electronics

MDPI

Article

A Practical Non-Profiled Deep-Learning-Based Power Analysis with Hybrid-Supervised Neural Networks

Fancong Kong, Xiaohua Wang, Kangran Pu, Jingqi Zhang and Hua Dang *

School of Integrated Circuits and Electronics, Beijing Institute of Technology, Beijing 100081, China
* Correspondence: danghuabit@163.com

Abstract: With the rapid advancement of deep learning, the neural network has become the primary approach for non-profiled side-channel attacks. Nevertheless, challenges arise in practical applications due to noise in collected power traces and the substantial amount of data required for training deep learning neural networks. Additionally, acquiring measuring equipment with exceptionally high sampling rates is difficult for average researchers, further obstructing the analysis process. To address these challenges, in this paper, we propose a novel architecture for non-profiled differential deep learning analysis, employing a hybrid-supervised neural network. The architecture incorporates a self-supervised autoencoder to enhance the features of power traces before they are utilized as training data for the supervised neural network. Experimental results demonstrate that the proposed architecture not only outperforms traditional differential deep learning networks by providing a more obvious distinction, but it also achieves key discrimination with reduced computational costs. Furthermore, the architecture is evaluated using small-scale and downsampled datasets, confirming its ability recover correct keys under such conditions. Moreover, the altered architecture designed for data resynchronization was proved to have the ability to distinguish the correct key from small-scale and desynchronized datasets.

Keywords: side-channel analysis; differential deep learning analysis; hybrid-supervised learning; autoencoder; data resynchronization

Citation: Kong, F.; Wang, X.; Pu, K.; Zhang, J.; Dang, H. A Practical Non-Profiled Deep-Learning-Based Power Analysis with Hybrid-Supervised Neural Networks. *Electronics* **2023**, *12*, 3361. https://doi.org/10.3390/electronics12153361

Academic Editors: Wei Hu, Jiaji He and Haoqi Shan

Received: 4 July 2023
Revised: 1 August 2023
Accepted: 3 August 2023
Published: 6 August 2023

1. Introduction

Side-channel analysis is a technique used to uncover confidential information within a device by extracting the physical information leaked during the operation of an encryption device [1]. It exploits the side effects, or "side channels," that occur during the operation of a device, such as power consumption, electromagnetic radiation, etc. [2]. Side-channel attacks can be classified into two main categories: profiled attacks and non-profiled attacks. In a profiled attack, the attacker possesses a device that is identical to the encrypted one, and the key used by that device is both known and can be altered [3]. On the other hand, a non-profiled attack involves the attacker having only the encryption device itself, with the key being unknown but fixed. Although non-profiled attacks are more challenging to execute compared to profiled attacks, they hold greater practical value because their underlying assumptions are weaker than those of the latter.

Profiled attacks, such as template attacks [4,5], machine-learning-based attacks [6], and stochastic attacks [7,8], have demonstrated promising results in experimental settings. Profiled attacks require the attacker to conduct statistical analysis on the power signal. However, it is difficult to find an identical and fully controllable device to collect the traces. Among the non-profiled attack methods, correlation power analysis (CPA) [9,10] and differential power analysis (DPA) [11] are noteworthy. CPA exploits the Pearson correlation coefficient between the power trace and the intermediate value of the encryption algorithm to reveal the correct key, whereas DPA utilizes the differences in power signals to infer the correct key.

The rapid advancement of deep learning technology has enabled its extensive application in various data classification problems, such as image classification. Furthermore, deep learning techniques have been employed in several profiled attacks [12–15]. In recent years, the common ground between deep learning technology and the underlying principles of DPA in data classification has led to its utilization in non-profiled attacks.

In 2019, Timon proposed an innovative method called differential deep learning analysis (DDLA) that applies deep learning techniques for non-profiled attacks [16]. Through his research, Timon demonstrated the effectiveness of this method in compromising software-implemented Advanced Encryption Standard (AES) encryption algorithms. Furthermore, the DDLA method has the capability to decrypt encryption algorithms that are fortified with masks, showing its versatility and robustness.

In practical applications, the collection of power signals is often affected by various factors, including equipment and environmental conditions [17–19]. These factors can introduce noise into the collected signals, thereby interfering with the feature extraction process of data by the neural network. Currently, research on signal denoising in side-channel attacks predominantly revolves around the traditional CPA algorithm [20–22]. In 2021, Kwon introduced a method that employs an autoencoder to denoise power trace signals in side-channel attacks. Experimental results demonstrated an improved signal-to-noise ratio (SNR) of the processed signal in relation to key information [23]. Nevertheless, the application of this approach, utilizing an autoencoder to reduce noise, has not yet been incorporated into DDLA [24,25]. In this paper, a hybrid-supervised neural network incorporating a self-supervised autoencoder aiming to reduce noise and enhance feature in side-channel analysis is proposed.

One of the fundamental characteristics of neural networks is their reliance on a substantial amount of data to ensure the accuracy of algorithm outputs. Adequate data quantity enables neural networks to effectively extract features embedded within the data. In the realm of research on DDLA, studies typically employ large-sized datasets [26–28]. However, practical applications often encounter difficulties in collecting such extensive datasets due to device protection measures that restrict the number of user attempts [29]. In this paper, we introduce a novel architecture for non-profiled differential side-channel attacks, aimed at enhancing the performance of DDLA when confronted with small-scale datasets.

1.1. Our Contributions

1.1.1. Presenting a Novel Architecture with a Hybrid-Supervised Neural Network

In this paper, we introduce a novel architecture for non-profiled deep learning side-channel attacks, which incorporates the technique of hybrid-supervised learning referred to as hybrid-supervised side-channel analysis (HSSCA). The architecture consists of two main components. The first part is a self-supervised autoencoder, which effectively reduces noise in the power traces and enhances their features. The second part is a supervised deep-learning-based differential power analysis network utilized to classify power traces. The architecture is realized in two structures: multilayer perceptron (MLP) and convolutional neural network (CNN). Through this innovative architecture, the power traces are more easily classified, leading to increased visibility of the correct key compared to traditional DDLA networks. Notably, the proposed architecture demonstrates remarkable robustness even in the presence of masking countermeasures.

1.1.2. Applying the Proposed Architecture for Feature Enhancement

In engineering applications, encryption devices typically incorporate protection mechanisms that impose limitations on the number of attempts a user can make. As a result, acquiring a large amount of traces, which are often required for most non-profiled side-channel attacks, becomes exceedingly challenging. Additionally, for the power traces collected from the device to contain ample information about the leaked value, it is crucial that the data collection equipment possesses a high sampling rate. Typically, the sampling rate needs to be several hundred times higher than the clock frequency. However, such

equipment is not easily accessible. Therefore, the development of analysis methods that require fewer data and a lower sampling rate would hold significant value. The experimental results demonstrate that the proposed novel hybrid-supervised architecture, designed to reduce noise and enhance features in the training data, yields improved performance even when working with significantly smaller and downsampled datasets.

1.1.3. Altering the Proposed Architecture for Data Resynchronization

Another commonly used countermeasure utilized on encryption devices is introducing desynchronization to the power traces. By adding a random jitter or delay in the program, the execution time of the encryption algorithm is different every time. Therefore, the positions of underlying features in the power traces are different. This method is low-cost and particularly effective against algorithms that heavily rely on the positional information of the power traces, such as MLP-DDLA. To cope with the method, the structure of the autoencoder has been modified in order for it to have the ability to realign traces. The test results demonstrate that the altered HSSCA implemented with a CNN structure exhibits the capability to discern the correct key even in scenarios when only a limited amount of misaligned traces are employed as input data.

1.2. Organization

The first part is the introduction, where a brief overview of the research background and the the highlight of our contributions are provided. The second part, called preliminaries, introduces key concepts such as deep learning, MLP and CNN structures, non-profiled side-channel analysis, autoencoder principles, and the structure of DDLA. In the third part, the architecture of HSSCA is presented, including its implementation using MLP and CNN, and several variants of the architecture are proposed. Particularly, an altered architecture with a new labeling scheme for data resynchronization is introduced in detail. The fourth part is experiment results; in this section, the experiment environment is introduced, and the comparison between the novel architecture and traditional DDLA on complete datasets is demonstrated. Furthermore, the experiment results of the architecture on reduced, downsampled and desynchronized datasets are demonstrated. In the end, the fifth part concludes the works of paper.

