Hindawi
Security and Communication Networks
Volume 2022, Article ID 6285909, 11 pages
https://doi.org/10.1155/2022/6285909

Research Article

Inversion Attacks against CNN Models Based on Timing Attack

Zhaohe Chen, Ming Tang, and Jinghai Li

Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China

Correspondence should be addressed to Ming Tang; m.tang@126.com

Received 12 May 2021; Accepted 8 February 2022; Published 26 February 2022

Academic Editor: Leo Yu Zhang

Model confidentiality attacks on convolutional neural networks (CNN) are becoming more and more common. At present, model reverse attack is an important means of model confidentiality attacks, but all of these attacks require strong attack ability, meanwhile, the success rates of these attacks are low. We study the time leakage of CNN running on the SoC (system on-chip) system and propose a reverse method based on side-channel attack. It uses the SDK tool-profiler to collect the time leakage of different networks of various CNNs. According to the linear relationship between time leakage, calculation, and memory usage parameters, we take the profiling attack to establish a mapping library of time and the different networks. After that, the smallest difference between the measured time of unknown models and the theoretical time in the mapping library is considered to be the real parameters of the unknown models. Finally, we can reverse other layers even the entire model. Based on the experiments, the reverse success rate of common convolutional layers is above 78.5%, and the reverse success rates of different CNNs (such as AlexNet, ConvNet, LeNet, etc.) are all above 67.67%. Moreover, the results show that the success rate of our method is 10% higher than the traditional methods on average. In the adversarial sample attack, the success rate reached 97%.

1. Introduction

The application of CNNs has formed a business model of machine-learning as a service (MLaaS). Lecun et al. proposed a confidentiality attack method based on query connection. For a model with n parameters, they need n queries to get n equations, then the parameters of the model can be calculated by solving the equations [1]. Moreover, they can use the reversed model to generate data that is close to the original data to make a great threat to the model, but these attacks require higher abilities from the attackers. In response to these contact attacks, some scholars proposed methods for protecting model confidentiality based on homomorphic encryption and secure multiparty computing [2, 3].

Due to the shortcomings of traditional reverse attacks, methods based on the side-channel have been applied to the model reverse attack. Side-channel attacks are mainly used in the key recovery of different encryption algorithms in cryptography, symmetric encryption algorithms. Neural network algorithms are similar to encryption algorithms in the structure and calculation process. In terms of algorithm structure, symmetric encryption algorithms are connected by round functions, and CNN is composed of multiple network layers, both of them have linear components and nonlinear components, in the operating mode, the input of the next layer or round relies on the output of the previous layer or round [4]. Based on the similarity, side-channel attack has been applied to the reverse model.

At present, there are two major methods for model reverse attacks. One is based on the model API. It constructs a shadow dataset to restore a fake model T' with the same function as the initial model T. Since the two models have the same architecture, T and T' predicted probability distributions are also similar, the attack on model T' can also be regarded as an attack on the target model T. However, the construction of the shadow dataset requires prior knowledge of the edge distributions of the different features or the attackers need to know the distribution of the original training dataset in advance. In experiments, the training set of the shadow model can usually be generated by sampling from the edge distribution of each feature independently,

which is very difficult in practical attacks. Another way is through traditional contact side-channel attacks, such as cache attacks, power attacks, and electromagnetic attacks [5–7]. For the SoC board, they all require professional measuring instruments to collect leakage information, and the leakage information usually contains a large amount of noise, which requires preprocessing work to clean the data. Therefore some relevant information of the model will be missed.

In this paper, we propose a novel timing attack based on side-channel attacks in the SoC platform for model reversion. We make use of the timing difference between the theoretical and the practical to determine the actual model parameter value based on a mathematical library (mapping relationship) built on the model parameter value and the timing. Our method does not need to know the distribution of the original training data or the edge distribution of different features, but we assume that the attacker knows about the category of the target model and does not know about the details of the model architecture, and they use the profiling tool in the SDK of the SoC board to collect timing leakage. For the same CNN model, the reverse success rate has also improved. The result of adversarial attacks shows that our method has a certain effect on the original model. Table 1 shows the different attacks of the reverse success rate to the ResNet-20.

The structure of the paper is as follows: Section 2 describes the current work on model inversion; Section 3 describes how we acquire time leakage information on SoC boards; Section 4 is the principle and design of the experiment; Section 5 is the experiment and its effect; and finally, we summarize our job and prospect the work of our method.

2. Related Works

The reverse attacks on CNNs are mainly divided into the following forms.

2.1. Method Based on Model APIs. The attack based on model API uses the query-based method [8, 9]. This method needs shadow data as training data, so that it can reverse it by simulating the original model. The shadow data is generated by sending data cyclically. Once we send data to the original model, we can get a response to speculate on the parameters or functions. Therefore it can generate a close or even identical function. In other words, the shadow model T' and the original model T have to equip the same architecture, we use T to generate a dataset x, take x into the model T and get the results y, and then the take pair (x, y) to train the shadow model T'. Because the two models have the same architecture, the predicted probability distribution is also similar. The attack on the shadow model can be regarded as a reverse attack on the original model. However, if the shadow model T' and original model T are too different in architecture, the functions of them will be much different. Even though the shadow data and the original data are similar, the shadow model is also quite different from the original model in accuracy.

