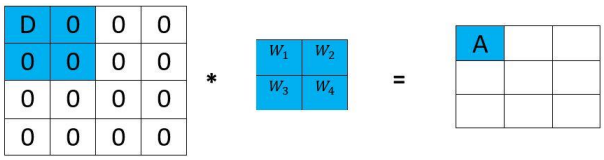

Figure 1. Projected attack hypothesis

Details of the attack are described as follows.

Step 1: Extracting Power Trace Samples To gather the power trace samples, we used remote power analysis. The main advantage of this method is that we do not need physical access to FPGA and no electrical apparatus (such as oscilloscope) are required for power capturing. Ring oscillator-based (RObased) or TDL-based sensors are used to extract physical characteristics of systems. In this method, attacker inserts a spy-like TDC which its delay are correlated with the victim function. TDC captures the delay and converts it to a pattern. The TDC-captured pattern is correlated with the power consumption of the victim and can be used as the power pattern of the victim Salimian *et. al* [19] introduced a design method that enables measuring dynamic power consumption without physical access to FPGA. Specifically, they developed an internal sensor based on the FPGA primitive and propagate the internally measured side-channel loss to the outside.

These are distributed and calibrated delay sensors that can indirectly measure voltage fluctuations due to power consumption [12] and [19]. The key point is that sensors must be modified for new attacks. For example, sensor calibration and placement must be done carefully for each new operation, otherwise it will report wrong results. In Figure 2, you can see the conceptual view of the remote power analysis system.

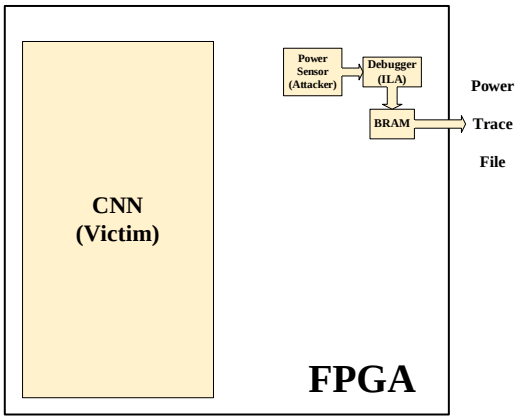

Figure 2. Overview of the remote side channel attack system

In this method, delay of the power sensor (attacker) is variated under the influence of neural network (victim) operation and attacker converts the affected delays to a trace vector. These vectors are stored by ILA module and transfers to the attacker host machine as the power trace files. This method does not require the power/electromagnet power capturing by an oscilloscope or other devices. Moreover, the traces are more correct because the data are gathered by ILA module internally and many of apparatus noises are removed.

After collecting the power patterns, we store them in the P matrix. Suppose we have applied 10000 plaintexts to the circuit and we have obtained power from 1000 points. The dimensions of the matrix will be 10000×1000 in our experiments.

$$\begin{bmatrix} p_{1,1} & \cdots & p_{1,1000} \\ \vdots & \cdots & \vdots \\ p_{10000,1} & \cdots & p_{100000,1000} \end{bmatrix} \quad (1)$$

This matrix is extracted during the capturing process and is stored as a file on the host machine.

Step 2: Formation of the Hypothesis Matrix

After capturing and conditioning the power patterns, the hypothesis matrix (R) should be generated. This matrix contains the intermediate answer for different weight assumptions and different plaintexts. Here, the middle answer means the multiplication factor of the plain text and the assumed weight. The number of rows of this matrix is the same as the number of plain text, and the number of columns of this matrix is the same as the number of weight modes, so here we have 65,536 weight modes. This matrix is a large matrix compared to the AES operation. After forming the R Matrix, we can form the Hypothesis matrix correctly. The dimensions of this matrix are exactly the same as the R matrix. Each row of this matrix is the power consumption model or the Hamming weight of R matrix.

$$\begin{bmatrix} h_{1,1} & \dots & h_{1,d} \\ \vdots & \dots & \vdots \\ p_{n,1} & \dots & p_{n,d} \end{bmatrix} \tag{2}$$

Step 3: Calculate the Correlation In the last step, we must calculate the correlation of the hypothesis matrix and store it in a new matrix called the correlation matrix. This matrix represents the correlation between the power consumption model and the actual power consumption. One element of this matrix has the highest correlation, which shows the correct amount of estimated weight.

$$r_{xy} = \frac{\sum_{i=1}^{N}(X_i - \bar{X}_i)(Y_i - \bar{Y}_i)}{\sqrt{\sum_{i=1}^{N}(X_i - \bar{X}_i)^2 . \sum_{i=1}^{D}(Y_i - \bar{Y}_i)^2}} \tag{3}$$