2. Preliminaries

In this section, relevant basic knowledge about the new architecture of HSSCA proposed in this paper is introduced.

2.1. Deep Learning

Deep learning (DL) is a novel research direction within the field of machine learning (ML) aimed at advancing its original objective of achieving artificial intelligence (AI). DL is designed to extract valuable information from data such as text, images, and sounds, contributing to their interpretation. It is primarily employed in supervised learning tasks, encompassing classification and regression problems. A neural network, an algorithm specifically designed for DL, is utilized to calculate the probability that the data belong to a particular class. The following Formula (1) provides a general depiction of the architecture of a neural network:

$$Output = Net(Input) \tag{1}$$

In Formula (1), $Input$ represents the input data fed into the neural network. $Output$ refers to the output of the network, which can include predicted values or classification results, etc. A neural network comprises a set of adjustable parameters that can be trained to optimize the performance of network toward its desired objective. During training, a key component is the utilization of a function referred to as the loss function that quantifies the difference between the desired output and the current output generated by the network.

2.2. Multi-Layer Perceptron and Convolutional Neural Network

A common neural network structure is a multilayer perceptron [30]. It is a type of artificial neural network with a forward structure that consists of an input layer, an output layer, and multiple hidden layers in between. In each layer of MLP, there are many perceptrons whose function is to weigh and sum the inputs of this layer and add bias. The data passing through the perceptrons in each layer is subjected to a nonlinear transformation using an activation function before being propagated to the next layer. In neural networks, a unit of a perceptron and an activation function is called a neuron. The operation of a neuron is demonstrated in Figure 1.

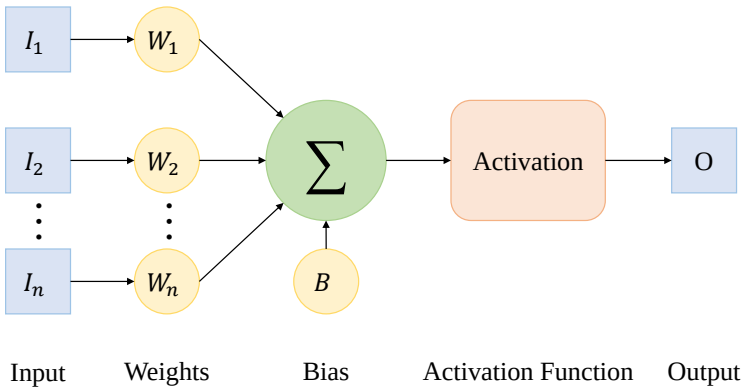

Figure 1. The structure of a neuron in a neural network.

A convolutional neural network is a type of artificial neural network designed for processing structured gridlike data, such as images or signals [31]. Not only it is widely used in computer vision tasks, such as image classification, but it also exhibits high performance dealing with 1D data [32]. A typical CNN consists of convolutional layers which involve a small filter/kernel over the input data and computing elementwise multiplications followed by pooling layers which reduce dimensions of data by downsampling. The advantage of CNN is that it extracts local features while preserving spatial relationships. Figure 2 shows the process of convolution and pooling.

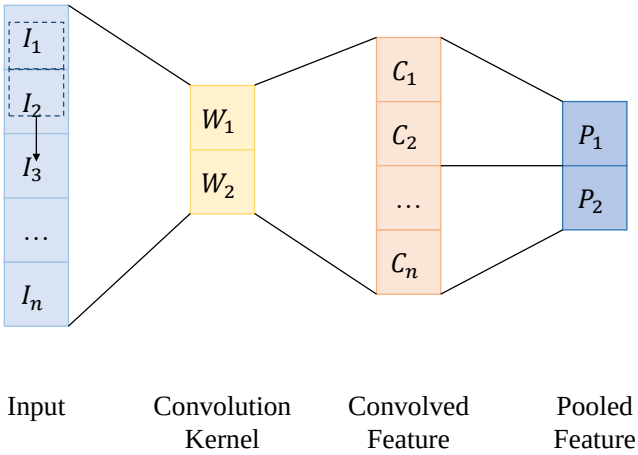

Figure 2. The process of convolution and pooling.

2.3. AES Encryption Algorithm

The advanced encryption standard is a widely used symmetric encryption algorithm [33]. It operates through multiple rounds of encryption. Each round, except the last one, consists of four steps: SubBytes, ShiftRows, MixColumns, and AddRoundKey. The number of rounds in AES iteration depends on the length of the plaintext. For a 16-byte plaintext, AES performs 10 rounds of iteration. Among these steps, SubBytes is the only nonlinear

operation. SubBytes is a simple operation that involves a lookup table. Specifically, it employs an S-box to substitute each byte of the plaintext. The calculation process can be summarized as follows:

$$D = Sbox(P \oplus K) \tag{2}$$

where D is the intermediate value, P is the plaintext, K is the key, and $\oplus$ denotes the XOR calculation.

2.4. Differential Deep Learning Analysis

In the context of classification using neural networks, when the label corresponding to the data is correct, the neural network can easily establish a relationship between the data features and the label. As a result, the loss function decreases rapidly during training, leading to an increase in the accuracy. In the case of side-channel attacks, the intermediate value of the encryption algorithm is directly linked to the power trace. Therefore, to create the dataset for each hypothetical key, the same power traces are used as training data and different intermediate values calculated from plaintext and hypothetical key are used as labels, and if the correct intermediate value is used as the label and the corresponding power trace is used as the data, the neural network can quickly learn the relationship between them during training. Conversely, if an incorrect intermediate value is used, the neural network training becomes slower and less effective. To exploit this behavior, an attacker can iterate through all possible key values within a specific key range **K**. When the correct key value is used, the neural network can observe the correlation between the known plaintext and the correct intermediate value during the training process, thus the loss and accuracy curve of the correct key is distinguished from other curves. Algorithm 1 outlines the steps involved in differential deep learning analysis (DDLA).

Algorithm 1 Differential Deep Learning Analysis

Input: N power traces $\{T_i\}_{1 \leqslant i \leqslant N}$ of length L with corresponding plaintexts $\{P_i\}_{1 \leqslant i \leqslant N}$

Output: Estimated secret key value K^* with the most distinguishable loss or accuracy curve

1: define the network Net and epochs n_e
2: **for** $K \in$ **K do**
3: Re-initialize the parameters of Net
4: Compute the hypothetical intermediate values $\{H_{i,k}\}_{1 \leqslant i \leqslant N}$
5: Compute the training labels with $\{H_{i,k}\}_{1 \leqslant i \leqslant N}$ according to the labeling scheme
6: Perform n_e epochs of deep learning training with $\{T_i\}_{1 \leqslant i \leqslant N}$ as the input of Net
7: Record the loss and accuracy curve of each key
8: **end for**
9: **return** K^* with the most distinguishable loss or accuracy curve

2.5. Autoencoder and Supervision Methods in Deep Learning

Autoencoder is an unsupervised learning model [34]. It leverages the backpropagation algorithm and optimization methods, such as gradient descent, to train a neural network using the input data X as supervision. The goal is to learn a mapping relationship that can reconstruct the input data, resulting in a reconstructed output X^R. Autoencoders are commonly used for tasks such as pretraining neural networks and reducing the dimensionality of data [35]. Self-supervised learning is a technique that transforms an unsupervised learning problem into a supervised one [36]. It differs from manually generating labels in traditional supervised learning. Instead, self-supervised learning calculates and generates labels from data samples to train neural networks. Hybrid-supervised learning refers to a deep learning approach that combines elements of both supervised learning and unsupervised or self-supervised learning, aiming to leverage the advantages of both approaches by incorporating labeled and unlabeled data during the training process.