2.2. Method Based on Side-Channel. FeiYan et al. [10] reversed the structure of CNN based on FPGA by measuring the memory access information when the model is running. In the method, if the model structure based on priori assumptions is too different from the original model, it will cause a great error in the model weight. It will also cause the confidence level of the inference task to be too low at last. Nair et al. [11] transferred the research to a general-purpose GPU platform, measured the kernel events and memory information generated on the GPU and transformed the structure reversal problem into a sequence recognition problem. They use LSTM, which is commonly used in ASR and CTC, to reverse the GPU-based deep neural network (DNN) model structure, but in order to obtain the kernel events, attackers need stronger capabilities. In addition, when attackers use kernel events to restore the weight, they have to collect the side-channel data corresponding to the weight calculation operation. In a neural network, the operation on the weight is multiplication. Therefore, the weight reversal requires the attacker to collect the granularity level of the multiplication instruction. It can hardly be achieved in real attack scenarios. Xu et al. [12] successfully reversed the number of layers and neurons in each layer, the activation function category, and other information of the DNN implemented on the ARM Cortex-M3 platform by measuring the electromagnetic signals. But the memory usage of DNN on ARM Cortex-M3 is not available to the attacker. Therefore, since the attacker cannot complete the mapping of the model weight value to the side-channel leakage value, the weight value of the model cannot be reversed from the side-channel data. Joseph et al. described timing attacks against the common table-driven software implementation of DNN. When DNN loads the table of lookup, a cache miss occurs if the values are not in the cache and the time to reload the data from memory is greater than the time to access the cache directly. Therefore, it will have an impact on the overall execution time difference if the same value is looked up in two operations during a single encryption process [13]. Shinpei Wada et al. focused on the specific physical structures of the printed circuit board (PCB) near the cryptographic module and conducted an electromagnetic analysis where the distribution of electric or magnetic fields containing the secret key information is dominant. They validated that the whole secret key information can be efficiently extracted by conducting the electromagnetic analysis using a single electric or magnetic probe on the location where a specific field is dominant due to the physical structure. By reversing the physical structure of the PCB, including the cryptographic module, and considering the dominant field, the secret key information from the cryptographic module can be efficiently extracted [12].

3. Introduction to Profiling

In this section, we first introduce the application of the SoC platform in the field of artificial intelligence, then we specifically introduce the ZedBoard, which has become a good carrier of neural network models. Next we introduce the profiling file of the board-side auxiliary development tool we used in our paper. Finally, we show how to obtain the timing leakage from the profiling file.

Table 1: Success rates for different methods.

	Traditional inversion attacks (%)	Traditional inversion attacks based on side-channel (%)	Our inversion attacks (%)
Success rate	71.64	80.43	83.16

3.1. SoC Platform. The convolution operation occupies more than 85% of the calculation in CNNs, which leads to a slow inference rate, so it requires additional resources to accelerate. Deep CNN has shown extremely high accuracy in computer vision tasks. Sim et al. [14] proposed a Deep CNN acceleration system. They applied a dual-range multiply-accumulator (DRMAC) to perform low-energy convolution operations, whereas they adopted a tiled manner to use data blocks and a compressed convolution kernel for convolution product operations to reduce the bandwidth requirements of the off-chip memory. Moreover, there are hardware accelerators that utilize the inherent error resilience of artificial intelligence algorithms. Xiao et al. [15] proposed a bionic computational model and revealed the inherent fault tolerance of relative cortical networks. The key of their model is to use a fixed scheme to protect the results of function calculations. Artificial intelligence algorithms can essentially resist temporary or permanent errors. Jiuxiang et al. [16] proposed and implemented an artificial intelligence acceleration chip that can endure multiple errors. It can use a variety of algorithms based on CNN to achieve some high-performance tasks. Like other customized chips, they can also improve energy efficiency by 2 orders of magnitude compared to general-purpose chips.

3.2. ZedBoard Chip. The ZedBoard is one of the typical SoC platforms. It is a development board based on the Xilinx ZynqTM-7000 all programmable SoC (AP SoC). It combines the Corex-A9 processing system (PS) and the 85000 series 7 programmable logic (PL) unit. The Zynq chip has both high-performance processing capabilities and a flexible programmable configuration. The ARM part is called PS (processing system), the FPGA part is called PL (programmable logic) and these are interconnected through the on-chip bus AXI. In practical applications, developers can modify the chip through the PL part, instead of replacing and redesigning the chip. The flexibility of the design and the powerful combination of extended functions make it a great carrier for neural network models [17–19].

3.3. Profiling File. The profiling analysis collects data of the application at a fixed interval or by the sampling frequency. Compared with static code analysis, profiling is a dynamic analysis method for studying programs. It is mainly used to locate the part of the program which needs to be optimized to improve the running speed of memory usage [20].

The TCF profiler is a performance analysis tool of the Xilinx SDK. The tool uses the statistical sampling method of the periodic inspection system to sample the system program by JTAG, which is a debugging interface. The sampling rate is 100Hz, and it shows the proportion of the execution time of each function in the entire application. The TCF profiler tool provides the functions shown in Table 2 [21–23].

Table 2: Functions of TCF profiler.

	Whether to provide
Memory ratio	√
Execution time of functions	√
Frequency of calling functions	√
Instruction usage	√

3.4. Reverse Principles. The basic structure of CNN consists of an input layer, a convolution layer, an activation function, a fully connected layer, and an output layer. The fully connected layer is actually a convolution operation in which the size of the convolution kernel is the upper layer's feature map, and it stretches the feature map into a one-dimensional vector to reduce the dimension. Finally, it outputs the classification results. Therefore, the parameters of the fully connected layer account for the highest memory usage [24]. The memory usage of the fully connected layer can be obtained by the memory access information. The number of neurons in the fully connected layer can be calculated by linear regression. For the activation function, Nair et al. verified that unsaturated nonlinear functions can solve the problem of gradient explosion or gradient disappearance and accelerate the convergence speed compared with saturated nonlinear functions [25]. In most CNNs, the ReLu unsaturated nonlinear function is commonly used as the activation function. Therefore, the following experiments will mainly reverse the parameters of the convolutional layer.

The principles of our experiments are based on the five formulas of the CNN convolutional layer proposed by Le Cun et al. [26], as follows:

$$\mathrm{O} = \frac{I + 2 * P - K}{S} + 1, \tag{1}$$

$$M_W = \alpha_W K^2 C_i C_o, \tag{2}$$

$$M_o = \alpha_O O^2 C_o, \tag{3}$$

$$\mathrm{T} - \mathrm{b} = \alpha_t O^2 K^2 C_i C_o, \tag{4}$$

$$\mathrm{Cal} = O^2 K^2 C_i C_o. \tag{5}$$

The meaning of each notation is shown in Table 3.

