The Journal of Supercomputing (2024) 80:4423–4450
https://doi.org/10.1007/s11227-023-05631-3

Side-channel analysis based on Siamese neural network

Di Li[1,2] · Lang Li[1,2] · Yu Ou[1,2]

Accepted: 25 August 2023 / Published online: 13 September 2023

Abstract

In recent years, the combination of deep learning and side-channel analysis has received extensive attention. Previous research has shown that the key recovery problem can be transformed into a classification problem. The performance of these models strongly depends on the size of the dataset and the number of instances in each target class. The training time is very long. In this paper, the key recovery problem is transformed into a similarity measurement problem in Siamese neural networks. We use simulated power traces and true power traces to form power pairs to augment data and simplify key recovery steps. The trace pairs are selected based on labels and added to the training to improve model performance. The model adopts a Siamese, CNN-based architecture, and it can evaluate the similarity between the inputs. The correct key is revealed by the similarity of different trace pairs. In experiments, three datasets are used to evaluate our method. The results show that the proposed method can be successfully trained with 1000 power traces and has excellent attack efficiency and training speed.

Keywords Side-channel analysis · Deep learning · Siamese neural network · Information security

✉ Lang Li
lilang911@126.com

Di Li
lidi9007@163.com

Yu Ou
blink_zip@foxmail.com

[1] College of Computer Science and Technology, Hengyang Normal University, Hengyang 421002, China

[2] Hunan Provincial Key Laboratory of Intelligent Information Processing and Application, Hengyang 421002, China

1 Introduction

Artificial intelligence, big data, cloud computing, and smart devices have penetrated into all aspects of people's work and life [1, 2]. It is imperative to adopt new technical means and tools to deal with new security threats [3, 4]. The cryptographic chip is the foundation of the hardware security system. In the past, the security of the cryptographic chip was mainly considered in the mathematical design of the encryption algorithm [5]. In fact, the encryption devices are accompanied by physical leakages such as time [6], power consumption [7, 8], and electromagnetic [9]. The physical leaks bring hidden dangers to information security [10]. Kocher was the first to crack keys using physical information in the encryption process. His research subverts the previous perception of information security and opens up new research directions.

At this stage, side-channel analysis (SCA) is mainly divided into two categories, one is non-profiling SCA. It includes Simple Power Analysis (SPA) [11], Differential Power Analysis (DPA), Correlation Power Analysis (CPA) [12], and so on. Another is profiling SCA which has high attack efficiency and accuracy when the model is built successfully. It assumes that the attacker has an encryption device identical to the target. The attacker collects the power traces during the encryption process of the device and constructs a power consumption probability model. Then, the key is cracked by feature matching according to this model. Profiling SCA includes Template Attack (TA) [13]. However, TA relies on the distribution of target device power consumption data to obey a multivariate Gaussian distribution. It makes the profiling process difficult because the assumption does not always hold.

In recent years, researchers have begun to combine deep learning (DL) with SCA. DL can automatically learn features from data and generalize the representation of the data. This property of it shines in the SCA field. [14] introduces some research on different model structures in SCA. Among DL methods, models based on convolutional neural networks (CNN) seem to be the most effective methods, since they exhibit excellent performance in extracting relevant features from raw traces. Therefore, a large literature explores CNN model structures suitable for SCA [15–17]. At the same time, ensemble learning [18] and random convolution kernel [19] technology are applied to DL-SCA, and some new ways of combining DL with SCA are constantly being proposed in [20, 21]. These studies have confirmed that DL-SCA has an overwhelming advantage over traditional SCA in terms of attack capabilities.

1.1 Related work

Although DL-SCA showed a powerful attack capability, researchers realized that the time cost of training an SCA model is expensive compared to traditional methods. The training time is very long under huge training data. Not providing enough data may mean that we are not reaching a method's full potential. In more extreme cases, the method shows very poor performance [22]. Learning good features via DL is computationally expensive, and getting good performance can be a challenge

if little data are available. In this regard, Picek et al. [23] pointed out that profiling attacks should limit the ability of attackers to obtain power traces and build a corresponding analysis framework. Specifically, attackers are assumed to always find the best possible attack, which rarely happens in practice as in the limited-criteria Common Evaluation Method (CEM) of Common Criteria. Wang et al. [24] pointed out that if the attack is applied to a specific application, we will not be able to collect enough traces due to various constraints such as time, resources, and countermeasures. Specifically, the encryption to be attacked may be a subroutine, or the location of the encryption is not fixed. The cost of power trace harvesting at this point is expensive. Meanwhile, in the CHES-CTF 2023 competition, the proponents encourage attackers to reduce the number of traces used during training and build efficient and fast models. Therefore, researchers have done a series of studies to reduce the data dependence of the SCA model. Semi-supervised learning was introduced into the field of SCA by Picek et al. [25]. Labeling unknown power traces through semi-supervised learning effectively reduces the model's need for labeled power traces. However, their method cannot achieve the attack effect of the supervised learning model, even if the number of labels is increased. Kim et al. [26] augmented the data by adding noise to the power traces to reduce the data required for training. However, low-quality samples added during data augmentation can lead to reduced accuracy because the actual probability distribution of the observed data is usually not known during augmentation. Wang et al. [24] used DL techniques to enhance power traces. They generated a new dataset through Generative Adversarial Networks (GAN) for dataset expansion and data augmentation. However, the training of GAN networks also requires a sufficient number of power traces. In the case of insufficient data, GAN networks cannot generate high-quality power traces. Ito et al. [27] used Synthetic Minority Oversampling Technique (SMOTE) and cross-entropy ratio to solve the problem of the inconsistent number of instances in each target class. Since SMOTE is a data augmentation method based on linear interpolation, the variation of the augmented data is reduced when the training data are small. It leads to rapid overfitting of the model. Hu et al. [28] proposed a cross-subkey DL-SCA. It overhauls the traces of non-target subkeys as data augmentation to reduce the data required during training. However, this method is currently only available for specific devices, and the number of power traces in the training set must be greater than 5000 to be valid. All of the above studies have focused on data augmentation methods to create new samples. This paper provides a new perspective for reducing the data dependence of DL-SCA. We start from the model itself and use the unique training method of Siamese neural network (SNN) to solve this problem. The SNN is a recently proposed idea to solve the problem of overfitting caused by insufficient training samples [29]. It uses a pair of samples as input and can evaluate the similarity between input samples. Even with limited training samples, multiple inputs can be combined. In Koch's work, a one-shot image recognition model was built using SNN, and it achieved good results with limited training samples [30]. Compared to the work on classification models, our approach is more suitable for training models based on a small number of datasets. This helps reduce training costs. The SNN learns the similarity or difference between two inputs through contrastive learning, which makes samples with the same label as close as possible and samples with

different labels as far away as possible. Therefore, it focuses on the differences in power traces, such as leakage characteristics, which is beneficial for SCA. In other literature, only [31] mentioned the combination of SCA and SNN. They used one-shot learning to combine with SCA and achieved good results in the attack of asymmetric cryptography. However, they only considered the recovery of the scalar bits of the key.

1.2 Our contribution

In this paper, a Siamese neural network-based side-channel analysis (SNN-SCA) is proposed. The method combines the characteristics of CPA and TA. The main contributions of this paper are listed as follows:

1. The key recovery problem is transformed into a similarity measurement problem. It leads us to design a network that adopts Siamese, CNN-based architecture.
2. A sample combination strategy suitable for SCA is proposed, which can augment the data and simplify the key recovery step.
3. An adversarial sample selection strategy is proposed to improve model performance.
4. A series of experiments are performed to analyze the advantages of SNN-SCA in the case of limited power samples.

The rest of the study is organized as follows. Section 2 briefly introduces the background of SCA and DL. The SNN-SCA is explained in detail in Sect. 3. In Sect. 4, we specifically explain the structure of the model and the training strategy. The experimental verification will be discussed in Sect. 5. Finally, Sect. 6 concludes the whole study.