3. Proposed Hybrid-Supervised Side-Channel Analysis

In this section, the model of hybrid-supervised side-channel analysis is elucidated, including its design principle, configuration, and variants for different situations and extreme conditions. The reason for combining a self-supervised autoencoder and a supervised deep-learning classification model is elaborated first; thereafter, two structures of it, that is, MLP-HSSCA and CNN-HSSCA, are introduced. Subsequently, the design of HSSCA aiming to solve problems encountered in practical applications is introduced and proved to be effective in theory, including the design for feature enhancement and data resynchronization.

3.1. Design Principle of Hybrid-Supervised Side-Channel Analysis

The construction of the neural network model for hybrid-supervised side-channel analysis is motivated by the observation that a self-supervised autoencoder network alone cannot directly extract the correct key from power traces. However, it can be used as a preprocessing step so that a supervised network can effectively calculate the correct key. Unlike in ideal cases, the traces collected in practical applications usually cannot be used directly due to various reasons. For instance, countermeasures deployed on encryption devices by designers or noise introduced during trace acquisition can severely diminish the performance of side-channel analysis. Therefore, it is essential to go through a preprocessing procedure to optimize the features of data. A hybrid-supervised side-channel analysis is composed of two parts: the first part is a self-supervised autoencoder, and the second part is an non-profiled deep learning network for data classification. To be noted, both the autoencoder and classifier network can be realized in two forms: MLP and CNN.

The self-supervised autoencoder follows a structure similar to a traditional autoencoder, but its purpose is not to reduce data dimensions. It consists of an encoder that compresses the original input data into a lower-dimensional representation and a decoder that reconstructs the data back to the same dimension as the input. After the autoencoder is trained, the decoder will not be removed because its goal is not to reduce data dimensions. However, its labels are different from the traditional autoencoder. Instead of using the original input data as labels, the labels in self-supervised autoencoders are computed from the original input data. The process of computing the label for each power trace is referred to as the labeling procedure, which will be introduced in detail in Sections 3.4 and 3.5. The main objective of the self-supervised autoencoder is to enhance the features of the data. Power traces collected from the device often contain interference or noise that can obstruct the classification of the traces. With the self-supervised autoencoder, the data are transformed to be more similar to the ideal data, which are assumed to be free of interference, making it easier to identify the correct key. The specific procedure for deriving the ideal data can vary depending on the countermeasures and noise present in the data. Algorithm 2 summarizes the procedure of training a self-supervised autoencoder.

Algorithm 2 The Procedure of training the Self-supervised Autoencoder

Input: N power traces $\{T_i\}_{1 \leqslant i \leqslant N}$ of length L with corresponding plaintexts $\{P_i\}_{1 \leqslant i \leqslant N}$

Output: Trained autoencoder AE^* for feature enhancement

1: Define the autoencoder AE and epochs n_{ae}
2: Compute the labels of each trace $\{L_i\}_{1 \leqslant i \leqslant N}$ according to the labeling procedure
3: **for** $1 \leqslant n \leqslant n_{ae}$ **do**
4: Perform an epoch of autoencoder training with $\{T_i\}_{1 \leqslant i \leqslant N}$ as the input of Net
5: **end for**
6: **return** trained autoencoder AE^* for feature enhancement

After the autoencoder has been trained, it is connected to the non-profiled deep learning side-channel network. Each power trace is passed through the self-supervised autoencoder initially, and the output is employed as the training data for the network. In

brief, the implementation of an HSSCA involves two phases: during the first phase, the self-supervised autoencoder is trained using power traces and the labels generated from them; in the second phase, the same power traces are processed using the trained autoencoder, and the resulting data is used to train the subsequent neural network. Following training with each hypothetical key, the estimated correct key is distinguished. Algorithm 3 introduces the working mechanism of HSSCA and demonstrates the modifications made to the traditional procedures in Algorithm 1. To verify the effectiveness of the proposed model, two of the most straightforward neural network models, MLP and CNN, are implemented to realize the model as they have been proved in related works to be the solid choices to perform non-profiled side-channel analysis and they can realize the most simple architecture of autoencoder.

Algorithm 3 Hybrid-Supervised Side-Channel Analysis (HSSCA)

Input: N power traces $\{T_i\}_{1\leqslant i\leqslant N}$ of length L with corresponding plaintexts $\{P_i\}_{1\leqslant i\leqslant N}$

Output: Estimated secret key value K^* with the most distinguishable loss or accuracy curve

define the network Net and epochs n_e
Train the self-supervised autoencoder AE with Algorithm 2
for $K \in$ **K do**
    Re-initialize the parameters of Net
    Compute the hypothetical intermediate values $\{H_{i,k}\}_{1\leqslant i\leqslant N}$
    Compute the training labels with $\{H_{i,k}\}_{1\leqslant i\leqslant N}$ according to the labeling scheme
    Calculate feature-enhanced training data $\{T_i'\}_{1\leqslant i\leqslant N}$ using the trained autoencoder AE^*
    Perform n_e epochs of deep learning training with $\{T_i'\}_{1\leqslant i\leqslant N}$ as the input of Net
    Record the loss and accuracy curve of each key
end for
return K^* with the most distinguishable loss or accuracy curve

3.2. Architecture of MLP-HSSCA

In the scenario of side-channel analysis, the MLP model is often considered a straightforward and effective choice for extracting features from the input data. When the classifier network is an MLP model, it is necessary for the autoencoder to have the same MLP structure.

1. The MLP self-supervised autoencoder primarily consists of two fully connected (FC) layers. The first layer takes an input of n_t data points and produces 50 output neurons. The second layer then takes these 50 inputs and outputs n_t neurons, matching the number of sample points in a data trace (n_t). After the first layer, a hard sigmoid activation function is applied. The output of the second layer is passed through a sigmoid activation function. The following formulas describe the operation of the MLP self-supervised autoencoder:

$$ENC_{MLP}(X) = HardSigmoid(FC_E(X)) \tag{3}$$

$$DEC_{MLP}(X) = Sigmoid(FC_D(X)) \tag{4}$$

$$AE_{MLP}(X) = DEC_{MLP}(ENC_{MLP}(X)) \tag{5}$$

$$Loss_{AE,MLP} = MSE(Label(X); AE_{MLP}(X)) \tag{6}$$

In the formulas above, ENC_{MLP} and DEC_{MLP} denote the encoder and decoder realized with FC layers, and $Label(X)$ denotes the labeling procedure of the autoencoder. After training, the output data of the autoencoder is closer to the ideal data cal-

culated by the labeling procedure. Figure 3 illustrates the architecture of an MLP self-supervised autoencoder.

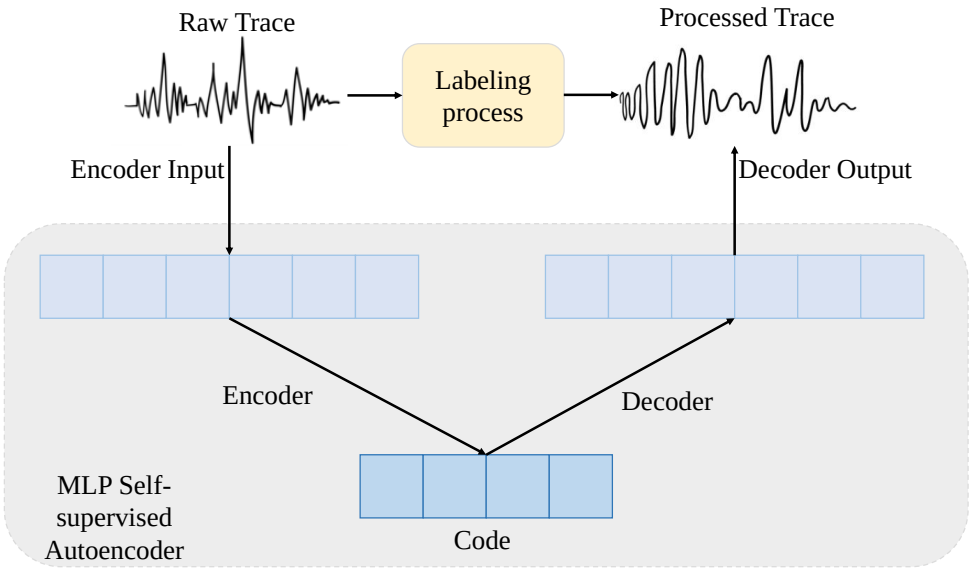

Figure 3. The architecture of MLP self-supervised autoencoder.