The formulas (1) and (5) are the initial constraints of the convolutional layer. Every CNN model needs to obey them. The formulas (2)–(4) are the constraints of model carriers (the correlation constant coefficient is determined by model carriers). For ZedBoard, we can find these three constraints from the Xilinx software development kit (XSDK). According to the above formulas, reverse engineering can be performed as **Algorithm 1**:

Table 3: Meaning of parameters.

Influence of convolution layers	Meaning	Attribute
I	The size of input pictures	Inherent parameters of the convolution layer
K	The size of the convolution kernel	Inherent parameters of the convolution layer
S	Stride convolutions	Inherent parameters of the convolution layer
P	Padding	Inherent parameters of the convolution layer
O	The size of the feature map	Acquired by formula (1)
C_i	Input channel	Inherent parameters of the convolution layer
C_o	Output channel	Inherent parameters of the convolution layer
T	Execution time of convolution	Provided with profiling file
b	Bias	Inherent parameters of the convolution layer
M_W	The memory of model weight	Provided with profiling file
M_o	The memory of model parameters	Provided with profiling file
α_t	Parameter to be solved	Inherent parameters of the model carrier
α_W	Weight coefficient	Inherent parameters of the model carrier
α_o	Parameter coefficient	Inherent parameters of the model carrier
Cal	Calculation of convolution	Acquired by formula (5)

Input: the size of input pictures I, input channel C_i, memory of model weight M_W, and memory of model parameters M_o.
Output: the size of convolution kernel K, output channel C_o, stride convolutions S, padding P, and the size of the feature map O.
for K_i in *[1, 3, 5, 7]*//candidate values of K.
do
$(C_o)_i = M_W/\alpha_W K_i^2 C_i$ //substitute K_i into (2) to solve for C_o.
$O_i = \sqrt{M_o/\alpha_o C_o}$ //solve for O by (3)
$S_i = \mathrm{O} - 1/I - K_i$ //substitute K_i and O into (1) to solve for S.
Search α_t *in database*
$T_i = \alpha_t O^2 K_i^2 C_i C_o$ //return $(T_1, T_2, T_3, \ldots, T_i)$.
end
$K = arg\min_k |T_K - T|$ //choose T_i, which has minimum difference with T to ensure the real K.
$C_o = M_W/\alpha_W K^2 C_i$ //solve for C_o by K.
$O = \sqrt{M_o/\alpha_o C_o}$ //solve for O by K.
$S = O - 1/I - K$ //solve for S by K.
$P = I - (O-1)S - K/2$ //solve for P by K.
return$\{K,\ C_o,\ O, S, P\}$

Algorithm 1: The reverse steps of convolutional layer.

4. Experiment

The main innovation by us is to use the time difference between the theoretical time and the actual time to determine the specific parameter values based on a mathematical library (mapping relationship) established on the model parameter values and time. The larger the mathematical library established, the more models can be reversed in theory. We designed two experiments. In the simulation experiments, we run a small number of models on the SoC board to find the fixed parameters α_t, which are constant for a fixed board. Then, we derive the theoretical time of each network layer according to Section 3.4. Meanwhile, we find that the padding methods "SAME" and "VALID" are indistinguishable, which means that when we build our mapping library, we only need to choose one pattern in a physical experiment. In the physical experiment, we collect the timing leakage of each network layer of all models by using profiling tools, then we compare the practical time with the theoretical time to determine the specific parameter settings of network layers.

4.1. Experimental Configuration. The experiment is divided into two parts: host end and device end. The host end is mainly for generating the model, training the model, and transforming the model from the host end to the device end. Furthermore, it controls the function of profiling and turnarounds the profiling file. The device end mainly completes the inference and deployment of the model and returns the profiling file to the host end. Tables 4 and 5 are the experimental configurations of the host end and the device end.

4.2. Simulation Experiment. In order to verify the linear relationship between the calculation and the time of the convolutional layer, actually to find out α_t, we select the single-channel image with the input size of $28 * 28$, the size of the convolution kernel are 3, 5, 7, the convolution step length is 1, and the padding mode selects 'SAME' and 'VALID'. For each candidate value of the convolution kernel, the output channels come from 64 to 128, so we have 768 different convolutional layers. As for the output channels,

Table 4: The configuration of upper-computer.

Experiment	Configuration
Operating system	Ubuntu 16.04 LTS 64-bit
Processor	Intel core i5-9400@2.90 GHz ∗ 6
Memory	16 GB

Table 5: The configuration of board.

Experiment	Configuration
Chip	Xilinx® XC7Z020-1CLG484 C
Board	Xilinx® Zynq-7000
Memory	512 MB DDR3, 16 GB SD
Port	USB OTG 2.0

they can be calculated from the convolution kernel and the input channels. Then, we take the output channels as a self-variable. The convolution kernel and the number of input channels are known. We take 65 sets of output channels to make the least square regression to get every α_t of different convolutional layers, and finally, we can acquire a table of α_t.

We plot the data when $K = 1, 3, 5$, and the padding method is fixed to "SAME" in Figure 1. From the figure, it can be seen that the calculation time increases as the calculation increases. Moreover, there are a few singular points because of some accidental errors. In addition, we calculate the correlation coefficient between the calculation and the time when we take different K values, as shown in Table 6.

The correlation coefficients are all above 0.83, which indicates that there is a proportional relationship between the calculation and the time.

At the same time, in the simulation experiment, we discover that the padding model has little influence on measured time. In other words, when the padding model is set to 'SAME' or 'VALID', it can hardly distinguish the correct parameter settings by only relying on the leakage of the timing information under the current accuracy. It shows the error between the collected data when the padding adopts different models in Figure 2.