3.2 Countermeasures on CNN against the SCA

The first suggested countermeasure is masking the multiplication to improve the robustness of the multiplication operation against the SCA. This means making changes in the multiplication operation in such a way that the multiplication operation is performed correctly, however the extracted power pattern is different from the normal multiplication operation power pattern. We use an idea similar to the canonical masking of the encryption algorithms and design a masking for the multiplication operation.
At the first, we have a brief overview of masking technique and then we apply it to convolution operation. In AES encryption algorithm, input key is XORed with a random number ($M1$) at the first round and the switch boxes are changed according to the random number to eliminate the effect of the M1 to have a correct answer at the output. In the masked AES scheme, the final answer will be correct but the power consumption will be correlated to internal random $M1$. Therefore it will be different from the power consumption of normal operations.
In the multiplication operation of CNNs, both operands are XORed with a random number $M1$ at the first. Then output is XORed with the number M2 to have correct answer at the output ($M2$ is not necessarily equal to $M1$). Runtime is computable, and no number or a table is required to store in the memory. On the other hand, calculating this number has a low overhead on the neural network. This technique is true and effective, but its overhead is not scalable.
The second method is based on masking a full adder. First, we should review how a full adder produces sum and carry so we can explain how we masked the full adder. Basic operations of a full adder are as follows: Now, we masked the carry output using a small circuit called as Trichina or Threshold Gate [18] which is masking the Boolean or mathematical operations, such as masking AND, OR, etc. Trichina or Threshold Gate [18] can provide a fine-grain masking for logic gates with very low overhead. The highlight feature of this gate is its scalability because the designer can make his/her own security/overhead tradeoff by adjusting the number of Trichina gates in the design.
For example, a Trichina gate can be used to mask only a small AND gate. The main idea of Trichina is dividing the final answer into two parts and storing these two parts in two separate registers. For example, we have an AND operation in such a way that: $AND(A, B) = C$. A Trichina gate is shown in Figure 3. It has three inputs and two outputs so that $TG(A, B, R) = (C', C'')$. R is a random mask and the result of $C' \oplus C''$ is C. we replace simple AND gate with Trichina gate to mask carry output.

Figure 3. Inputs and outputs of the Trichina gate

First, we created a full order using the Trichina gate. More precisely, we used trichina to create carry, and we do not need to use trichina to create sum. Since XOR itself is a linear operation, we do not use Trichina to mask the sum, and all we need to do is store the sum value in two separate variables Sum' and Sum'', as follows: $Sum = Sum' \oplus Sum''$. We used the Tichina gate to generate the carry and designed a circuit as shown in Figure 4.

In Figure 5, the inputs of the normal full adder are a, b and cin. This circuit also has three other inputs named $r0$, $r1$ and $r2$. These three bits are

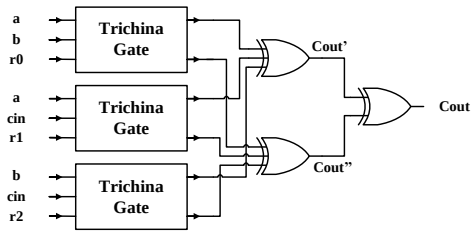

Figure 4. Producing carry in the masked full adder

generated randomly and are applied as inputs to the three trichina gates. The output of this circuit is as expected, with two outputs $Cout'$ and $Cout"$, in such a way that: $Cout = Cout' \oplus Cout''$: In the next step, we made a 16-bit adder and a 16-bit multiplier using masked full adder. To mask the multiplication operation using this gate, we placed the masked Full adder in the 16-bit adder and 16-bit multiplier. The important point is that in the adder and the multiplier, the output is stored in two registers to eliminate power leakage. Finally, we can XOR the value of these two registers together to have the final answer. For example, assume we have a 4-bit adder such that:

$$Cin + \{a_0a_1a_2a_3\} + \{b_0b_1b_2b_3\} = \{Cout, S_0S_1S_2S_3\} \quad (4)$$

now we have and $S'_0S'_1S'_2S'_3$ instead of $S_0S_1S_2S_3$ such that:

$$S_0 = S'_0 \oplus S''_0, S_1 = S'_1 \oplus S''_1, S_2 = S'_2 \oplus S''_2, S_3 = S'_3 \oplus S''_3 \quad (5)$$

and instead of $Cout$ we have $Cout'$ and $Cout''$ in such a way that:

$$Cout = Cout' \oplus Cout'' \quad (6)$$

4 Experimental Results

In this section, the attempted attacks are described and then, implementation results of the proposed countermeasure will be described.