2. The architecture of the non-profiled deep learning side-channel network is an MLP model. It is composed of two hidden layers of 70 and 20 neurons, respectively. The first layer has n_t input data, and between each layer, the Rectified Linear Unit (ReLU) activation function is implemented, and at the output of the second hidden layer, the probability of the two classes is generated through a SoftMax activation function. For the labeling scheme, the Least Significant Bit (LSB) is chosen as the label of the data. The overall architecture of MLP-HSSCA can be observed in Figure 4.

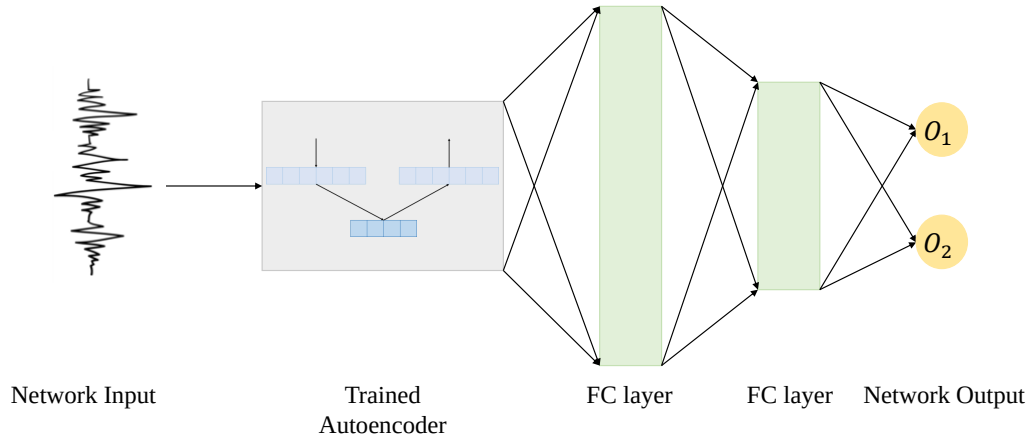

Figure 4. The architecture of MLP-HSSCA.

The details of hyperparameter of proposed MLP-HSSCA model are presented in Table 1 below.

Table 1. Hyperparameters of MLP-HSSCA.

Parameter	MLP Autoencoder	MLP Classifier
Hidden layer	1 (50 FC)	2 (70 FC $\times$20 FC)
Output size	700	2 (SoftMax)
Label	Plaintext/Binary	Binary
Initialization	Uniform	Uniform
Optimization	Adam	Adam
Learning rate	0.001	0.001
Batch size	1000	1000

3.3. Architecture of CNN-HSSCA

The network architecture of CNN determines one of its crucial properties: translation invariance. Because a convolutional filter of CNN would scan the complete data to yield a result. This means that local features within the data have the same impact on the output regardless of their position. As a result, when applying CNNs to side-channel attacks, 1D convolutional filters can effectively extract local information leakage from power traces, regardless of the leakage location within the traces. This characteristic becomes particularly valuable when the input traces are desynchronized. In such cases, CNNs can capture the same leakage and distinguish the correct key. Conversely, models such as MLP struggle with this task because their parameters are position-dependent.

1. The encoder of a CNN self-supervised autoencoder is constructed with convolutional layers and pooling layers, and the decoder is constructed with convolutional layers and upsampling layers. The first layer of the encoder is an 1D convolutional layer with 4 output channels, kernel size of 17, and padding of 8 to keep the 1D input data in the same length. Following is a MaxPooling layer with kernel size of 2 to make output data length half of the input. After that is an 1D convolutional layer with 8 output channels, same kernel size and padding followed by another same MaxPooling layer. The input of the encoder is 1D power trace data whose size is (n_t, 1), and the output of the encoder is a code whose size is ($n_t/4$, 8). As for the decoder, the structure is symmetric to the encoder, only the pooling layers are replaced by linear upsampling layers. In the autoencoder, after each convolutional layer there is a Randomized leaky Rectified Linear Unit (RReLU) activation funtion. In the end, the output size of the decoder is restored to (n_t, 1). The following formulas describe the operation of the CNN self-supervised autoencoder:

$$EL_{CNN,n}(X) = MaxPool(RReLU(Conv_{E,n}(X))) \tag{7}$$

$$ENC_{CNN}(X) = EL_{CNN,2}(EL_{CNN,1}(X)) \tag{8}$$

$$DL_{CNN,n}(X) = RReLU(Conv_{D,n}(Upsampling(X))) \tag{9}$$

$$DEC_{CNN}(X) = DL_{CNN,2}(DL_{CNN,1}(X)) \tag{10}$$

In the formulas above, $EL_{CNN,n}$ and $DL_{CNN,n}$ denote the encoding and decoding layers composed of convolutional layers, pooling layers, upsampling layers, and activation functions. Moreover, ENC_{CNN} and DEC_{CNN} denote CNN encoders and decoders. The architecture of a CNN self-supervised autoencoder is demonstrated in Figure 5.

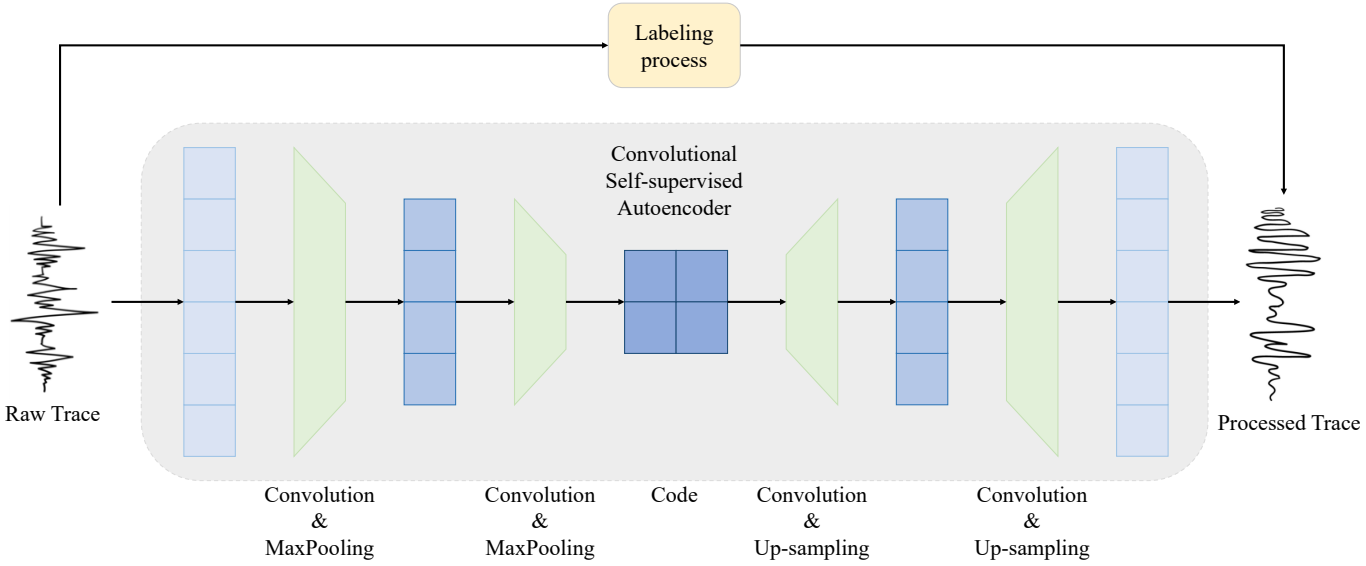

Figure 5. The architecture of CNN self-supervised autoencoder.

2. The CNN non-profiled deep learning side-channel network is composed of 3 hidden convolutional layers with the kernel size of respectively 32, 16, and 8, and they all have 4 output channels. The first layer has n_t input data, and after every convolutional layer, the ReLU activation function is implemented. Between convolutional layers, there are 3 AveragePooling layers. After the last AveragePooling layer, the data are flattened and passed through 2 hidden FC layers. At the output of the second hidden FC layer, the probability of the two classes is generated through a SoftMax activation function. The labeling scheme are the same as MLP. Figure 6 elaborates the structure of CNN-HSSCA.