In order to verify the findings, we select the single-channel image which the input size is $28 * 28$, kernel size is 7, the convolution step is 1, the output channels come from [64, 128] and the padding is "SAME" and "VALID," so we have 256 sets of data, in order to verify the above findings we take the T-test and χ^2 test on the data.

4.2.1. T-Test. Introduction to T-Test. In the field of side-channel, the T-test is often used as a discriminator to verify whether two datasets share the same distribution. Welch's T-test considers two-sample sets and assumes H0: the data of the two sets obey the same normal distribution. Given a constant value (confidence level) then judge whether to accept or reject H0 under the confidence level, the methods are as follows [24, 28].

Let Q_0 and Q_1 are two datasets. The data size, mean, and variance of the two datasets are (n_0, μ_0, s_0^2) and (n_1, μ_1, s_1^2). The test statistics t and the degrees of freedom v are calculated as follows:

$$\begin{aligned} \mathrm{t} &= \frac{\mu_0 - \mu_1}{\sqrt{s_0^2/n_0 + s_1^2/n_1}}, \\ \mathrm{v} &= \frac{\left(s_0^2/n_0 + s_1^2/n_1\right)^2}{\left(s_0^2/n_0\right)^2/n_0 - 1 + \left(s_1^2/n_1\right)^2/n_1 - 1}. \end{aligned} \tag{6}$$

If the test statistic t satisfies the T distribution, the probability density function of accepting H0 and the probability of accepting H0 are as follows:

$$\begin{aligned} \mathrm{f}(\mathrm{t}, \mathrm{v}) &= \frac{\Gamma(v + 1/2)}{\sqrt{\pi v}\Gamma(v/2)}\left(1 + \frac{t^2}{v}\right)^{-v+1/2}, \\ \mathrm{P} &= 2\int_{|t|}^{\infty} \mathrm{f}(\mathrm{t}, \mathrm{v})\mathrm{d}t. \end{aligned} \tag{7}$$

$\Gamma(\cdot)$ is the gamma function and $\Gamma \mathrm{x} = \int_0^{\infty} t^{x-1}e^{-t}\mathrm{d}t$.

T-test is mainly to test whether the mean of the data is the same or not. When the difference in the means between Q_0 dataset and Q_1 dataset is large, the probability P of accepting H0 is small. When the difference of the means between Q_0 dataset and Q_1 dataset is small, the probability P of accepting H0 is large [29]. Meanwhile, the T-test ignores the influence of different variances on the data distribution, so it has a better detection effect for data with a large mean difference.

The Result of T-Test. The data we collect in the simulation experiment obeys the normal distribution or approximately obeys the normal distribution. We take a two-sample T-test and assume H0 as the data collected by the two padding models has little difference under the current accuracy. Then we output the test results with 90% confidence. The test results are shown in Table 7.where h = 0, t = 4.8820, and p = 0.7996 mean that H0 is accepted with 90% confidence, df and sd represent the estimation of the degree of freedom and the sample standard deviation.

4.2.2. χ^2-test. Introduction to χ^2-Test. The data is expressed as a contingency table of the number of leakages. The row j represents the amount of leakage, the column i represents the number of sample classifications, and $F_{i,j}$ represents the times of leaks corresponding to (i, j) and the total number of samples $\mathrm{N} = \sum_{i=0}^{r-1}\sum_{j=0}^{c} F_{i,j}$, the test statistics X and the degrees of freedom v are calculated as follows [30]:

$$\begin{aligned} \mathrm{f}(\mathrm{t}, \mathrm{v}) = \mathrm{f}(\mathrm{x}) &= \begin{cases} \dfrac{x^{v/2-1} \cdot e^{-x/2}}{e^{v/2} \cdot \Gamma(v/2)}, & x > 0, \\ 0, \ \text{otherwise}, \end{cases} \\ P &= \int_x^{\infty} f(x, v)\mathrm{d}x. \end{aligned} \tag{8}$$

In the above equation, $\Gamma(\cdot)$ is the gamma function.

If the statistic f is larger, the probability P-value is smaller, that is, the probability of accepting H0 is smaller, then H0 can be rejected with a high probability.

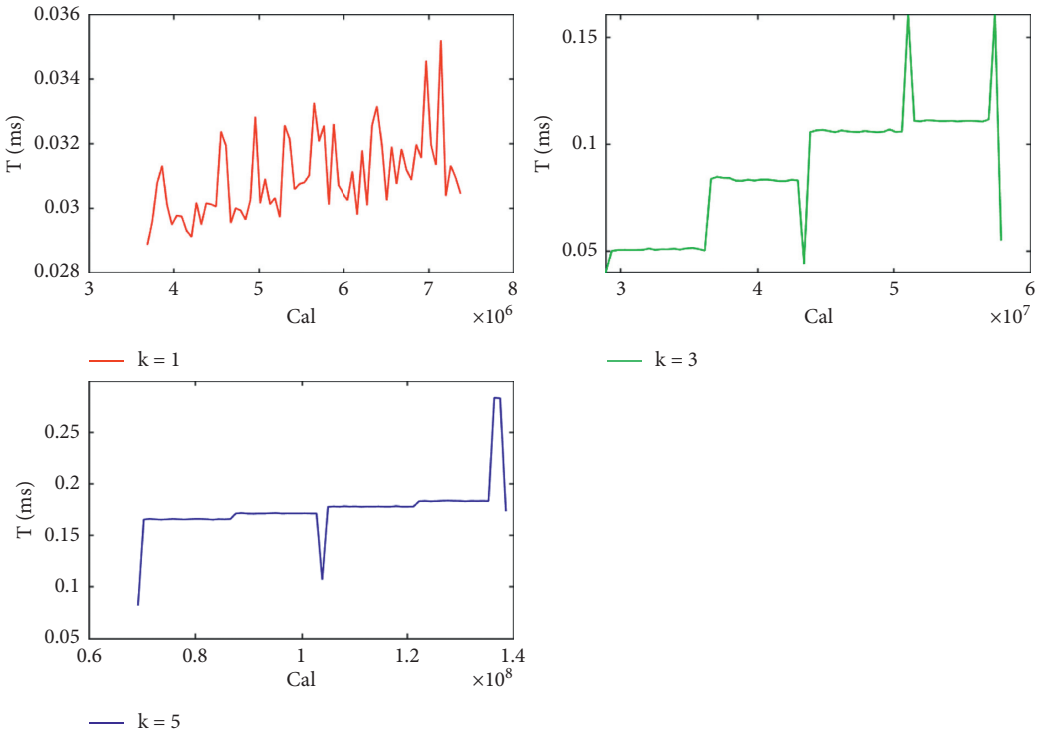

Figure 1: The relationship between calculation and measured time. Cal refers to calculation.