2 Preliminaries

2.1 DL-based profiling attacks

DL-SCA is seen as an extension of TA. The profiling and attack phases in TA are very similar to training and testing in DL. Therefore, the recovery key problem is transformed into a classification problem in DL. In the training phase, the attacker collects N_t power traces $T = \{t_i | i = 0, 1, \ldots N_t\}$. Each of the power traces t_i corresponds to a label with a known key and the label y_i can be an intermediate or Hamming weight (HW)/Hamming distance (HD) value. The power traces t_i, plaintext m_i and key k_i of each item are organized in pairs, and a so-called concept of training set in deep learning is obtained. The training data $D_{\text{profiling}}$ can be expressed as follows:

$$D_{\text{profiling}} = \left\{(t_i, k_i, m_i), i = 0, 1 \ldots N_t\right\}. \tag{1}$$

The main purpose of the profiling process is to establish a probability distribution function between the sensitive intermediate value and power consumption. In DL,

this probability function problem can indeed be solved by directly training a classifier, letting the classifier be f_c. It can be expressed as follows:

$$f_c(t_i) = (P_i | i = y_i) \tag{2}$$

In the attack phase, the power traces generated by each plaintext encryption process are collected to form an attack dataset. Unlike the profiling phases, the key is unknown and needs to be solved. The classifier f_c receives the power traces in the attack dataset and outputs a probability vector P_i. Attacks can obtain the key based on the probability vector and the plaintext.

2.2 Convolutional neural network

Convolutional neural network (CNN) is a deep neural network with features such as local connection and weight sharing. It includes convolutional layers and pooling layers. The convolutional layers of CNN are linear layers that use filters to convolve the input power traces. Different filters extract features from the filter window corresponding to the input power traces to obtain feature maps. When the input x is convolved by a filter g, the output of the conv layer is $y = f(g \oplus x + b)$. Define $z = g \oplus x$, fix stride to 1, in 1-D context, set the length of g to m, we have

$$z[i] = \left(\sum_{k=1}^{m+1} g[k] \cdot x[i - k + 1] \right) / m \tag{3}$$

High-level abstract features of the input data are extracted as the convolution operation goes deeper. These high-level abstract features are arranged side by side in deeper data dimensions. The geometric property makes the CNN very robust to the temporal deformation of the power traces.

The pooling layer of CNN is a nonlinear layer, which further processes the feature map obtained by the convolution operation to reduce the number of elements. The pooling layer statistically summarizes feature values at different locations on the feature map. In this paper, the power trace is a one-dimensional (1D) tensor. Therefore, the convolutional layers and filters used in this paper are one-dimensional.

2.3 Siamese neural networks

The SNN can be understood as an algorithm to measure similarity or correlation. It can be trained to capture features with fewer training samples. The idea behind it is not to learn to classify labels, but to learn to distinguish between input vectors. It learns an explicit relationship between two inputs. The inputs are mapped to a high-dimensional feature space and the corresponding representation is the output. The similarity of the two inputs is obtained by calculating the distance between the two input surface features. It can be trained on small sample datasets and is not easily disturbed by false samples, even if the training category is unknown and the number of training samples for a single class is very small [30]. The specific structure of the SNN is shown in Fig. 1, which

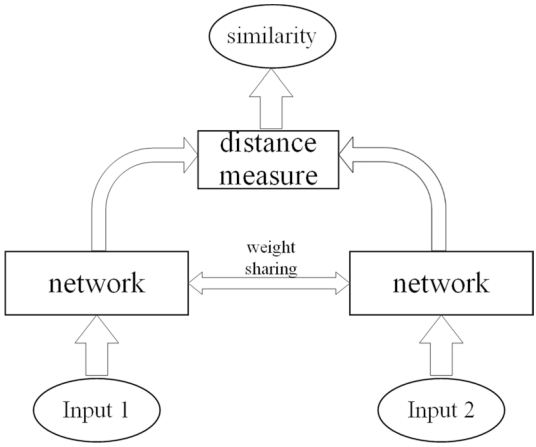


Fig. 1 Siamese neural networks architecture

has two branches; different branches usually use the same neural network and share the weights W. These two neural networks are feed-forward multilayer perceptrons that work in concert and are trained in a back-propagation fashion. Let X_1 and X_2 be two input vectors, and $G_W(X_1)$, $G_W(X_2)$ be the two-point mapping of X_1 and X_2 in the generated low-dimensional space. Then the metric function $E_W(X_1, X_2)$ is shown in (4).

$$E_W(X_1, X_2) = ||G_W(X_1) - G_W(X_2)|| \tag{4}$$

During the calculation of the metric function, the difference in embedding can use a contrastive loss to map samples with similar labels to close positions in the feature space and map samples with different labels to distant positions. The Contrastive Loss can be expressed as follows:

$$L = y * d^2 + (1 - y) * \max(\text{margin} - d, 0)^2 \tag{5}$$

where y is the label of the sample (0 means similar sample, 1 means dissimilar sample), d is the Euclidean distance between the feature vectors of two samples, and margin is a predefined threshold to control the boundary of similarity.

On the other hand, the difference in embedding is handled by dense layers for binary classification, which uses binary cross-entropy (BCE) loss to optimize the objective. The BCE loss expression is as follows:

$$\text{BCEloss}(p, y) = -(y * \log(p) + (1 - y) * \log(1 - p)) \tag{6}$$

where p represents the binary prediction probability of the model, the value range is [0, 1], and y represents the target label.

In this paper, we chose the second option. The L_1 distance is used for distance measurement, because the calculation of the L_1 distance function is simple, and it is not easily affected by outliers. The similarity is calculated by the sigmoid function.

3 Proposed method

3.1 Challenges for DL-SCA

Both DL-SCA with intermediate values as labels and HW/HD values as labels face numerous challenges. It can be summarized as follows:

1. The model is too data-dependent, and the training time is too long.
2. The number of training data classes is unbalanced, and the output distribution of the model is biased toward the distribution of labels.

First, classification tasks in DL usually assume that a given contains enough data for classification, and each class of data has distinct classification criteria. However, it is difficult to derive a uniquely identified intermediate or HW/HD value from a power trace due to the impact of noise and countermeasures. A power trace may correspond to multiple intermediate values or HW/HD values for parallel implementations of the algorithm. The number of classifications is 256 when the training data are labeled with intermediate values. There are some similarities between adjacent intermediate values. Therefore, the model needs more data to learn to complete the task. When an attacker builds a deep neural network that requires a large amount of data for training, the training time will far exceed traditional attack methods. More importantly, some cryptographic chips have control logic, which can only work at a fixed speed. The sampling time can be very long. In this case, data-dependent models will be difficult to train. Models require reduced data dependencies that avoid circumventing chip physics constraints. At the same time, overtraining on the same dataset can lead to overfitting. Therefore, the model must be forced to focus on the detailed differences in different classes of power consumption (distinguishing leakage). Second, the model output is 9 when the training data are labeled with HW/HD values. The HW distribution of the 256 intermediate values is severely unbalanced (e.g., the number of HW of 4 is much higher than the number of HW of 8); the distribution of HW/HD affects the model prediction distribution, especially when the relationship between power traces and HW/HD values is weak. During this time, the power traces are independent of the correct label. The neural network output probability is biased such that the probability of $HW = 4$ is the highest and the probability of $HW = 0$ or 8 is the lowest. It is likely to be affected by the imbalanced data when the label classification rate of the model remains unchanged at 27% ($\frac{70}{256} \approx 27\%$). The above problems limit the practical application of DL-SCA.

3.2 Framework of the proposed method

The SNN can be trained on a small number of samples and are not easily disturbed by erroneous samples. These characteristics are very suitable for SCA. It is worth exploring how the SNN structure can be used in the SCA context. According to the traditional SNN, we can combine the true power trace into

different data pairs and add them to the model training. However, the ultimate purpose of the model in DL-SCA is not to classify but to recover the key. The strategy of combining true power traces needs to rely on the known true power trace as a template to traverse the power trace with unknown information. For example, a model trained based on true power traces pair can classify and integrate the power traces of target devices. However, it is not clear what the intermediate value of the power trace is, because the key information is unknown. We expect that the trained model does not need to rely on training data samples as templates to analyze keys. Instead, the information of the intermediate state is obtained from the plaintext, and the key is obtained by analyzing the information of the intermediate state and the true power trace. Therefore, a combination strategy suitable for SCA is proposed.

In this method, we transform the key recovery problem into a similarity measurement problem of intermediate state leakage and power consumption. In classical SCA, the attacker can quantify the leakage of intermediate states through the leakage model. We call this set v the intermediate operating states of the entire device. x consists of the secret key and the input dependent variable on which the leaky function L acts. It describes how an implementation can leak information through a given side channel. Typically, the leakage model takes the form of an additive Gaussian noise function [32].

$$L(v) = L(v)_d + \mathcal{N}(0, \sigma) \tag{7}$$

where $\mathcal{N}(0, \sigma)$ is a Gaussian noise, and $L(.)_d$ is a deterministic function.