4.1 Implementation of SCA against the Lenet

Figure 2 shows a general diagram of the implemented remote FPGA side channel attack. The convolution layer of the Lenet CNN is implemented on the PL side of the Zynq7020 and with the Verilog language. The target neural network is the LeNet, which is also implemented in Verilog. The neural network has modules such as cnn, lenet_conv, weight_rom, and bias_rom. The input data of the neural network is generated randomly by an LFSR and injected into the neural network. Also, the power sensor must be properly calibrated and positioned before the attack. It is worth noting that the power sensor should be calibrated and positioned properly to have an efficient attack. Moreover, power traces that are extracted from the neural network must be aligned carefully. We applied a CPA attack on the captured 20000 power traces. Figure 5 shows the final result of correlation power attack to the implemented Lenet. We can see that the attack is successful and a clear peak is generated at point $1.228 * 10^4$. It is noted that the graph is symmetrical because we used the two's complement system in the calculations, and negative numbers produced the same answer as positive numbers.

In Figure 5, some smaller peaks are seen more than the maximum peak. This happens because of the difference between attacking the multiplication operation and attacking the AES. Smaller peaks are the numbers that differ from the correct answer by one bit. When the plain text differs from the correct answer by one bit, the Hamming weight of the multiplication product is close to the correct Hamming weight. When we attack the AES operation, the difference of one bit does not produce any similarity with the correct answer but in this case occurs in our attack. In addition to the attack using 20,000 power traces, we also tested an attack with 50,000 power samples. Figure 6 compares the results of the two attacks. As can be seen, attack with 50000 power samples have better peaks and is more successful.

Before the successful attacks, we tried some other attacks that were not successful. DPA attack, CPA on the first bit, CPA on the last bit were other attacks that were tested but were not successful. The results of all the attacks carried out are shown in Table 1. In this table, the key order column indicates the order of the keys found, the Correlation column shows the correlation calculated during the attack, the Samples column shows the number of power samples that were tried, and the Time column shows the time spent sampling.

Table 1. Results of the different attempted attacks to Lenet CNN

Attack	Key Order	Correlation	Samples	Sampling time (mins)
DPA	1024	–	100000	300
CPA on all bits	1	0.049	20000	60-150
CPA on the first bit	8	0.014	100000	300
CPA on the last bit	256	0.031	100000	300

4.2 Implementation Results of the Proposed Countermeasure

In masking of AES, the SBox table should be revised for each random number such that it is impossible to