Table 6: The correlation coefficient between the calculation time and the calculation.

The size of convolution kernel	Correlation coefficient
1	0.8325
3	0.9664
7	0.9789

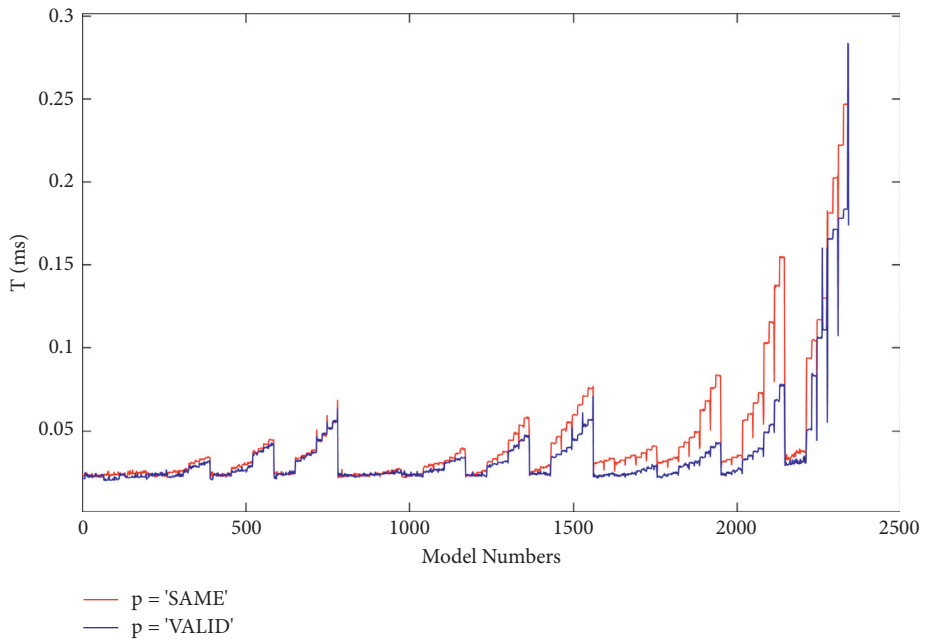

Figure 2: The difference of measured time when $p =$ "SAME" and $p =$ "VALID".

Table 7: The results of T-test.

Parameter	Value
h	0
t	4.8820
p	0.7996
df	15
sd	0.0081

The Result of χ^2-Test. We take a two-sample F-test on the two sets and H0 is assumed to be that the data collected by the two padding models has little difference under the current accuracy. The test results are shown in Table 8.where h = 0, t = 0.0951, and p = 0.4800 mean that accepting H0 with 90% confidence. df_1 and df_2 represent the test degree of freedom 1 and freedom 2.

After the T-test and χ^2-test, the two sets can be regarded as coming from the same distribution. We now verify the indistinguishability of the time leakage collected by the two padding models under the current accuracy, which provides a theoretical basis for simplifying the reverse process and reducing the sampling complexity in physical experiments.

4.3. Physical Experiment. We take the above formulas to find the linear relationship between the calculation and the time of different convolutional layers, then reverse the convolutional layer parameters. The reverse steps are essentially iterative processes of solving for the optimal solution of candidate values.

As to the α_t table, we get it in the simulation experiment. Now, we have 4680 models. For every model, other structures remain the same except for the above parameters. According to Table 9, we select the common single-channel input image size, convolution kernel size, input channels, output channels, step length, and padding models to establish a mapping library.

4.3.1. Attack Library. We generate the above models and train them, then send the models to the device end. It is worth noting that in order to reduce the accidental errors, we make every model perform the task of image recognition 100 times, so for each convolutional layer, we can collect 100 groups of data, as to all the models we have $4680 * 100$ groups of data.

According to formula (4), we use collected data to calculate the α_t table by the least square regression and show the residual of α_t, as shown in Figure 3.

It can be seen from the figure that α_t fluctuates within a small range for different input picture sizes. Among them, numbers 1 to 24 are models with an input image size of 7, numbers 25 to 48 are models with an input image size of 14, and the rest are models with input image size of 28. Figure 4 shows the fluctuations under different input sizes.

We select a K from candidate values, the feature map size I, and the number of input channels C_i of the first input layer are two known values for CNNs. So, we know I, C_i, K, then we can get a theoretical time according to Section 3.4. For each assumed K, we calculate the difference between theoretical time and the measured time. The minimum difference corresponding to the K will be regarded as the real convolution kernel size. Once the convolution kernel size is obtained, all subsequent convolution layer parameters can be calculated.

Table 8: The results of χ^2-test.

Parameter	Value
h	0
t	0.0951
p	0.4800
df_1	779
df_2	779

Table 9: Library of convolution layers.

Parameters of convolution layers	Candidate values
Size of input pictures	7, 14, 28
Size of the convolution kernel	1, 3, 5
Input channel	16, 64, 128, 256
Output channel	[64, 128]
Stride	1, 2
Padding model	SAME

4.4. Inversion of Common Models. After the attack library is established, we make an attack on some popular CNN models: an 8-layer AlexNet, an 18-layer SqueezeNet, LeNet, and ConvNet. For the reverse success rate, we define it as the following: for every assumed K, we can calculate all the parameters by formulas, if these parameters match the real parameters, the inversion is said to be successful. Table 10 summarizes the number of possible structures identified by our algorithm. From the table, the reverse success rate of common CNNs is higher than 66.67%.