For a flip-flop in a register involved in data processing in the system, the power consumption is directly related to the processed data. This relationship is manifested at the CMOS gate circuit level as the charge and discharge of the load capacitor. At the register level, it is expressed as the flip-flop of 0 and 1 in the register, and at the operand level, it is expressed as the HW of the data before and after the execution of the instruction. This function is a HW function, so the $L(.)_d$ function is usually selected as HW, which is expressed as follows:

$$\mathrm{HW}(x) = \sum_{0 \le i < l} x^i \tag{8}$$

where l is the number of bits of x.

Therefore, generating simulated power traces based on HW values for intermediate states is correlated with power traces with the same HW label. We can introduce the SNN architecture to establish a similarity evaluation model when the input pairs of the same label are significantly differ from those of different labels. In this paper, we will generate simulated power traces based on the HW values of the intermediate states. Simulated and true power traces are used as a set of inputs for the model for training. A trained model is able to assess the similarity between inputs. Finally, the correct key is solved by the similarity of different power consumption pairs. The similarity measurement problem can be expressed as:

For the power traces t_i, the learning goal is to acquire an encoder f that meets the following conditions:

$$\text{score}(f(h_{i,t}), f(t_i)) >> \text{score}(f(h_{i,f}), f(t_i)) \tag{9}$$

where $h_{i,t}$ is the simulated power traces for the correct intermediate value, $h_{i,f}$ is the simulated power traces for the incorrect intermediate value, t_i is the true power trace, and score is a similarity measure function.

We design a combination scheme of SNN and SCA based on the above ideas. The overview of SNN-SCA is shown in Fig. 2.

3.3 SNN for SCA

In this section, we explain the details of the SNN-SCA. The data consisting of a simulated power trace and a true power trace are named trace pair for the convenience of description.

During the profiling (training) phase, the correct key is known. The attacker generates labels for simulated power traces and true power traces based on known plaintext and key information. The simulated power trace generation process is as follows:

$$h_{i,x} = Hw(v_{i,x}) + \mathcal{N}(0, \sigma) \tag{10}$$

where $h_{i,x}$ indicates the simulated power traces for the correct intermediate value when $x = t$, and it indicates the simulated power traces for the wrong intermediate value when $x = f$. We can control the variable $\mathcal{N}(0, \sigma)$ by setting the standard deviation of Gaussian noise. The noise level difference between the simulated power traces and the true power traces will affect the model training effect. Therefore, the standard deviation of Gaussian noise should also be used as a hyperparameter to

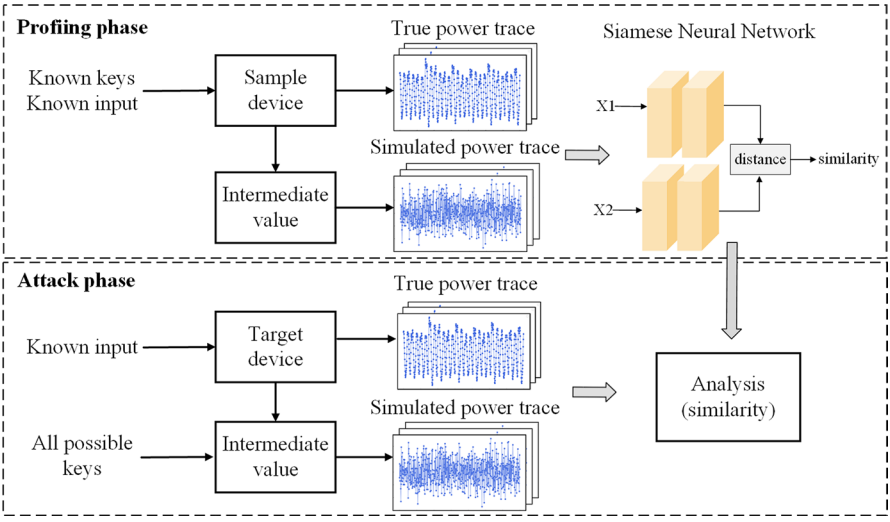


Fig. 2 Overview of SNN-SCA

debug this method. We verify this in the experimental part and find that the DL model is somewhat robust to changes in noise.

The purpose of training the model based on power traces is to be able to identify traces that have the same label. Let the simulated power traces in training set be H_s, $H_s = \{(x_i, y_i)\}_{i=1}^{n}$ and $y_i \in \{1, 2 \ldots, C\}$, and let the true power traces in training set be T_s, $T_s = \{(x_j, y_j)\}_{j=1}^{n}$ and $y_j \in \{1, 2 \ldots, C\}$. where n represents the number of training samples with labels and C represents the type of labels. Each set of trace pairs also corresponds to a label, which represents the similarity between the two samples. A label of 1 means that the two inputs are similar, and a label of 0 means that the two inputs are not similar. In other words, trace pairs with label 1 are also called positive samples, and trace pairs with label 0 are called negative samples. Suppose $[x_i, x_j]$ is a trace pair randomly selected from the training set. The labels of trace pairs are as follows:

$$\text{label}([x_i, x_j]) = \begin{cases} 1 & y_i = y_j \\ 0 & y_i \neq y_j \end{cases} \tag{11}$$

The model can obtain a multidimensional feature when different inputs X_1, X_2 enter the feature extraction layer w. Then, the units in the final multidimensional feature are flattened into a single vector, and their feature distances are calculated by the L_1 distance function. The process is shown in (12).

$$d = |w_1(X_1, \theta_{\text{encoding}}) - w_2(X_2, \theta_{\text{encoding}})| \tag{12}$$

where θ_{encoding} is the parameter of feature layer; w_i represents the feature layer of the ith channel.

The similarity p between two vectors can be calculated by logistic prediction. This step is as follows:

$$p = \varphi\left(\sum_j \alpha_j d^{(j)}\right) \tag{13}$$

where φ is the sigmoid function, α_j is the weight share of the jth component of d, $p \in (0, 1)$. It is learned by the model during training, weighing the importance of the component-wise distance.

During the testing (key recovery) phase, the key is unknown. The attacker enumerates all possible intermediate values according to plaintext and generates simulated power traces H', $H' = \{(x_a, y_a)\}_{a=1}^{n}$. For any power trace t_a in the test set, it will form a trace pair $[x_a, t_a]$ with each simulated power trace. The network will calculate the similarity p_i between them, and the predicted label $\hat{y}$ is as follow:

$$\hat{y} = \{y_m | m = \arg\max_i(p_i)\}, \tag{14}$$

Eventually, the similarity of all intermediate values is derived from the similarity of different trace pairs. The correct intermediate state value has the greatest similarity,

and the key can be deduced from the plaintext and intermediate value information. The process is shown in Algorithm 1.

Algorithm 1 recovery key

Input: The trained model $\hat{f}$, the set of power traces T, the set of plaintext M
Output: $realkey$

```
for i ← to size(M) do
    for k ← 0 to 256 do
        h_{i,k} = Simulate(Sbox(m_i ⊕ k))
        p_{i,k} = f̂(h_{i,k}, t_i)
    end for
end for
p_sum = SumByColumn(p_{i,k})
realkey = MaxIndex(p_sum)
return realkey
```

3.4 Adversarial example selection strategy

The choice of positive and negative samples is crucial for the SNN training process. Negative samples with significant differences contribute little to model learning, while the negative samples that are very close to each other in the feature space contribute more to the model training [33]. For tasks such as image classification or audio recognition, it is challenging to determine the similarity between them based on labels. For example, images labeled as "Jackson" and "Jackie" cannot be judged to be similar even if their names are similar. For the SCA domain, the correlation between label distance and feature distance is much stronger. Trace pairs with closer HW values are more correlated in SCA, and the model often misclassifies this set of trace pairs (this result can be seen in Sect. 5.1). An adversarial example generation strategy is constructed according to this phenomenon. We can divide the effective negative samples according to the HW value. The model should choose a simulated power traces adversarial sample $[t_i, h_j^*]$ whose HW value is different but very close to the HW value corresponding to the real traces.

Specifically, for true power traces t_i, its corresponding HW value is y_i. We use y_i as the axis to intercept the value of two units within the range of the HW value. The values in the selected range as candidate HW values for the simulated power traces. This candidate area is $D_s = \{x | y_i - 2 \leq x \leq y_i + 2, x \neq y_i\}$. The data selection process is shown in Fig. 3.

Finally, the simulated traces h_j^* are selected from the interval D_s and combined with the true power trace t_i to form an adversarial sample for model training.