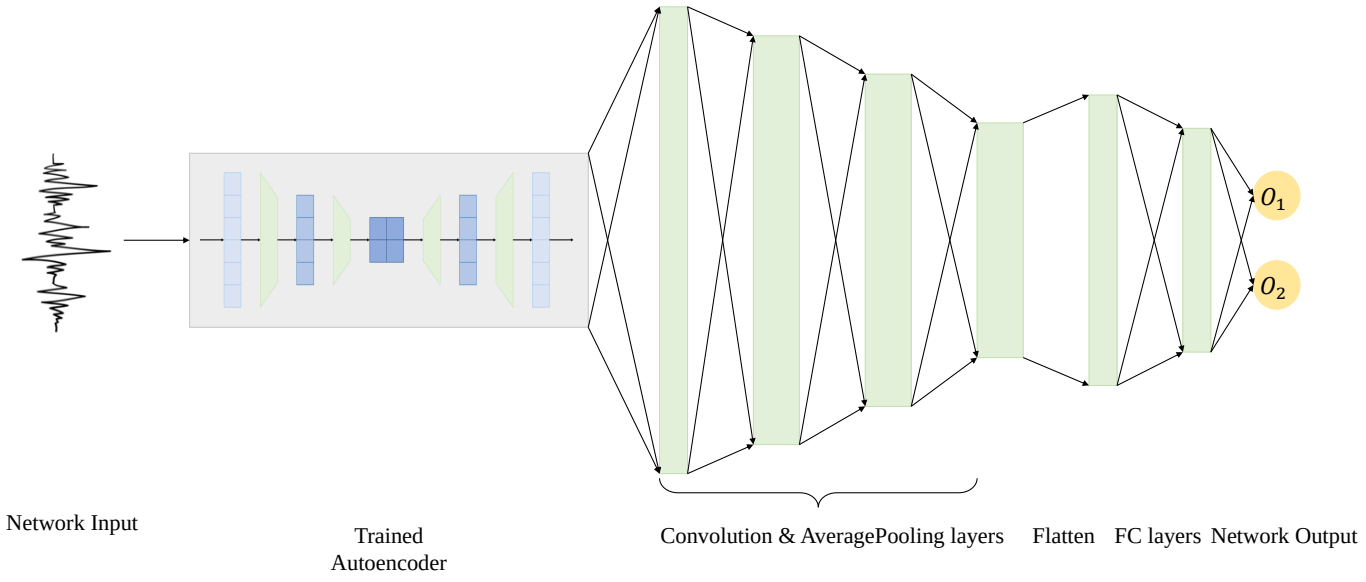

Figure 6. The architecture of CNN-HSSCA.

It can be seen from Figure 6 that the overall structure of CNN-HSSCA is the same as that of MLP-HSSCA, consisting of an autoencoder and a classifier network. Each power trace is preprocessed by the convolutional self-supervised autoencoder and used as the training data of the classifier network.

Table 2 below demonstrates the hyperparameters of CNN-HSSCA.

Table 2. Hyperparameters of CNN-HSSCA.

Parameter	CNN Autoencoder	CNN Classifier
Hidden layer	3 $4 \times 17 \times 1$ filters $8 \times 17 \times 1$ filters $4 \times 17 \times 1$ filters	4 $4 \times 32 \times 1$ filters $4 \times 16 \times 1$ filters $4 \times 8 \times 1$ filters 18 FC
Output size	700/350/2000	2 (SoftMax)
Label	Plaintext/Binary/Realign	Binary
Initialization	Xavier Normal	Xavier Normal
Optimization	Adam	Adam
Learning rate	0.001	0.001
Batch size	1000	1000

3.4. HSSCA for Small-Scale Datasets

The size of the dataset is a critical factor in the training of a network. Sufficient training data allows the network to learn and extract relevant features from the input data. When the dataset is limited in size, the impact of noise becomes more significant. This is because the influence of noise is less likely to be compensated for when the number of samples is small. However, if the training data possesses strong features and contains minimal noise, the network can still achieve promising results even with a smaller dataset. A self-supervised autoencoder in HSSCA can diminish the noise and strengthen the features of training data. With the application of HSSCA, the original data is processed to be more trainable with a small-scale dataset.

To reduce the noise in the training data, there are two main labeling schemes to train the autoencoder. The first one, referred to as plaintext labeling, is to use the average value of traces with the same plaintext. In scenarios where the key is fixed, the intermediate value calculated with Formula (2) is determined once the plaintext P is known. Since the intermediate value D exhibits a strong association with the data in the power traces, it is reasonable to assume that the sensitive data in traces with the same plaintext should be identical. To implement this labeling scheme, the traces are classified into 256 categories based on their plaintext values (for an 8-bit key). The average value of each category is then calculated, resulting in the ideal data serving as the label for each trace in that category. With this method, the autoencoder only needs to be trained once and can be used by every hypothetical key. This is because the traces with the same plaintext are consistently in the same category, regardless of the actual key value.

$$\{L_i\}_{1 \leqslant i \leqslant N} = \frac{\sum \{T_{i,p}\}_{1 \leqslant i \leqslant N_p}}{N_p} \tag{11}$$

In Formula (11), $\{L_i\}_{1 \leqslant i \leqslant N}$ represents the labels calculated according to the plaintext labeling scheme, $\{T_{i,p}\}_{1 \leqslant i \leqslant N_p}$ represents all power traces with the same plaintext p, and N_p represents the number of traces for each p.

The second labeling scheme named binary labeling uses the average value of traces with the same Least Significant Bit (LSB) or Most Significant Bit (MSB) in the intermediate value. In practical trace classification, the intermediate value is usually not directly used as the label, but the LSB or MSB of it. Therefore, classifying the traces by this method is more straightforward. However, the MSB or LSB of the intermediate value changes with the value of key. Under this circumstance, the parameters of the autoencoder have to be reinitialized and trained for every hypothetical key, adding complexity to the structure.

$$\{L_i\}_{1 \leqslant i \leqslant N} = \frac{\sum \{T_{i,b}\}_{1 \leqslant i \leqslant N_b}}{N_b} \tag{12}$$

In Formula (12), $\{T_{i,b}\}_{1 \leqslant i \leqslant N_b}$ represents all power traces with the same MSB or LSB denoted as b, and N_b represents the number of traces for each b.

3.5. HSSCA for Datasets with Desynchronized Traces

One of the most commonly implemented countermeasures to protect the encryption device is adding random delay to the encryption program. This technique introduces variations in the execution time of the encryption process, resulting in misaligned power traces during side-channel attacks. As a result, extracting leakage information of the intermediate value becomes challenging. Although structures like CNN-DDLA can study features in different positions, the computational cost would be higher because they require a larger amount of training data and more parameters in the model. By applying an self-supervised autoencoder for data preprocessing, it becomes possible to realign the data to make it easier to study instead of using a complex model and more training data to extract the desynchronized features. Algorithm 4 describes a new labeling procedure exploiting the statistical knowledge of power traces utilized for data resynchronization.

In Algorithm 4, **P** represents all possible values of plaintext, which are integers from 0 to 255 in the case of AES-128. By choosing the trace in $\{T_{i,p}\}_{1\leqslant i\leqslant N_p}$ with the highest average Pearson Correlation Coefficient, the trace considered at the center of the desynchronized traces is selected as label. Without considering the noise, traces in $\{T_{i,p}\}_{1\leqslant i\leqslant N_p}$ should be the same data sequence with different level of shifting. Therefore, the function of the autocoder with the new labels should be shifting the traces toward the label to realign them. To exploit the property of translation invariance, the labeling procedure in Algorithm 4 is implemented on the proposed CNN-HSSCA structure.