4.5. Experimental Performance Analysis. In the simulation experiment, we get the α_t and theoretical time of different models. In physical experiment, we make use of the profiling tool to collect the timing leakage, then we build the mapping library. In order to prove the validity of our mapping library, we make a strict comparison and process between the profiling data: T-test and χ^2-test. Next, we mainly describe the effects from the following indicators: (1) the complexity of our inverse algorithm; (2) the distribution consistency of the measured value and the predicted value; (3) inverse success rate of convolutional layer; and (4) the bias of reversing and the success rate of adversarial attack.

For (1), the attacker only needs to get the profiling file to obtain the timing leakage. As to the profiling file, it is a tool for tuning. Any developer has access to it on the SoC board. Malicious attackers only need to buy the same type of device on the market. After obtaining the profiling file, the attacker can make attacks on the device [31].

For (2), the complex operation is mainly concentrated in matching the α_t, so the complexity is $O(n)$. Compared with the traditional model reverse attacks, the differences are shown in Table 11.

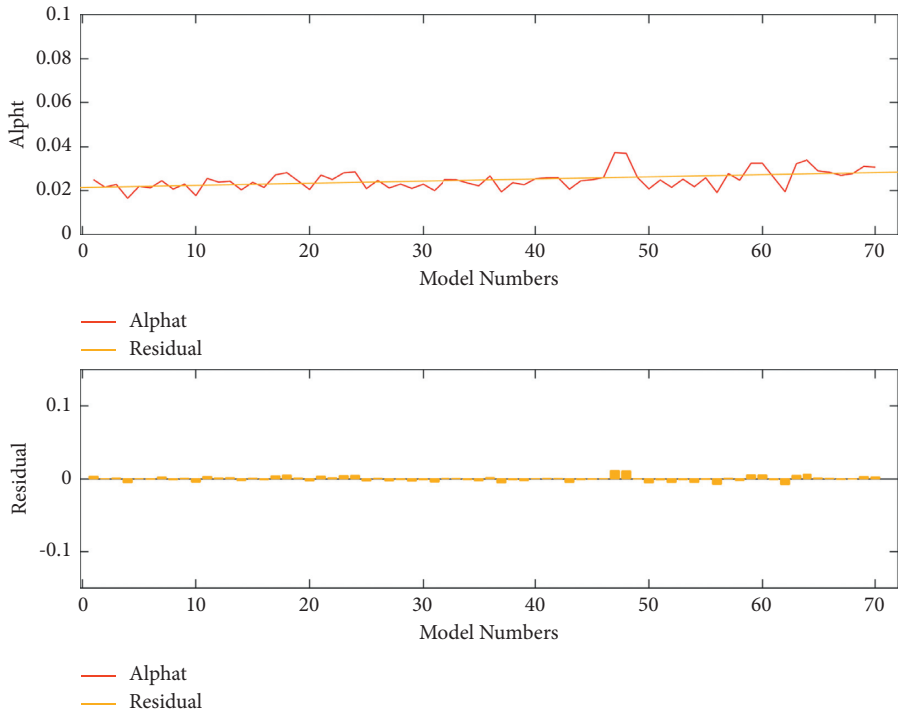

Figure 3: The trend of α_t and its residual.

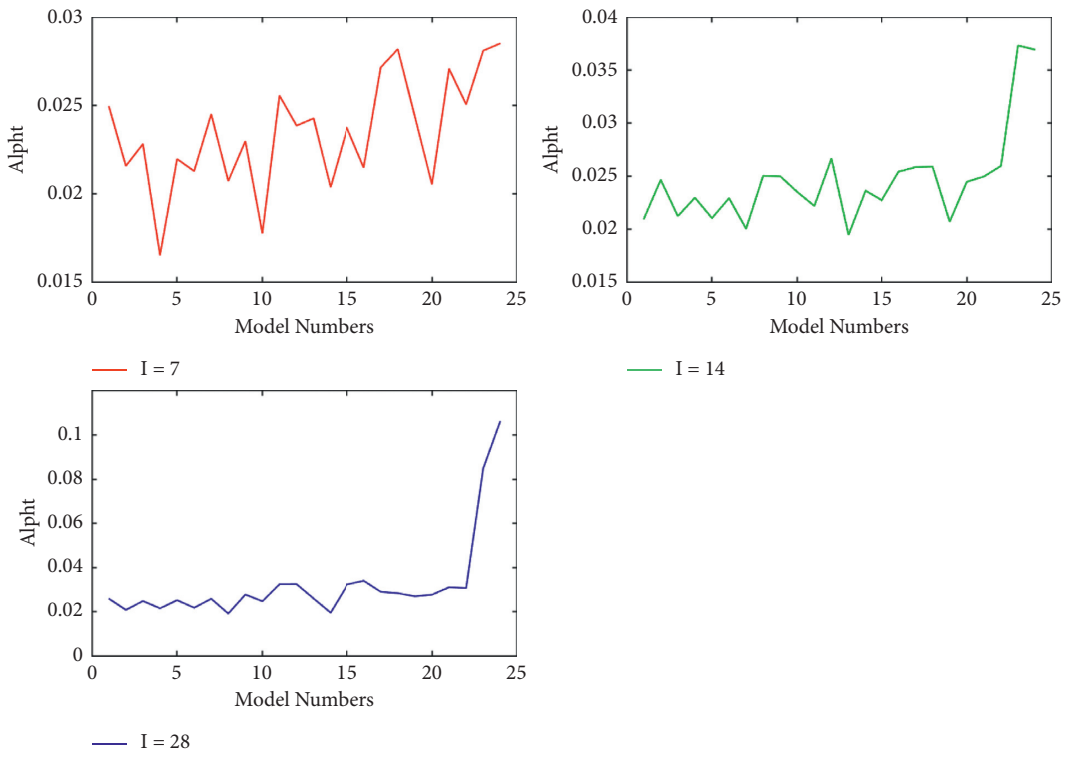

Figure 4: The trend of α_t for different input pictures.