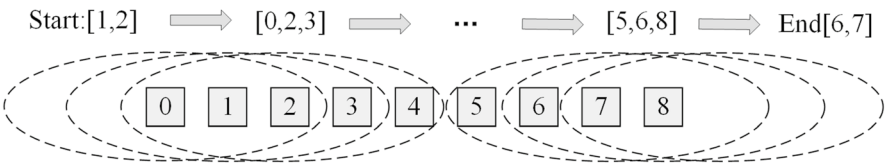


Fig. 3 Overview of the adversarial example selection strategy

3.5 Advantage of the SNN-SCA

In this section, we explain how SNN-SCA can reduce data dependencies data. According to the concept of trace pair mentioned in Section 3.3, we know that the proposed method uses a combination of simulated power trace and true power trace to augment the data. The combination of simulated power trace and simulated power trace does not make sense in our method. Let the number of true power traces of class i be $N_t^i, i \in [0, 8]$, and let the number of simulated power traces of class i be $N_s^i, i \in [0, 8]$. Then the total number of combinations will be $\sum_{i=0}^{8} \sum_{j=0}^{8} \left(N_t^i \cdot N_s^j\right)$. When applying the adversarial example selection strategy, the total number of generated samples N_{total} is as follows:

$$\begin{aligned} N_{\text{total}} = &\sum_{j=0}^{2}(N_t^0 \cdot N_s^j) + \sum_{j=0}^{3}(N_t^1 \cdot N_s^j) + \sum_{i=2}^{6}\sum_{j=i-2}^{i+2}(N_t^i \cdot N_s^j) \\ &+ \sum_{j=5}^{8}(N_t^7 \cdot N_s^j) + \sum_{j=6}^{8}(N_t^8 \cdot N_s^j) \end{aligned} \tag{15}$$

Therefore, the combination strategy can generate a large number of samples even when the power trace is constrained. The generated samples will be used conditionally, which will be explained in Sect. 4.2.

Second, SNN is more robust to imbalanced data. It does not directly output the class of each power trace, but the relationship (similar and dissimilar) between two input samples. The model predicts binary classification. Therefore, the data class distribution should obey the ratio of positive and negative samples. We only need to design a class-balanced data loader, which makes training the same number of positive and negative samples to solve the imbalanced data problem. While traditional DL-SCA classifies HW, the data class distribution obeys the HW distribution. The imbalance data problem becomes more difficult to handle because the HW distribution is more complex.

4 Design and analysis of SNN-SCA model

4.1 Structure of SNN-SCA model

In this section, we build a network suitable for our task based on the SNN architecture and CNN network. The CNN network originated in the field of image

recognition. Its current largest application field is image recognition. The experience gained in image recognition is not entirely suitable for the identification of power traces in the context of SCA. The useful feature of power consumption data is only a very small fluctuation in the operation data value. Its correlation is further weakened by the influence of noise and protection measures. It is completely unrecognizable to humans and the signal-to-noise ratio of the side-channel data is much lower than that of the image. The modeling strategy adopted by DL-SCA should be different.

Fortunately, there are many research papers on CNN applicable to the SCA field. We can gain experience building models from it. It is claimed in [16] that for synchronous power datasets with uniform distribution of interest points, it is recommended to use a small number of filters. Because adding convolutional blocks increases the risk of entanglement when interest points are temporally close to each other, in such datasets, they suggest that attackers can use CNNs with short filters (i.e., 1) that help focus their interest on local perturbations and significantly reduce the complexity of the network. For the ASCAD and DPAcontest v4 datasets in the first-order leakage environment, we refer to this concept and design a lightweight model. The model contains only two convolutional layers, each with stride 2 and padding 0.

For the SAKURA-AES dataset with a low signal-to-noise ratio, which leaks less information, the above strategy is not applicable. We refer to the model used in [17], whose core idea is to keep the amount of global information processed by different layers as constant as possible. The model is based on the VGG network and contains 5 convolutional layers, each with a stride of 5 and a padding of 1. The specific parameters of their single channel are shown in Table 1. The mark "FC-*n*, selu" means a fully connected layer of n neurons using selu activation function, "BN" means a batch normalization, "Conv*n*-*m*" refers to m convolutional kernels of size n, and "Average pooling, n by n" means an average pooling layer, whose pooling window size is n and the stride is also n.

The difference in embedding is handled by dense layers for binary classification, as shown in Fig. 4. It has two channels, which have the same components and share weights. Each channel is divided into three parts; the first part is the feature extraction layer, which contains 1D convolution, activation function, BatchNorm1d, and average pooling layer. Second, the linear layer consists of fully connected layers with different parameters. The last layer is the prediction layer. It computes the L_1 distance between each channel and gives an output unit. Finally, the similarity of the two inputs is given by the sigmoidal activation function.

4.2 Training the SNN-SCA model

Our goal is to optimize network parameters with few power traces. However, the problem of unbalanced positive and negative samples needs to be solved. Positive and negative samples are generated by (11). The training sample contains a trace pair consisting of simulated power traces H, true power traces T, and the similarity between them. Since a true power trace corresponds to only one HW value, other HW values are regarded as wrong HW values. It can form 1 positive sample

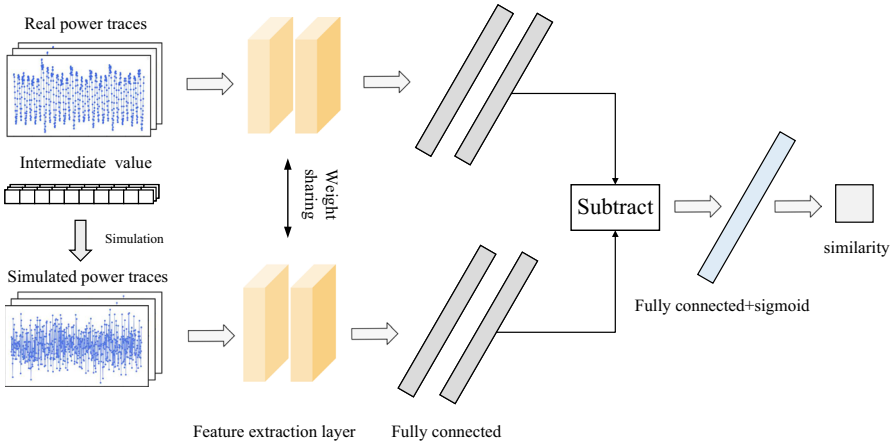


Fig. 4 Model structure

Table 1 Parameter details of single-channel model

ASCAD and DPAcontest v4	SAKURA-AES
5 weights layers	7 weights layers
input	input
Conv1-4, selu	Conv11-64, relu
BN	Average pooling, 2 by 2
Average pooling, 2 by 2	Conv11-128, relu
Conv1-8, selu	Average pooling, 2 by 2
BN	Conv11-256, relu
Average pooling, 2 by 2	Average pooling, 2 by 2
FC-20, selu	Conv11-512, relu
FC-20, selu	Average pooling, 2 by 2
FC-1, sigmoid	Conv11-512, relu
–	Average pooling, 2 by 2
–	FC-2048, relu
–	FC-1, sigmoid

and 8 negative samples with the simulated power traces. The number of negative samples is larger than the number of positive samples. This situation is common in other fields and is allowed in the training of SNN. It is necessary to control the number of positive samples and negative samples for model training. Therefore, we train according to the following principles:

1. Each training selects the sample that contributes more to the model training according to the adversarial sample strategy.
2. Set the probability of adding positive and adversarial examples in training to 50%.

More specifically, we first need to traverse the power traces T in the training set. When the power trace T is selected, we need to match the corresponding simulated power trace H according to the label (the probability of label value 0, 1 is evenly divided) to form a trace pair. The data sampling process is shown in Fig. 5.