Algorithm 4 Labeling Procedure of desynchronized traces

Input: N power traces $\{T_i\}_{1\leqslant i\leqslant N}$ of length L with corresponding plaintexts $\{P_i\}_{1\leqslant i\leqslant N}$

Output: Reference traces as labels of autoencoder $\{L_i\}_{1\leqslant i\leqslant N}$

1: **for** $p \in$ **P do**
2: Group the traces with the same plaintext p into $\{T_{i,p}\}_{1\leqslant i\leqslant N_p}$
3: **for** $1 \leqslant i \leqslant N_p$ **do**
4: Calculate average value of the Pearson Correlation Coefficient between $T_{i,p}$ and every other trace in $\{T_{i,p}\}_{1\leqslant i\leqslant N_p}$ denoted as $\overline{C}_{i,p}$
5: **end for**
6: Choose $T_{m,p}$ where $\overline{C}_{m,p} = max(\{\overline{C}_{i,p}\}_{1\leqslant i\leqslant N_p})$ as the label $\{L_{i,p}\}_{1\leqslant i\leqslant N_p}$ for every trace in $\{T_{i,p}\}_{1\leqslant i\leqslant N_p}$
7: Group labels $\{L_{i,p}\}_{1\leqslant i\leqslant N_p}$ with the same plaintext into $\{L_i\}_{1\leqslant i\leqslant N}$
8: **end for**
9: **return** Reference traces $\{L_i\}_{1\leqslant i\leqslant N}$

4. Experiment Results

In this part of the paper, the performance of the proposed novel architecture is validated. In the first subsection, the construction of experiment environment and information of datasets used in the experiments are introduced. In the second subsection, the performance of HSSCA for feature enhancement is verified. Both MLP and CNN realization of the architecture are evaluated. The results are compared with the traditional DDLA architecture on both complete and small-scale datasets, including the dataset with fewer traces and dataset with downsampled traces. In the third subsection, CNN-HSSCA is proved to outperform the traditional DDLA on small-scale datasets with desynchronized data.

4.1. Experiment Environment and Datasets

4.1.1. Experiment Environment

The experimental neural networks are implemented using PyTorch version 1.12.1. The computations are performed using CUDA version 11.6. The experiments are conducted on a personal computer equipped with an NVIDIA GeForce RTX 2070 SUPER GPU, an Intel i5-4590 CPU, and 16 GB RAM.

In all experiments, for both the proposed model and the traditional DDLA model, the Adam optimizer is selected with a learning rate of 0.001, the Mean Square Error (MSE) with the regularization factor of 10^{-8} is chosen as the loss function, and the batch size is determined as 1000. All input data are preprocessed by removing its average value and scaling it in the range of $[-1, 1]$ before training.

4.1.2. ASCAD Dataset

ASCAD is a publicly available database that contains power traces and corresponding information designed for deep-learning-based side-channel analysis [37]. The power traces included in the ASCAD dataset are obtained from a first-order protected software implementation of the AES algorithm running on an 8-bit AVR ATMega8515 development board. The power traces are captured using an electromagnetic sensor operating at a frequency of 2 GHz. To ensure the quality and relevance of the data, an SNR analysis is performed. This analysis helps identify the 700 sampling points that contain the relevant information about the SubByte step of the third byte in the first round of AES encryption. The selected power traces are divided into two parts: 50,000 profiling traces and 10,000 attacking traces.

4.1.3. Desynchronized ASCAD Dataset

On the basis of the original ASCAD dataset, a parametrized desynchronization is employed on the power traces to imitate the jitter of devices. The desynchronization of the traces are simulated by generating for each trace a random number δ in $[0 \ldots N_{max}]$ and by shifting the original trace of δ points to the left, where N_{max} is 50 in the experiment.

4.1.4. AES_RD Dataset

AES_RD Dataset is introduced by Coron and Kizhvatov [38]. The power traces in the AES_RD dataset are obtained from a software AES implementation executing on an 8-bit Atmel AVR microcontroller with a countermeasure of random delay. Each delay is a multiple of 3 processor cycles. In the experiments, the attacks are conducted against the first byte of the key, targeting the first SubByte operation. The dataset consists of 50,000 traces of 3500 features each. The information of datasets is demostrated in Table 3.

Table 3. Summary of datasets in the experiments.

Dataset	Traces	Samples	Protection
ASCAD	60,000	700	First-order Boolean Mask
ASCAD_desync	60,000	700	First-order Boolean Mask & Random Delay
AES_RD	50,000	3500	Random Delay

4.2. Implementation Results of HSSCA on Feature Enhancement

To primarily test the efficiency of HSSCA in finding the correct key, 10,000 attacking traces in the ASCAD dataset are used as training data. The labeling scheme in Formula (11) is selected because it is effective in recovering the correct key when there is sufficient data. The experiment also includes the performance of DDLA with binary labeling under the same conditions for comparison. Instead of using the rank of each key candidate as the evaluation metric, which is the standard procedure of profiled analysis, the estimated key is selected based on the loss and accuracy curve of each key for deep-learning-based non-profiled analysis.

It is shown in Figure 7 that when there is sufficient data, the two structures of HSSCA and DDLA all recover the correct key successfully. In all graphs, the blue curves that represent the correct key are separated from the gray curves that represent the wrong keys, which indicates that all attacks are successful. Notably, the result of HSSCA presents the correct key with more distinction. Additionally, MLP-HSSCA is able to distinguish the correct key within only five epochs of training, while DDLA requires more epochs to

identify the most distinct curve. This advantage of HSSCA over the traditional DDLA indicates a reduction in computational complexity.

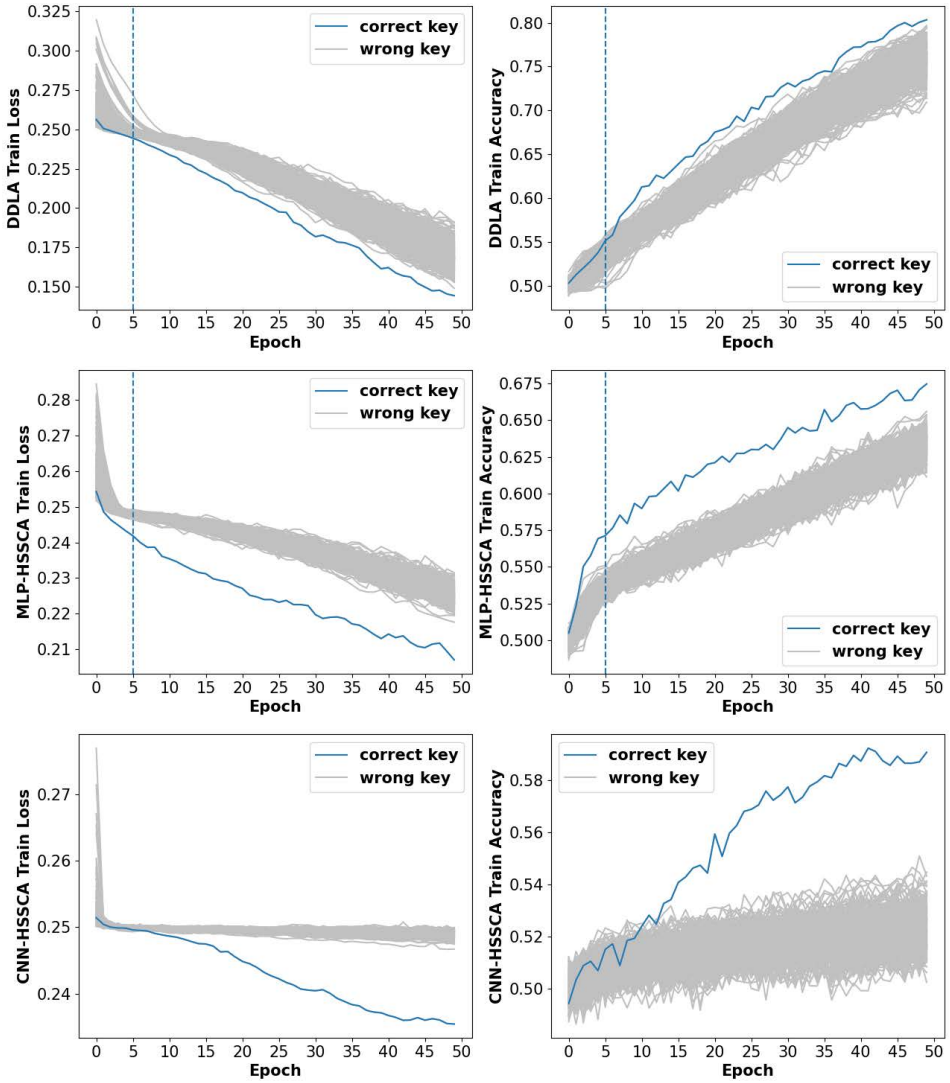

Figure 7. Loss and accuracy of DDLA and HSSCA with 10,000 training data over the training epochs. **Upper**: DDLA. **Center**: MLP-HSSCA. **Lower**: CNN-HSSCA.