4.5.1. Consistency of Measured and Predicted Value Distribution. The measured time refers to the time leakage collected on the device, and the theoretical time refers to the time data calculated from the formulas. The trend of theoretical time and measured time is shown in Figure 5. The red line represents the theoretical time and the blue line represents the measured time. It can be seen from the figure that the red line almost completely covers the blue line, that

Table 10: Potential structures of different networks and success rates.

	LeNet	ConvNet	AlexNet	SqueezeNet
Structure of networks	4	4	8	18
Candidate structures	4	6	12	25
Success rate	100%	66.67%	66.67%	72%

Table 11: Comparison with different algorithms.

Factors	Traditional inverse algorithm	Our inverse algorithm
Attack scene	Contact	Noncontact
White box	100%	100%
Gray box	≥ 82%	≥ 90%
Black box	≥ 70%	≥ 78%
Complexity	≥ O (N)	O (N)

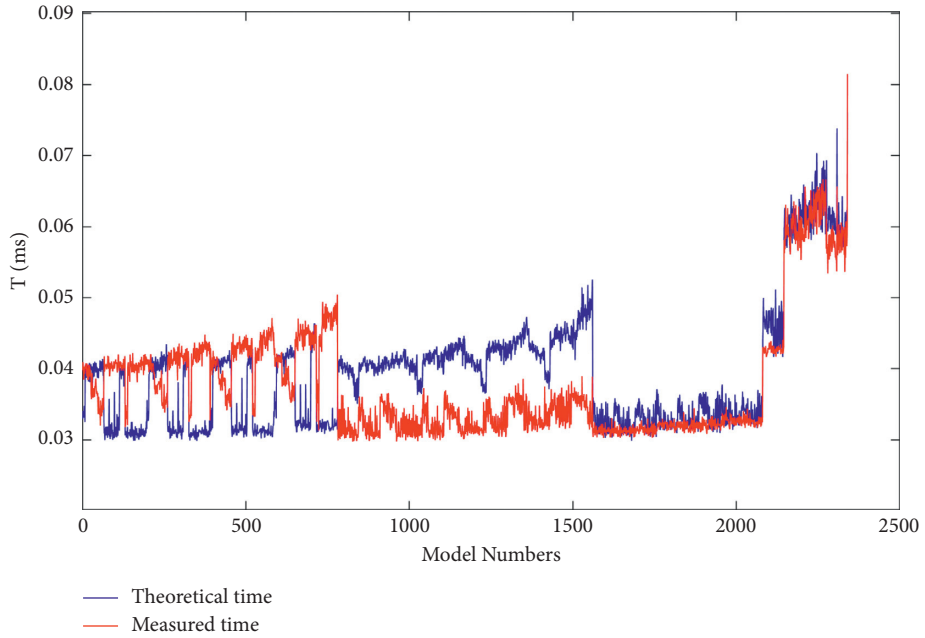

Figure 5: Measured time and theoretical time.

is, the difference between the theoretical time and the measured time is extremely small.

We take t-test and χ^2 test on the theoretical time and the measured time, the results are displayed in Tables 12 and 13.

It can be concluded from the test results that the measured time and the theoretical time can be regarded from the same distribution.

4.5.2. Inverse Success Rate of Convolutional Layer. Table 14 shows the success rates when the input image sizes are 7, 14, 28, 56, and 112. From the table, we can see that for convolutional layers with different input picture sizes, the reverse success rate is above 75%.

4.5.3. The Bias of Reversing and the Success Rate of Adversarial Attack. In order to further evaluate the reverse performance, we take MAE (mean absolute error) and RMSE (root mean squared error) to be the evaluation indicators of the inverse bias of the models. Meanwhile, we make an adversarial sample attack on the reversed model to evaluate the threat to the original model. Specifically, the accuracy of the reverse is measured by calculating the deviation between the theoretical values and the measured values. Moreover, the threat level of the inverse model to the original model is measured by the success rate of the adversarial attack [32].

Table 12: The results of T-test.

Parameter	Value
h	0
t	6.7001
p	0.9996
df	95
sd	0.0062

Table 13: The results of χ^2-test.

Parameter	Value
h	0
t	3.3820
p	0.5376
df	46
sd	46

Table 14: Success rates of different input pictures.

Size of input picture	Success rates (%)
7	82.01
14	74.56
28	78.52
56	85.47
112	89.35

Table 15: Results of indicators.

Indicators	Value
MAE	3.3527×10^{-6}
RMSE	2.2936×10^{-4}

MAE is the deviation between the theoretical time and the measured time, defined as follows:

$$MAE = \frac{\sum_i^L |r_i - \hat{r}_i|}{L}. \tag{9}$$

RMSE is defined as follows:

$$RMSE = \sqrt{\frac{\sum_i^L (r_i - \hat{r}_i)^2}{L}}. \tag{10}$$

In the above equation, r_i and $\hat{r}_i$ represent measured time and theoretical time. L represents the total groups of the theoretical time. *MAE* and *RMSE* are sensitive to large errors. So, the smaller the two values are, the better the inverse performance is. Table 15 shows the values of the two indicators.

Furthermore, we use the reversed model to generate a certain number of adversarial samples and make an adversarial attack on the original model. The result shows that the original model outputs the different label with a 97.73% probability compared with the former outputs after adding adversarial samples to the same picture, so the inverse model can produce enough threats to the original model.

5. Conclusions

We study the time side-channel leakage when CNN is running on the SoC system and propose a reverse attack method based on the time side-channel. It uses the SoC system performance analysis tool to collect the time information of each network layer of CNN. According to the linear relationship between the calculation, the parameters, and the memory usage in different network layers, we use a profiling attack to establish a mapping library corresponding to different network layers and time, and finally we can reverse the entire model. The results show that the reverse success rates of common convolutional layers are above 78.5%, and the reverse success rates of common CNNs (such as AlexNet, ConvNet, LeNet, etc.) are all above 67.67%. Moreover, the 97.73% success rate of the adversarial attack shows that our reverse algorithm can make enough threats to the real model [33].