4.3 Datasets

In this paper, three different datasets are used to evaluate our model. We are more concerned with first-order leakage when dealing with the similarity between simulated power traces and true power traces. The masks in the dataset as known values and thus are easily converted to unprotected scenes. A brief introduction to the three datasets is given below:

1. ASCAD: The ASCAD dataset is dedicated to evaluating the attack efficiency of DL models in SCA. It contains the power traces of AES mask implementation on ATMega 8518. There are 60,000 traces each with 100,000 sample points. The proposer keeps only 700 feature points between 45,400 and 46,100 to avoid useless and massive data processing. These feature points correspond to the time sample points of the third S-box leakage operation. The datasets are available at https://github.com/ANSSI-FR/ASCAD.
2. DPAcontest v4: The dataset is acquired on an 8-bit ATmega163 smart card controlled by the SASEBO-W board at a sampling rate of 500 MS/s [34]. Each power trace consists of 4000 features around the S-box part. In this paper, we only select 700 features between 550 and 1250 for experiments. The datasets are available at http://www.dpacontest.org/v4/42_traces.php.
3. SAKURA-AES: The dataset is unprotected AES-128 implemented on a XilinxSpardan-6 FPGA mounted on a SAKURA-G FPGA board designed for hardware security research and development. The AES-128 core is written in a round-based

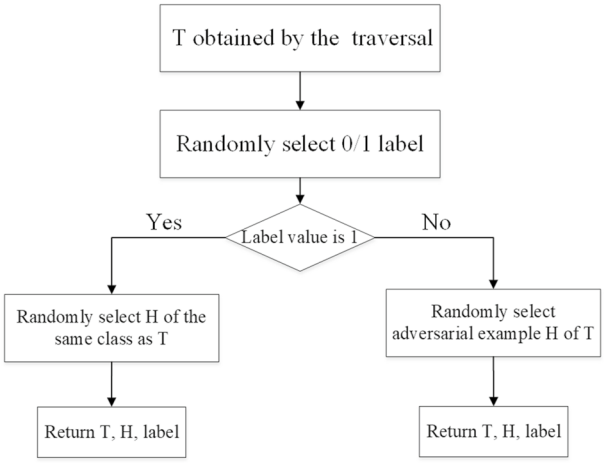


Fig. 5 The process of sample selection

architecture, requiring 11 clock cycles per encryption. The power traces are measured by monitoring the power waveform of the main FPGA core voltage. Power trace contains 10,000 sample points, and we choose to attack the first byte of the key. The segment from 1630 to 2330 is selected for analysis. The power trace collection process is shown in Fig. 6.

Since the leak point of SAKURA-G is in the last encrypted ciphertext added to the current encrypted round key, the sensitive intermediate value is

$$Z = \{z_i | z_i = c_{i-1} \oplus (p_i \oplus k_i), i \in [1, N_t]\} \tag{16}$$

where z_i is the intermediate value, c_i is the ciphertext, p_i is the plaintext, and k_i is the key. The data sample used in this paper is shown in Fig. 7.

4.4 Evaluation metrics

The validation accuracy of model training is a common metric in DL that expresses the ability of a model to classify data. This metric is often a direct reflection of a model's ability to achieve its application goals in the traditional machine learning domain. However, data classification is not the ultimate goal of the model in SCA. Its ultimate goal is to recover the key, and this metric can be evaluated by guessing entropy (GE). In this paper, we use the GE metric to evaluate the effectiveness of DL-SCA, which is the rank of correct keys. In this case, the trace pair with different labels will be added to the model. The similarity of different HW label will be obtained and finally converted to the similarity of intermediate values. Multidimensional similarity vectors can be obtained, as

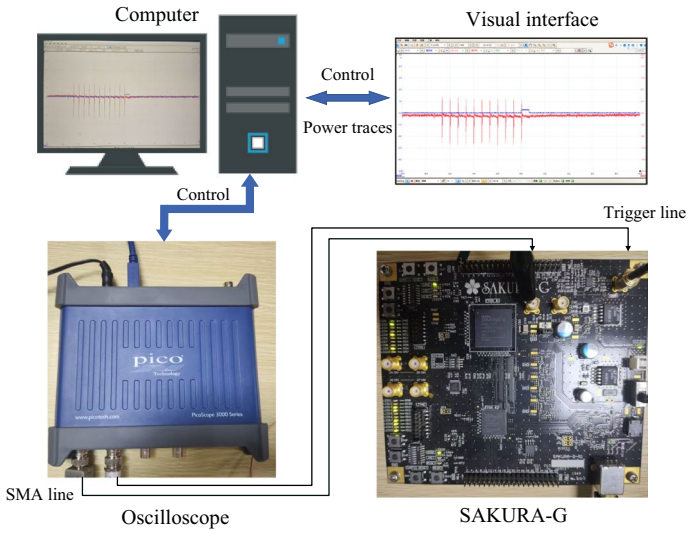


Fig. 6 Power traces acquisition platform

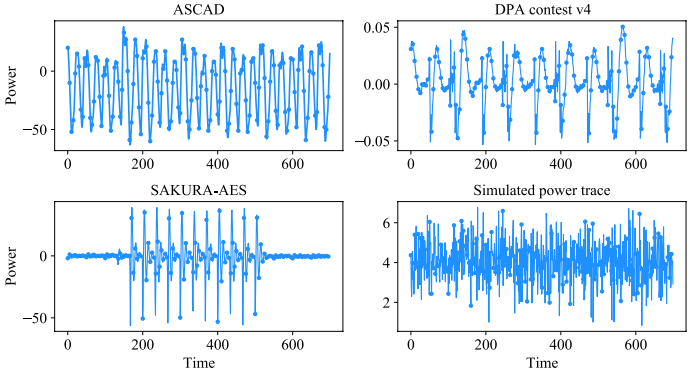


Fig. 7 The power traces used in this paper

the number of power trace increases. The rank metric is obtained through cumulative and sum operations, which is defined as follows:

$$\text{rank} = |\{k \in K | d[k] > d[k^*]\}| \tag{17}$$

where K is the key space, k^* is the correct key, and d is the similarity of 256 guessed keys, $|\{x\}|$ represents the number of elements in the set x.

The similarity of the correct key is the highest when the GE value equals 0. The minimum number of power traces required for a model attack is defined as N_{GE}. For a good estimation of N_{GE}, the traces in the attacking dataset are randomly shuffled and 20 N_{GE} are computed to give the average value for N_{GE}.

5 Experimental results

5.1 Experiment settings

The experiments were conducted on the PC with core intel i9-12900 H, GeForce RTX 3060 Ti GPU acceleration and the deep learning architecture PyTorch with Python language.

Before model training, it is very important to choose the appropriate hyperparameters. Different problems require different optimal hyperparameters. In addition to model parameters, hyperparameters include epoch (number of iterations), learning rate, optimizer type (such as SGD, AdaGrad, RMSProp, Adam), and batch size. To determine the best combination of parameters, a combination of grid search and empirical testing of the parameters can be used. Finally, the best parameters are selected for evaluation and combination. The selection of hyperparameters for different datasets in this paper is shown in Table 2.

5.2 Effect of sample selection strategies and loss function

In this section, we analyze the impact of different sample selection strategies and loss functions on model performance. The datasets used in the experiment are ASCAD and DPAcontest v4 datasets. For the DPAcontest v4 dataset, the standard deviation σ of Gaussian noise is set to 3.0 and the number of samples in the training set is 1000. For the ASCAD dataset, the standard deviation σ of Gaussian noise is 2.0 and the number of samples in the training set is 1200. The hyperparameters of the model are set consistently in the comparison experiments with different sample selection strategies.

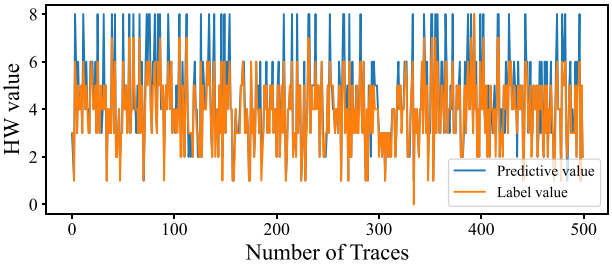
First, we randomly select the first 500 power traces in DPAcontest v4 to predict the HW values and compare the predicted values with the labeled values. Figure 8 shows the prediction results on DPAcontest v4 using different sample selection strategies. It can be seen from Fig. 8 that the model trained with the random sampling strategy has a large number of errors in the predicted values. Its label classification accuracy is 52.8%. The difference between the predicted values and the labels is usually within 2 because the similarity between adjacent HW values is high and the model can easily misidentify them. This is the reason why we designed the adversarial sample selection strategy. Many wrong predictions are corrected and the label classification accuracy is improved to 76.4% when the model is trained with the adversarial sample selection strategy.

Figure 9 shows the prediction results using different sample selection strategies on ASCAD. It can be seen from Fig. 9 that the model trained with the random sampling strategy is prone to misclassify HW labels with an adjacency distance of 2. Its label classification accuracy is 34.8%. Many wrong predictions are corrected and the label classification accuracy is improved to 46.4% when the model is trained with the adversarial sample selection strategy. The results show that the adversarial sample selection strategy can effectively improve the performance of the model.