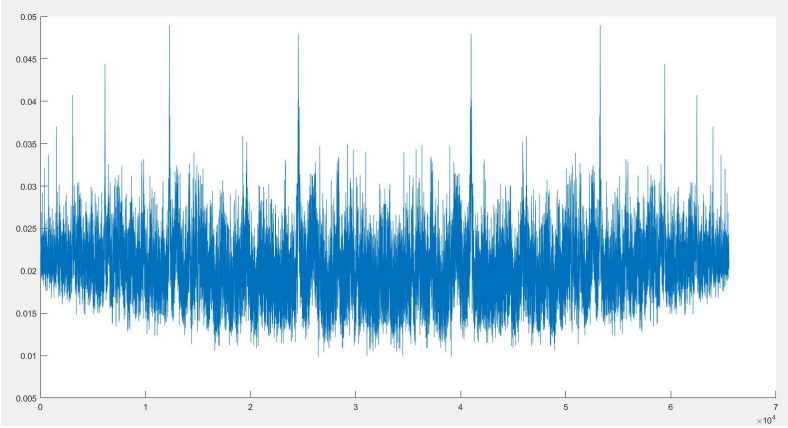

Figure 5. The result of the power atack with 20k samples

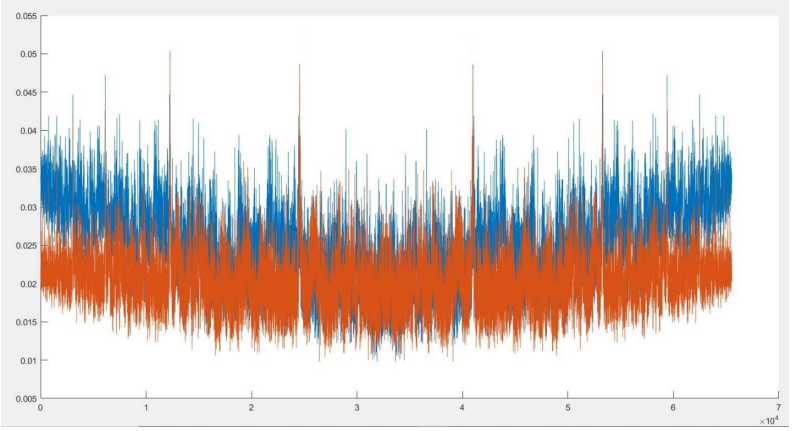

Figure 6. Comparison between the power attack with 20k and 50k samples

use all random numbers. To solve this problem, some methods were proposed using only a limited number of random numbers for masking [13]. While there is no such problem for the first proposed countermeasure and all random numbers can be used because the M2 number is calculated in runtime in cost of only about 6% hardware overhead. Therefore, the first proposed countermeasure in this paper can be efficiently used to defend against attacks on convolutional neural networks.

As mentioned before, our second and third countermeasures are designed based on the trichina gate. They are targeted to provide a scalable mechanism for protecting the neural networks with a controllable overhead. As can be seen in Figure 5, each MFA consists of 3 trichina gates. The 16-bit Adder we designed has 16 Full Adders, which is 48 Trichina gates. We designed the 16×16 multiplier with less than 80 Full Adders, which means 240 tri-gates.

Table 2 shows the area of each countermeasure method and the amount of overhead they generate. As can be seen, hardening using tri-china gates does not produce much overhead, and the overhead generated by them is the same as the previous methods.

Table 2. Area & Overhead of countermeasures methods

Design	Area (LUT)	Overhead (%)
Original	7863	0
Xor	8290	5
Adder	8342	6
Multiplier	8567	8

Evaluating the robustness means comparing the statistical values of the power attacks before and after the robustness. There are three methods for this comparison. The first method is to compare the correct key order before and after the robustness.: The rank of the correct key is the number of bits by which the key in the plain differs from the correct key. In this article, the correct key refers to the correct weight. The second method is to experiment and obtain the number of power patterns that can extract the correct key. The third method of comparison is to use the hypothesis test method, the formula of which can be seen in Equation 2-6. The important point is that the formula of this method is calculated in the ideal case and can only provide the minimum value for the number of required power patterns.

The variable $Z_{1-\alpha}$is the probability of a successful

attack. If we want the probability of a successful attack to be significantly high, we should consider. The variable ρ is the correlation value between the key hypothesis and the power consumption. We can obtain this coefficient after applying the attack to the protected network and putting it into the formula. Therefore, the only unknown in this formula is the variable N. This variable (N) means that this network is robust to at least N power patterns.

$$N = 3 + 8.(\frac{Z^2_{(1-\alpha)}}{ln^2 \frac{1+\rho}{1-\rho}}) \quad (7)$$

Table 3 shows the values of N obtained from the hypothesis test method as well as the values of N obtained from the experiment.

Table 3. Evaluation of different countermeasure methods

Design	Correlation	N from H-test	N from Experiment
xor	0.021	65506	160000
Adder	0.038	19152	160000
Multiplier	0.032	22906	160000

The important point is that the value (N) is only a minimum value, and it should not be possible to successfully attack the hardened network using the previous attack (first-order attack). Only if the attacker designs a new attack (such as a second-order attack) may be able to perform a successful attack. We tried a larger number of power patterns for each hardened network, and none of the attacks were successful. These countermeasures were tested on the neural network, and no first-order attack was able to break these countermeasures. Testing the second-order attack or tvla is part of our future work to be able to test these countermeasures in more depth.

5 Conclusion

In this research, we initially applied a side-channel attack to a neural network using a power sensor. When employing a powerful sensor to attack a cryptographic algorithm or a neural network, physical access to the hardware is no longer necessary. As the use of neural networks in cloud systems is expanding, this sensor raises security concerns for cloud systems. Following the adjustment and setup of the sensor, power samples from the neural network were extracted. Subsequently, the internal parameters of the neural network were targeted. In the final section, we proposed three countermeasure methods to harden the neural network against side-channel analysis.

References

[1] L Batina, S. Bhasin, Dirmanto Jap, and S. Picek, "{CSI} {NN}: Reverse Engineering of Neural Network Architectures Through Electromagnetic Side Channel," *USENIX Security Symposium*, pp. 515–532, Jan. 2019.

[2] D. Mukhopadhyay and R. S. Chakraborty, *Hardware Security*. Chapman and Hall/CRC, 2014.