However, the method we proposed still has some problems. We assume that the attacker can obtain the memory access information in the profiling file, but not all SDK tools of the device can provide the purchasers with access to the information, so it will be a strong assumption in a real environment. However, the memory usage can be calculated by batch size and other settings, such as optimizers like SGD, momentum, and Adam. Attackers can take the above settings to calculate the memory usage by formulas to complete the attack [34]. Secondly, if the accuracy of sampling is lower than 0.01 s, the reverse effect will be greatly reduced. Next, we will focus on applying the reverse algorithm to actual application scenarios and evaluating the attack effect. [26].

Data Availability

The dataset and codes are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the National Natural Science Foundation of China under Grant no. 61972295, the Wuhan Science and Technology Project Application Foundation Frontier Special Project no. 2019010701011407.

References

[1] W. S. McCuloch and W. Pits, "A logical calculus of the ideas immanent in nervous activity," *Bull let in of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[2] Y. Bengio, P. Lamblin, and D. Popovici, "Gredylayer wise training of deep networks," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 153–160, Vancouver, Canada, 2007.

[3] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, 2009.

[4] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[5] G. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[6] R. Salakhutdinov and G. Hinton, "Deep Boltzmann machines," *Journal of Machine Learning Research Proceedings Track*, vol. 9, no. 1, pp. 448–455, 2009.

[7] L. Chang, X. Deng, M. Zhou et al., "Convolutional neural networks in image understanding," *Acta Automatica Sinica*, vol. 42, no. 9, pp. 1300–1312, 2016.

[8] G. Hinton, L. Deng, and D. Yu, "Deep neural networks for acoustic modelling in speech recognition:The shared views off our research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[9] Y. Sun, W. Xiao Gang, and T. Xiao, "Deep learning face representation from predicting 10000 classes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1891–1898, Columbus, USA, June 2014.

[10] Z. FeiYan, J. LinPeng, and J. Dong, "Review of convolutional neural network," *Chinese Journal of Computers*, 2017.

[11] V. Nair, G. E. Hinton, and C. Farabet, "Rectified linear units improve restricted Boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning*, pp. 807–814, Haifa, Israel, 2010.

[12] B. Xu, Y. WangNai, Q. ChenTian, and M. Li, "Empirical evaluation of rectified activations in convolution network," 2015, https://arxiv.org/abs/1505.00853.

[13] B. Joseph and I. Mironov, *Cache-Collision Timing Attacks against AES*, Springer, Berlin, Germany, 2006.

[14] K. Jaret and K. Kavukcuoglu, "Marc'Aurelio Ranzatoetal what is the best multi-stage architecture for object recognition," in *Proceedings of the IEEE 12th International Conference on Computer Vision*, pp. 2146–2153, Kyoto, Japan, 2009.

[15] H. Xiao, B. Biggio, G. Brown, C. Eckert, and F. Roli, "Is feature selection secure against training data poisoning," 2018, https://arxiv.org/abs/1804.07933.

[16] G. Jiuxiang, W. Zhenhua, K. Jason et al., "Recent advances in convolutional neural networks," 2017, https://arxiv.org/abs/1512.07108.

[17] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," 2013, https://arxiv.org/abs/1301.3557.

[18] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," 2015, https://arxiv.org/abs/1511.08458.

[19] Z. Dong, Y. Wu, M. Pei, and Y. Jia, "Vehicle type classification using a semi-supervised convolutional neural network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 2247–2256, 2015.

[20] S. Bahrampour, N. Ramakrishnan, L. Schott, and M. Shah, "Comparative study of de ep learning software frameworks," 2016, https://arxiv.org/abs/1511.06435.

[21] S. Iofe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," 2015, https://arxiv.org/abs/1502.03167.

[22] X. G. Wang, "Deep learning in image recognition," *Communications of the CCF*, vol. 11, no. 8, pp. 15–23, 2015.

[23] Y. Taigman, M. Yang, and M. A. Ranzato, "Deep Face: closing the gap to human level performance in face verification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1701–1708, Columbus, USA, 2014.

[24] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1322–1333, ACM, 2015.

[25] J. Zhang, Z. Gu, and J. Jang, "Protecting intellectual property of deep neural networks with watermarking," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pp. 159–172, 2018.

[26] Y. Le Cun, B. Boser, J. S. Denker et al., "Handwritten digit recognition with a back-propagation network 07733 NIPS'89," in *Proceedings of the 2nd International Conference on Neural Information Processing SystemsJanuary*, 1989.

[27] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, pp. 3–18, IEEE, San Jose, California, CL, USA, 2017.

[28] S. Joon, M. Augustin, and M. Fritz, "Towards reverse-engineering black-box neural networks," *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, Springer, Cham, Switzerland, pp. 121–144, 2019.

[29] F. Tramer, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *Thorsten Holz and Stefan Savage*, pp. 601–618, USENIX Security Symposium, Austin, TX, USA, 2016.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pp. 770–778, IEEE Computer Society, Las Vegas, NV, USA, June 2016.

[31] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," *Lecture Notes in Computer Science*, pp. 16–29, 2004.

[32] A. Kurkin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 2016, https://arxiv.org/abs/1607.02533.

[33] W. Hua, Z. Zhang, and G. Edward Suh, "Reverse engineering convolutional neural networks through side-channel information leaks," in *Proceedings of the 55th Annual Design Automation Conference, DAC*, vol. 4, pp. 1–4, ACM, 2018.

[34] L. Wei, B. Luo, L. Yu, and Q. Xu, "I know what you see: power side-channel attack on convolutional neural network accelerators," in *Proceedings of the 34th Annual Computer Security Applications Conference*, pp. 393–406, ACM, 2018.