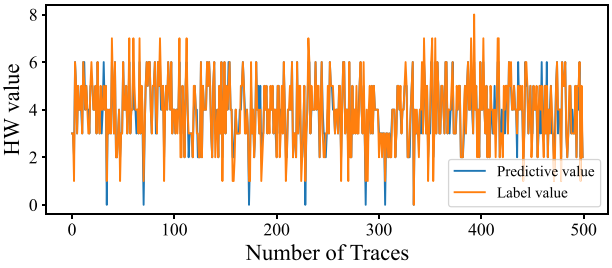
Then, we further discuss the effect of adopting BCE loss and contrastive loss on the attack results. In the experiment of contrastive loss, the margin parameter needs to be adjusted. We set the margin between 0.5 and 5.0 and test the attack effect. The experimental results are shown in Fig. 10. It can be seen that the margin setting has a greater impact on the attack results. Especially for the ASCAD dataset, the model using contrastive loss is very unstable. It is difficult to achieve good attack performance. For DPAcontestv4, the key can be cracked when the margin is set to 1, 3, and 5, but the same attack performance as BCE loss cannot

Table 2 Hyperparameter settings for different datasets

Dataset	Learning rate	Batch size	Epoch	Optimizer	Weight decay
DPAcontestV4	0.001	256	150	Adam	6e-5
ASCAD	0.01	256	150	Adam	6e-5
SAKURA-G	0.0005	256	200	Adam	6e-5

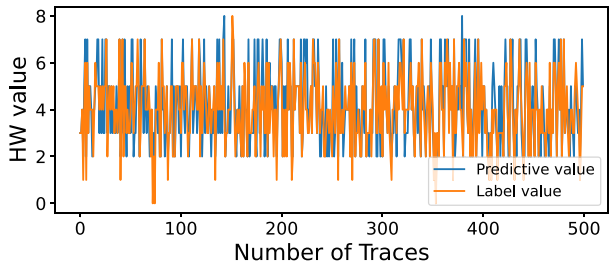


(a) Random selection strategy

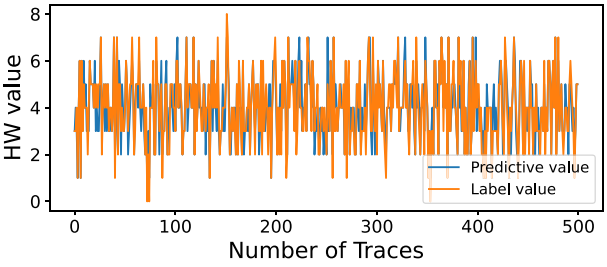


(b) Adversarial example selection strategy

Fig. 8 Effect of Different Sampling Strategies on Model Performance (DPAcontest v4)

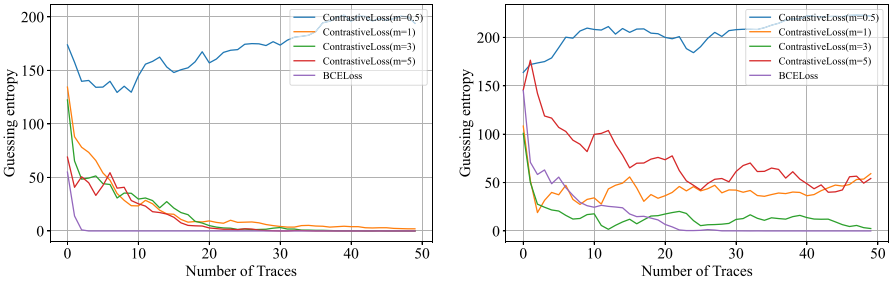


(a) Random selection strategy



(b) Adversarial example selection strategy

Fig. 9 Effect of Different Sampling Strategies on Model Performance (ASCAD)



(a) Results on the DPA contest v4 dataset (b) Results on the ASCAD dataset

Fig. 10 Effects of different loss function on guessing entropy

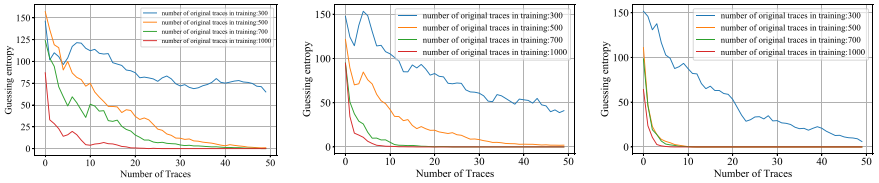
be achieved. Therefore, we recommend handling the difference in embeddings for binary classification through dense layers, which are more stable and efficient in the face of different datasets.

5.3 Results on DPAcontest v4 Dataset

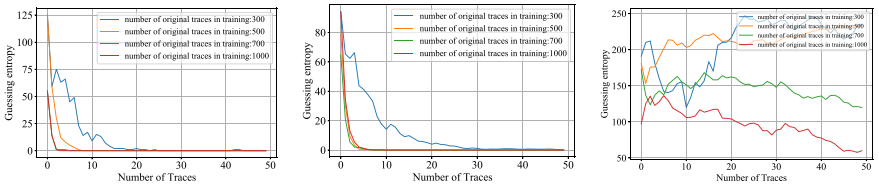
In this section, we perform performance testing on the DPAcontest v4 dataset. The trace pair is composed of the simulated power trace and the original trace without changing the data value of the original trace. The number of original traces in training is set to 300, 500, 700, 1000, respectively, to verify the performance of the model under the limited number of original traces. The test set is randomly selected to better evaluate the performance of the model.

First, we tested the effect of simulated power traces on N_{GE} for different standard deviations σ. The simulated power trace is generated by adding Gaussian noise to the HW values of the intermediate states. The standard deviation σ of the Gaussian noise is set to 0.1, 1.0, 2.0, 3.0, 4.0, and 10.0, respectively. This means we retrained 6 models on Gaussian noise with different standard deviations σ. The results evaluated on the testset are shown in Fig. 11. We can see that the simulated power traces generated by different noises have some influence on the model prediction results. The performance of the model is severely disturbed when the level deviation of the noise is set to 10. It cannot obtain effective attack results. Fortunately, DL technology can adjust the parameters of the model to adapt to different data in training when the level deviation of the noise is set in the range of 0.1–4.0. Table 3 lists the N_{GE} values achieved by the models trained by different trace pairs when the number of original traces in training is set to 1000. The trained model performs better when the standard deviation of the noise is set to 2.0 and 3.0. The N_{GE} of the models is reduced to 6, 3.

For the convenience of comparison, we provide the experimental results of SNN-SCA with 10,000 and 4000 training samples, respectively. Meanwhile, we reproduce the state-of-the-art experiments of Zaid et al. [16] on DPAcontest v4. The result is shown in Fig. 12, and Table 4 shows more experimental comparison results. It can be seen that our method can find a good balance between training cost and attack



(a) standard deviation $\sigma = 0.1$ (b) standard deviation $\sigma = 1$ (c) standard deviation $\sigma = 2$



(d) standard deviation $\sigma = 3$ (e) standard deviation $\sigma = 4$ (f) standard deviation $\sigma = 10$

Fig. 11 Effects of different Gaussian noises on guessing entropy

Table 3 Attack data in the case of Gaussian noise with different standard deviations

Standard deviation	Number	N_{GE}
$\sigma = 0.1$	1000	23
$\sigma = 1$	1000	14
$\sigma = 2$	1000	6
$\sigma = 3$	1000	3
$\sigma = 4$	1000	7
$\sigma = 10$	1000	>50

Number means the number of original traces in training

efficiency. Our method is basically maxed out in its ability to recover keys with 1000 training samples. This is the advantage of our method, which can achieve better attack efficiency with a very small training cost. There is no difference from the GE evaluation index when adding the training set to 4000 and 10,000. The comparison results show that our method can achieve excellent results even when the number of training samples is small. Compared to state-of-the-art-1 and state-of-the-art-2, the proposed method ($\sigma = 3$) reduces the number of training data requirements by 75% and 90%. It achieves the same attack efficiency with limited trace samples. Compared to GAN and RF, our method ($\sigma = 3$) can reduce N_{GE} by 204 and 17 with 1000 original traces. It is worth mentioning that the training time of the proposed method only needs 42 s in the case of 1000 training samples. These data show that the proposed method has a better performance in attack efficiency and attack cost.