In order to evaluate the capability of HSSCA in distinguishing the correct key with a smaller amount of data, the first 2000 traces from the attacking traces in the ASCAD dataset are utilized as training data. Both labeling schemes described earlier are implemented on HSSCA to assess their effectiveness under this extreme condition. The traditional DDLA is put through the experiment under the same condition as the comparison. Figure 8 presents the experiment results of HSSCA and DDLA when the number of training data is 2000.

As is demonstrated in Figure 8, when the number of training data is reduced to 2000, the DDLA network fails to distinguish the correct key. For the experiment results of HSSCA, although the distance between the correct curves and wrong curves becomes shorter, they can still be distinguished as the correct key, especially for CNN-HSSCA, whose performance remains stable as in the previous experiment. It is noted that the binary labeling scheme yields better results than the former in the early epochs. This observation aligns with the theoretical understanding that the self-supervised autoencoder employed in HSSCA is effective in identifying the Points of Interest (PoI) in a power trace and reducing noise in the signal.

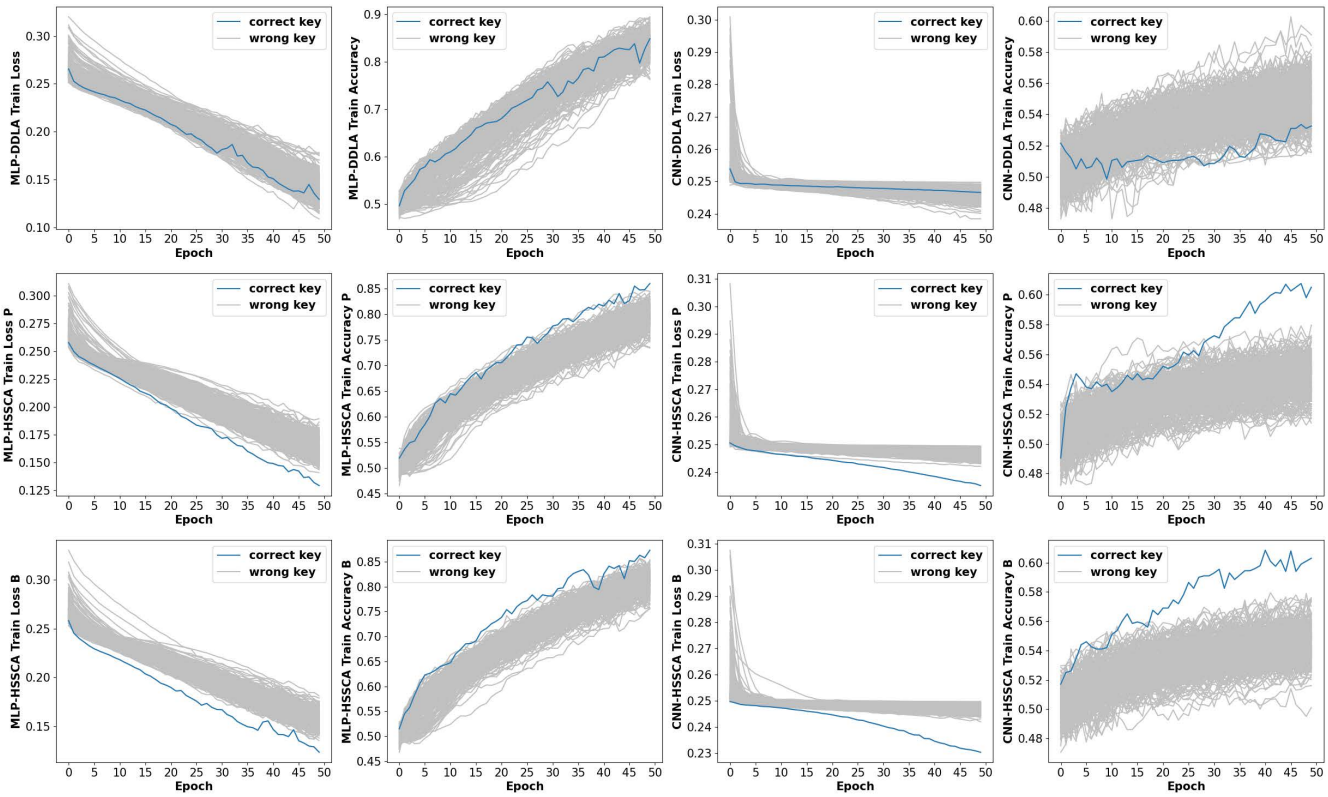

Figure 8. Loss and accuracy of DDLA and HSSCA with 2000 training data over the training epochs. **Upper Left**: MLP-DDLA. **Center Left**: MLP-HSSCA with plaintext labeling. **Lower Left**: MLP-HSSCA with binary labeling. **Upper Right**: CNN-DDLA. **Center Right**: CNN-HSSCA with plaintext labeling. **Lower Right**: CNN-HSSCA with binary labeling.

Table 4 lists the number of traces and epochs required to reveal the correct key in some previous models and the proposed model. It is demonstrated in the table that although all models can distinguish the correct key, the amount of data needed to train HSSCA in both architectures is significantly less than related works on attacking the ASCAD dataset.

Table 4. A review of related works of deep-learning side-channel attacks on ASCAD dataset.

Reference	Network Architecture	Number of Traces	Epochs
[26]	MLP	30,000	100
[16]	MLP	20,000	50
[19]	MLP/CNN	20,000	50
[39]	CNN	20,000	50
[24]	profiled CNN	60,000	-
[40]	profiled MLP/CNN	10,000	-
[28]	MLP/BNN	10,000	100
This work	MLP-HSSCA	2000	50
This work	CNN-HSSCA	2000	50

To push the performance of CNN-HSSCA to the limit, an experiment where downsampled power traces are used as input data is conducted. The experiment has practical meaning because in the process of collecting power leakage data from encryption devices, a digital oscilloscope with extremely high sampling rate and accuracy is required to gather sufficient data in a very short period of time. However, this kind of equipment is unreachable for average researchers. Under this circumstance, it would be significant to design a method of analysis that is able to reveal the correct key with data collected by devices with lower sampling rate. The downsampled dataset is utilized to imitate such case. In the

following experiment, only plaintext labeling is implemented on CNN-HSSCA because the structure is simpler and the results are similar with that of binary labeling. The details of the performance of the two labeling schemes are introduced in Section 4.4.

It can be observed from Figure 9 that the novel architecture of CNN-HSSCA maintains the ability of revealing the correct key when the function of CNN-DDLA has been compromised. The 700 points in a power trace of ASCAD dataset are downsampled to 350, simulating a scenario where the sampling rate of the device is reduced to half of the original rate. As for CNN-DDLA, it is constructed with the same layers as the classifier network in CNN-HSSCA. To be noted, parameters of the network such as kernel size of convolutional layers are all reduced to half to match the size of input data. It indicates that the novel architecture can capture the sensitive data to associate it with key information even if the data are sampled at a lower rate.