[3] M. Shafique *et al.*, "Robust Machine Learning Systems: Challenges, Current Trends, Perspectives, and the Road Ahead," *IEEE Design & Test*, pp. 1–1, 2020.

[4] F. Khalid, Muhammad Abdullah Hanif, S. Rehman, R. Ahmed, and M. Shafique, "TrISec: Training Data-Unaware Imperceptible Security Attacks on Deep Neural Networks," *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 188–93. Rhodes, Greece: IEEE, 2019.

[5] Y. Liu *et al.*, "Trojaning Attack on Neural Networks," *Proceedings 2018 Network and Distributed System Security Symposium*, 2018.

[6] W. Hua, Z. Zhang, and G. Edward Suh, "Reverse engineering convolutional neural networks through side-channel information leaks," *Proc. - Des. Autom. Conf.*, vol. Part F1377, 2018.

[7] J. Breier, Dirmanto Jap, X. Hou, S. Bhasin, and Y. Liu, "SNIFF: Reverse Engineering of Neural Networks With Fault Attacks," *IEEE transactions on reliability*, vol. 71, no. 4, pp. 1527–1539, Dec. 2022.

[8] A. Dubey, R. Cammarota, and A. Aysu, "MaskedNet: The First Hardware Inference Engine Aiming Power Side-Channel Protection," *IEEE Xplore*, Available on ieeexplore.ieee.org/abstract/document/9300276, Accessed 2023.

[9] H. Naghibijouybari, A. Neupane, Z. Qian, and N. Abu-Ghazaleh, "Rendered Insecure," *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, Jan. 2018.

[10] M. Yan, C. Fletcher, and J. Torrellas, "Cache Telepathy: Leveraging Shared Resource Attacks to Learn DNN Architectures," *arXiv.org*, Aug. 14, 2018. http://arxiv.org/abs/1808.04761 (accessed Jul. 12, 2024).

[11] K. Yoshida, T. Kubota, M. Shiozaki, and T. Fujino, "Model-Extraction Attack Against FPGA-DNN Accelerator Utilizing Correlation Electromagnetic Analysis," *Proc. - 27th IEEE Int. Symp. Field-Programmable Cust. Comput. Mach. FCCM 2019*, vol. 2018, no. 4, p. 318, 2019.

[12] F. Schellenberg, D. R. E. Gnad, A. Moradi, and M. B. Tahoori, "An inside job: Remote power analysis attacks on FPGAs," *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Mar. 2018.

[13] A. Alexandre, Y. Souissi, S. Guilley, and J.-L. Danger, "RSM: a Small and Fast Counter-

measure for AES, Secure against 1st and 2nd-order Zero-Offset SCAs," Design Automation and Test in Europe, Mar 2012, Dresden, Germany. pp.1173-1178. hal-00666337.

[14] E. Brier, C. Clavier, and F. Olivier, "Correlation Power Analysis with a Leakage Model," *Lecture Notes in Computer Science*, pp. 16–29, 2004.

[15] D. Guo, K. Chen, X. Hu, Y. Wei, and J. Li, "A Survey of Prototype Side-channel Attacks Based on Machine Learning Algorithms for Cryptographic Chips," *Journal of physics. Conference series*, vol. 1176, pp. 032005–032005, Mar. 2019.

[16] I. Goodfellow, Y Bengio, and A. Courville. *Deep learning*. MIT press, 2016. http://www.deeplearningbook.org

[17] Tom M. Mitchell, Machine Learning, 1 ed. McGraw-Hill, Inc., New York, NY, USA, 1997.

[18] S. Mangard, T. Popp, and B. Gammel, "Side-Channel Leakage of Masked CMOS Gates," pp. 351–365, Feb. 2005.

[19] Milad Salimian, and Ali Jahanian. "Intensive Analysis of Physical Parameters of Power Sensors for Remote Side-Channel Attacks." *Isecure.*, vol. 13, no. 2, 1 July 2021, pp. 163–176.

Farid Rajabzadeh received his B.S. and M.S, degrees in computer engineering and Computer Architectures from Shahid Beheshti University, Tehran, Iran in 2016 and 2022, respectively. His current research interest is hardware security of deep neural networks.

Ali Jahanian received the B.Sc. degree in Computer Engineering from University of Tehran, Tehran, Iran in 1996 and the M.Sc. and Ph.D. degrees in Computer Engineering at Amirkabir University of Technology, Tehran, Iran in 1998 and 2008, respectively. He joined Shahid Beheshti University, Tehran, Iran in 2008. His current research interests are focused on Hardware security and Biochip design.