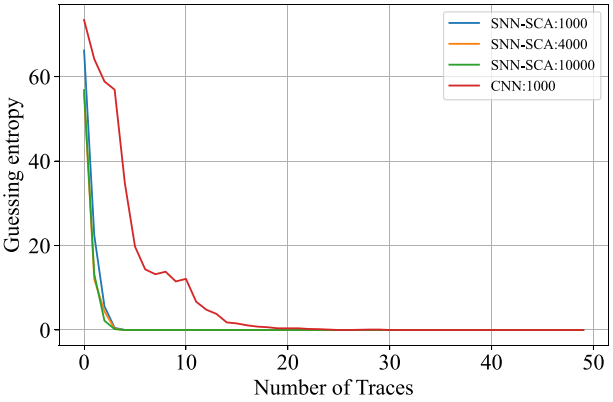


Fig. 12 Comparison results on DPAcontest v4, where "model: n" refers to the model trained under n samples

Table 4 Comparison on DPAcontest v4 dataset

Model	Number	N_{GE}
GAN [24]	1000	207
RF [26]	1000	20
MLP [35]	10,000	35
State-of-the-art-1 [16]	1000	20
State-of-the-art-1 [16]	4000	3
State-of-the-art-2 [23]	10,000	4
Stochastic model [36]	10,000	4
This work ($\sigma = 2$)	1000	6
This work ($\sigma = 3$)	1000	3
This work ($\sigma = 3$)	4000	3
This work ($\sigma = 3$)	10,000	3

Number means the number of original traces in training

5.4 Results on ASCAD dataset

In this section, we perform performance testing on the ASCAD dataset. The Gaussian noise standard deviation σ of the simulated power traces is set to 2.0. The trace pair is composed of the simulated power trace and the original trace without changing the data value of the original trace. The number of original traces in training is set to 600, 800, 1000, and 1200, respectively, to verify the performance of the model under the limited number of original traces. The test set is a random selection of power traces to better evaluate the performance of the model.

Figure 13 shows the evaluation results of the model with a different number of training samples. More detailed results are given in Table 5. From Fig. 13 and Table 5, we see that the model needs at least 800 original traces to obtain good attack efficiency. The model does not have sufficient ability to recover the key when

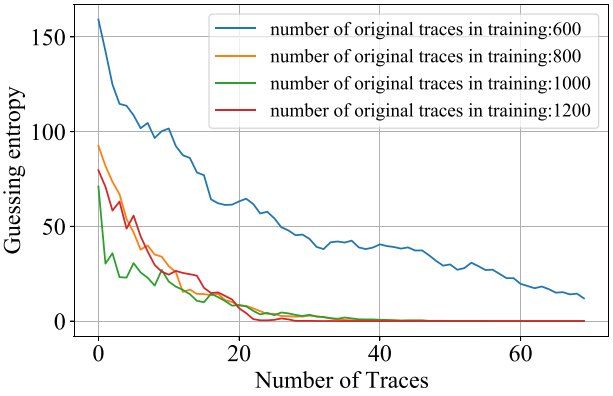


Fig. 13 The relationship between the number of training samples and guessing entropy ($\sigma = 2$)

Table 5 Comparison on ASCAD dataset

Model	Number	N_{GE}	Training time
This work ($\sigma = 2$)	600	>70	36 s
This work ($\sigma = 2$)	800	50	39 s
This work ($\sigma = 2$)	1000	45	43 s
This work ($\sigma = 2$)	1200	28	47 s

Number means the number of original traces in training

the number of original traces in training is 600. It cannot recover the correct key within 70 attack traces. As the number of original traces in training increases, the model can recover the key when 50, 45, and 28 attack traces are added to the model, respectively. Experimental result shows that the proposed method also performs well on the ASCAD dataset. At the same time, the proposed method has good performance in terms of time overhead. The model training time is 39 s, 43 s, 47 s, respectively, when the training samples are 600, 800, 1200.

5.5 Results on SAKURA-AES dataset

In this section, we perform performance testing on the SAKURA-AES dataset. It has a lower signal-to-noise ratio compared to the previous two datasets. The Gaussian noise standard deviation σ of the simulated power traces is set to 3.0. The trace pair is composed of the simulated power trace and the original trace without changing the data value of the original trace. The number of original traces in training is set to 1000, 1200, 1400, and 1600, respectively, to verify the performance of the model under the limited number of original traces. The test set is randomly selected to better evaluate the performance of the model.

Figure 14 shows the evaluation results of the model with a different number of training samples. More detailed results are given in Table 6. From Fig. 14 and

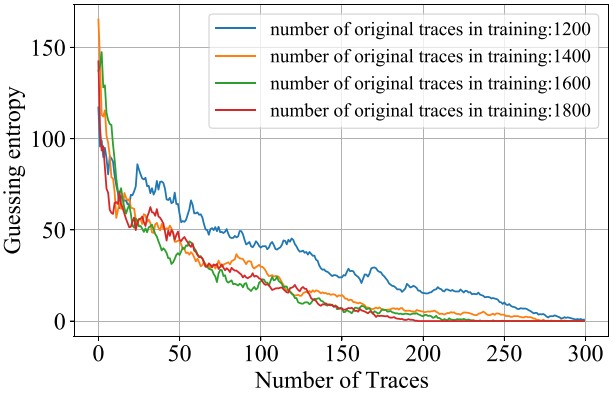


Fig. 14 The relationship between the number of training samples and guessing entropy ($\sigma = 3$)

Table 6 Comparison on SAKURA-AES dataset

Model	Number	N_{GE}	Training time
This work ($\sigma = 3$)	1200	>300	80 s
This work ($\sigma = 3$)	1400	276	89 s
This work ($\sigma = 3$)	1600	234	97 s
This work ($\sigma = 3$)	1800	197	105 s

Number means the number of original traces in training

Table 6, we see that the model needs at least 1400 original traces to obtain good attack efficiency. The model does not have sufficient ability to recover the key when the number of original traces in training is 1000. It is unable to successfully recover the key within 300 attack traces. As the number of original traces in training increases, the model can recover keys under the number of attack traces of 276, 234, and 197, respectively. Due to the larger amount of model parameters used by SAKURA-AES dataset, the model training time is longer than the previous two experiments. The model training time is 89 s, 97 s, 105 s, respectively, when the training samples are 1200, 1400, 1800. Generally speaking, the running time is still within the acceptable range. Experimental result shows that the proposed method also performs well on the SAKURA-AES dataset.

5.6 Discussion

The experimental results in the previous part confirm that the introduction of SNN architecture can effectively reduce the data dependence of the model. In this section, we further discuss what may cause the attack to fail. Attack failure in SCA can be defined as the failure to obtain the expected sensitive information during the attack process. It can be expressed in this paper that the ranking of the correct key is not

the highest among the 256 guessed keys. We discuss the reasons for key recovery failure from the following three aspects:

1. Model or hyperparameter selection is unreasonable.

Blindly transferring models from other fields such as images to the SCA field may not achieve good results. Attackers need to build reasonable models based on the power consumption leakage of the target device. This article recommends the use of CNN networks to build SNN single-channel models, which can refer to some classic CNN modeling strategies in the SCA field. We propose that the difference in embeddings is classified by a binary dense layer, trained with a BCE loss. In addition, the choice of hyperparameters may also lead to classification failure. Hyperparameters include learning rate, regularization parameters, etc. Choosing unreasonable hyperparameters may lead to overfitting or underfitting of the model, making it impossible to obtain the correct key.

2. The noise standard deviation setting of the simulated power consumption is unreasonable.

In SNN-SCA, the choice of the noise standard deviation of the simulated power consumption affects the attack results of the model. Because the noise difference between the simulated power trace and the real power trace will affect the model training effect, therefore, the standard deviation of the Gaussian noise should also be used as a hyperparameter to tune the method. We verified this in the experimental section. If the standard deviation of noise selected by the model is too small, SNN-SCA may overfit the noise, resulting in poor attack effect. On the contrary, if the selected noise standard deviation is too large, then SNN-SCA may not be able to accurately capture the characteristics of the data, resulting in a complete failure of the attack. In addition, we found that the standard deviation setting of 1.0–3.0 is a good choice for most cryptographic devices, which is consistent with the noise level of general cryptographic devices.

3. The number of power traces is too small.

The SNN architecture can greatly reduce the demand for model training datasets, but there is also a limit threshold. If the number of power traces selected during training is less than the threshold, the SNN-SCA model cannot learn enough data features, resulting in the failure of the attack. Especially for devices with a lower signal-to-noise ratio, choosing a small number of power traces for training will limit the ability of the model to capture enough information. In the previous experiment, we gave the minimum training datasets for DPAcontest v4, ASCAD, and SAKURA-AES three different datasets to obtain better attack efficiency, which are 1000, 1200, and 1800, respectively. These data will change with the model structure and hyperparameters.

6 Conclusions

In this paper, we analyze the challenges faced by DL-SCA and show that DL-SCA should reduce its dependence on data. Therefore, the key solving problem in SCA is transformed into a similarity measurement problem. Unlike previous classification models, the purpose of model learning is to distinguish different objects. The

simulated power traces and the true power traces form trace pairs to expand the data. The best sample is selected based on the HW value of the trace. Experiments show that SNN-SCA can be successfully trained in a small number of trace samples and can achieve excellent performance. We emphasize that this type of research is not designed to force attackers to use a small number of metrics in the analysis phase, nor to limit the number of experiments in the hyperparameter tuning phase. Instead, it forces the SCA model toward a more lightweight design and reduces its reliance on the amount of power traces. In future work, we plan to combine the power consumption characteristics of different devices and use the features of SNN to implement cross-device attacks. On the other hand, there is a new similarity learning method named SimSiam [37] in the image domain. It proposes a new idea, which uses the "stop gradient" method to avoid the "collapsing solutions" of the model. It does not need to introduce negative samples in training. It will be very interesting to explore the combination of SimSiam and SCA.

Acknowledgements We are grateful to the anonymous reviewers for their insightful comments. This work is supported by the Hunan Provincial Natural Science Foundation of China (2022JJ30103), "the 14th Five-Year Plan" Key Disciplines and Application-oriented Special Disciplines of Hunan Province (Xiangjiaotong [2022] 351), the Science and Technology Innovation Program of Hunan Province (2016TP1020), Open fund project of Hunan Provincial Key Laboratory of Intelligent Information Processing and Application for Hengyang Normal University (2022HSKFJJ011, 2021HSKFJJ038).

Author contributions DL helped in conceptualization and writing—original draft. LL was involved in writing—review and editing and supervision. YO contributed to term, project administration, formal analysis.

Data availability Data will be made available on request.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Ethical Approval Not applicable.

References

1. Li J-H (2018) Cyber security meets artificial intelligence: a survey. Front Inf Technol Electron Eng 19(12):1462–1474
2. Chong K-S, Ng J-S, Chen J, Lwin NKZ, Kyaw NA, Ho W-G, Chang J, Gwee B-H (2021) Dual-hiding side-channel-attack resistant FPGA-based asynchronous-logic AES: design, countermeasures and evaluation. IEEE J Emerging Sel Top Circuits Syst 11(2):343–356
3. Moon J, Jung IY, Park JH (2018) IoT application protection against power analysis attack. Comput Electr Eng 67:566–578
4. Saeedi E, Kong Y, Hossain M et al (2017) Side-channel attacks and learning-vector quantization. Front Inf Technol Electron Eng 18(4):511–518
5. Wei Y, Xu P, Rong Y (2019) Related-key impossible differential cryptanalysis on lightweight cipher twine. J Ambient Intell Humaniz Comput 10(2):509–517
6. Van Cleemput J, De Sutter B, De Bosschere K (2017) Adaptive compiler strategies for mitigating timing side channel attacks. IEEE Trans Dependable Secure Comput 17(1):35–49

7. Ding Y, Shi Y, Wang A, Wang Y, Zhang G (2020) Block-oriented correlation power analysis with bitwise linear leakage: an artificial intelligence approach based on genetic algorithms. Future Gener Comput Syst 106:34–42
8. Zhang F, Guo S, Zhao X, Wang T, Yang J, Standaert F-X, Gu D (2016) A framework for the analysis and evaluation of algebraic fault attacks on lightweight block ciphers. IEEE Trans Inf Forensics Secur 11(5):1039–1054
9. Go B-S, Le D-V, Song M-G, Park M, Yu I-K (2018) Design and electromagnetic analysis of an induction-type Coilgun system with a pulse power module. IEEE Trans Plasma Sci 47(1):971–976
10. Samadi Bokharaie V, Jahanian A (2022) Power side-channel leakage assessment and locating the exact sources of leakage at the early stages of asic design process. J Supercomput 1–26
11. Park A, Han D-G (2016) Chosen ciphertext simple power analysis on software 8-bit implementation of ring-LWE encryption. In: 2016 IEEE Asian hardware-oriented security and trust (AsianHOST). IEEE, pp 1–6
12. Chakraborty A, Mondal A, Srivastava A (2017) Correlation power analysis attack against STT-MRAM based cyptosystems. IACR Cryptol. ePrint Arch. 413
13. Chari S, Rao JR, Rohatgi P (2002) Template attacks. In: International workshop on cryptographic hardware and embedded systems. Springer, Berlin, pp 13–28
14. Maghrebi H, Portigliatti T, Prouff E (2016) Breaking cryptographic implementations using deep learning techniques. In: International Conference on Security, Privacy, and Applied Cryptography Engineering. Springer, Berlin, pp 3–26
15. Benadjila R, Prouff E, Strullu R, Cagli E, Dumas C (2020) Deep learning for side-channel analysis and introduction to ASCAD database. J Cryptogr Eng 10(2):163–188
16. Zaid G, Bossuet L, Habrard A, Venelli A (2020) Methodology for efficient CNN architectures in profiling attacks. IACR Trans Cryptograph Hardware Embed Syst 2020(1):1–36
17. Masure L, Dumas C, Prouff E (2020) A comprehensive study of deep learning for side-channel analysis. IACR Trans Cryptographic Hardware Embed Syst 348–375
18. Perin G, Chmielewski Ł, Picek S (2020) Strength in numbers: improving generalization with ensembles in machine learning-based profiled side-channel analysis. IACR Trans Cryptographic Hardware Embed Syst 337–364
19. Ou Y, Li L, Li D, Zhang J (2022) ESRM: an efficient regression model based on random kernels for side channel analysis. Int J Mach Learn Cybernet 1–11
20. Zhang L, Xing X, Fan J, Wang Z, Wang S (2020) Multilabel deep learning-based side-channel attack. IEEE Trans Comput Aided Des Integr Circuits Syst 40(6):1207–1216
21. Wu L, Perin G, Picek S (2022) The best of two worlds: deep learning-assisted template attack. IACR Trans Cryptographic Hardware Embed Syst 413–437
22. Mukhtar N, Batina L, Picek S, Kong Y (2022) Fake it till you make it: data augmentation using generative adversarial networks for all the crypto you need on small devices. In: Cryptographers Track at the RSA Conference. Springer, Berlin, pp 297–321
23. Picek S, Heuser A, Perin G, Guilley S (2019) Profiling side-channel analysis in the efficient attacker framework. Cryptology ePrint Archive
24. Wang P, Chen P, Luo Z, Dong G, Zheng M, Yu N, Hu H (2020) Enhancing the performance of practical profiling side-channel attacks using conditional generative adversarial networks. arXiv preprint arXiv:2007.05285
25. Picek S, Heuser A, Jovic A, Knezevic K, Richmond T (2018) Improving side-channel analysis through semi-supervised learning. In: International Conference on Smart Card Research and Advanced Applications. Springer, Berlin, pp. 35–50
26. Kim J, Picek S, Heuser A, Bhasin S, Hanjalic A (2019) Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. IACR Trans Cryptographic Hardware Embed Syst 148–179
27. Ito A, Saito K, Ueno R, Homma N (2021) Imbalanced data problems in deep learning-based side-channel attacks: analysis and solution. IEEE Trans Inf Forensics Secur 16:3790–3802
28. Hu F, Wang H, Wang J (2022) Cross subkey side channel analysis based on small samples. Sci Rep 12(1):1–11
29. Zhu J, Jang-Jaccard J, Singh A, Welch I, Harith A-S, Camtepe S (2022) A few-shot meta-learning based Siamese neural network using entropy features for ransomware classification. Comput Secur 117:102691
30. Koch G, Zemel R (2015) Salakhutdinov, Siamese neural networks for one-shot image recognition. In: ICML deep learning workshop, vol 2

31. Lee N, Hong S, Kim H (2022) Single-trace attack using one-shot learning with Siamese network in non-profiled setting. IEEE Access
32. Fumaroli G, Martinelli A, Prouff E, Rivain M (2010) Affine masking against higher-order side channel analysis, In: International workshop on selected areas in cryptography. Springer, Berlin, pp 262–280
33. Karpukhin V, Oğuz B, Min S, Lewis P, Wu L, Edunov S, Chen D, Yih W-t (2020) Dense passage retrieval for open-domain question answering. arXiv preprint arXiv:2004.04906
34. Martinasek Z, Dzurenda P, Malina L (2016) Profiling power analysis attack based on MLP in DPA contest v4. 2. In: 2016 39th International Conference on Telecommunications and Signal Processing (TSP). IEEE, pp 223–226
35. Wu L, Perin G, Picek S (2022) On the evaluation of deep learning-based side-channel analysis. In: International Workshop on Constructive Side-Channel Analysis and Secure Design. Springer, Berlin, pp 49–71
36. Zaid G, Bossuet L, Carbone M, Habrard A, Venelli A (2023) Conditional variational autoencoder based on stochastic attacks. IACR Trans Cryptographic Hardware Embed Syst 310–357
37. Chen X, He K (2021) Exploring simple Siamese representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 15750–15758