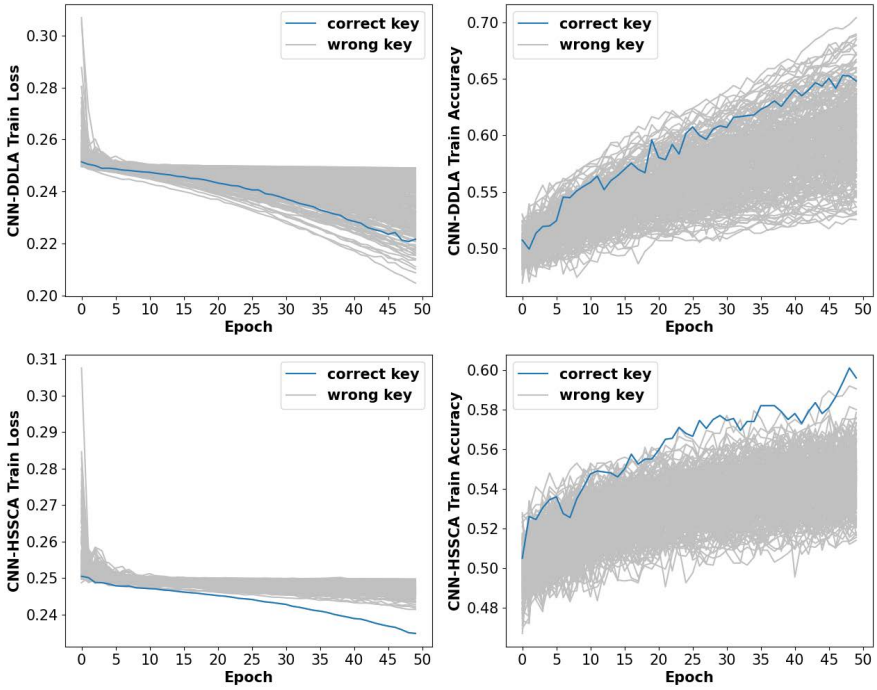

Figure 9. Loss and accuracy of DDLA and HSSCA with 2000 downsampled training data over the training epochs. **Upper**: CNN-DDLA. **Lower**: CNN-HSSCA with plaintext labeling.

4.3. Implement Results of HSSCA on Desynchronized Datasets

In order to evaluate the efficiency of HSSCA in exploiting the leakage from desynchronized datasets, two datasets are chosen as the input data of the proposed novel architecture with the labeling procedure in Algorithm 4. As the comparison, the two datasets are also used to train the CNN-DDLA, which is expected to be capable of distinguishing the correct key from desynchronized traces. Moreover, because the function of feature enhancement of the self-supervised autoencoder still exists in the new labeling procedure, the experiments are conducted with fewer data than in related works.

For desynchronized ASCAD dataset, all 700 shifted data are collected in the process of attacking the third byte of the key. However, for AES_RD dataset, the 3500 points contains irrelevant information about other bytes of the key, making it difficult to train the network. Therefore, a larger number of traces are needed in the analysis against the AES_RD dataset. The feature selection process such as principal component analysis (PCA) is difficult to conduct because of the desynchronization countermeasure; thus, a simple solution is employed to choose the first 2000 points, as the first byte is the target.

From Figures 10 and 11, it can be concluded that while CNN-DDLA is capable of extracting leakage from desynchronized datasets when there is a sufficient amount of data, it fails to distinguish the correct key when the input data is both desynchronized and

limited. For instance, on the upper left of Figure 11, an incorrect key is distinguished, indicating that the analysis has failed. On the other hand, although it takes more epochs for desynchronized ASCAD, CNN-HSSCA still succeeds in revealing the correct key in both datasets, and the results resembles the results on the synchronized datasets. It proves that the self-supervised autoencoder can realign the features in data for the classifier.

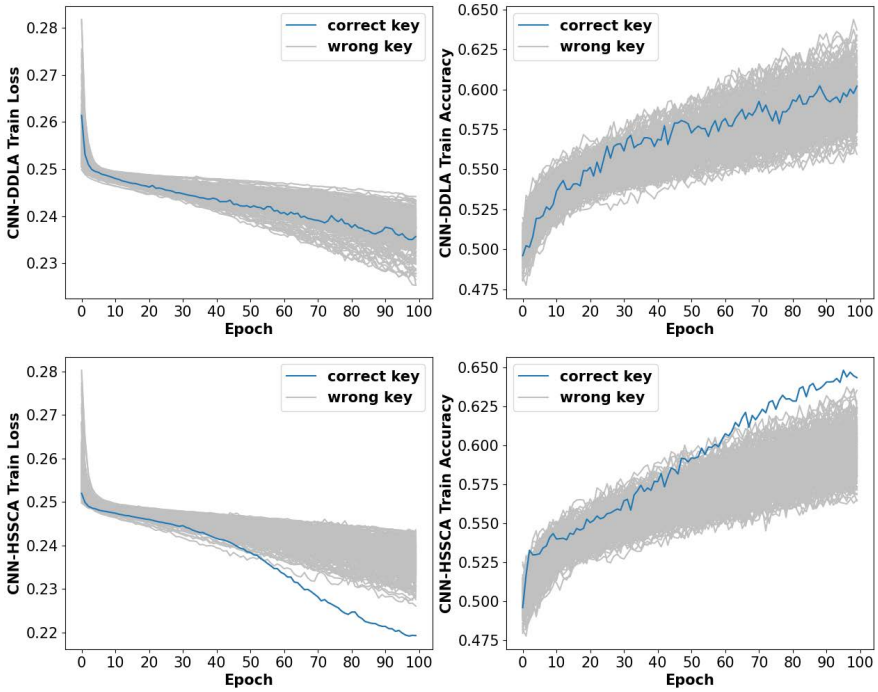

Figure 10. Loss and accuracy of DDLA and HSSCA with 4000 desynchronized training data in ASCAD over the training epochs. **Upper**: CNN-DDLA. **Lower**: CNN-HSSCA with labeling procedure in Algorithm 4.

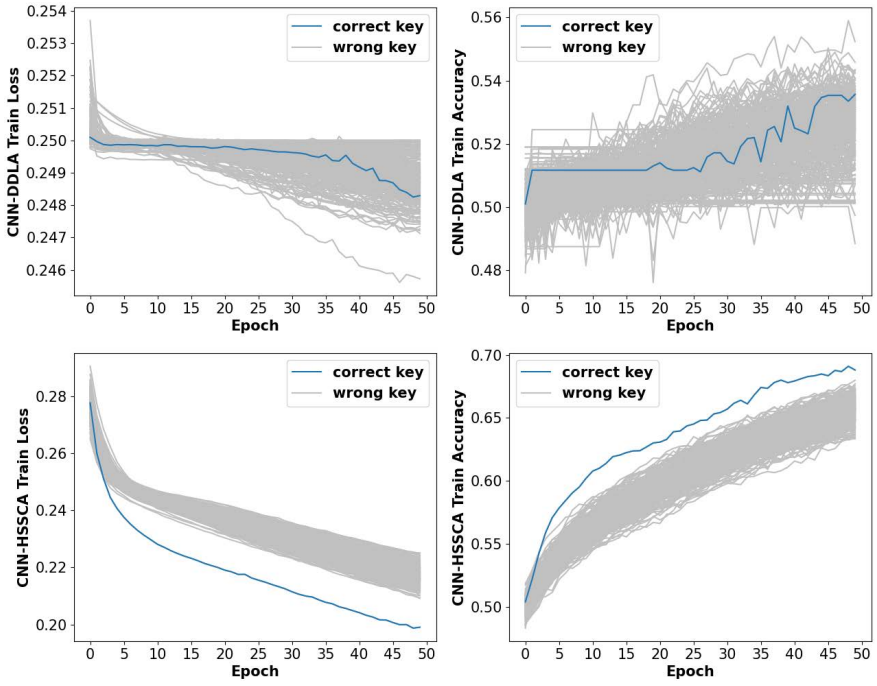

Figure 11. Loss and accuracy of DDLA and HSSCA with 6000 desynchronized training data in AES_RD over the training epochs. **Upper**: CNN-DDLA. **Lower**: CNN-HSSCA with labeling procedure in Algorithm 4.

4.4. Reliability and Computational Efficiency Analysis of HSSCA

To evaluate the reliability of the proposed model, 100 independent replicate experiments are conducted on all of the experiments above, and the success rates of side-channel analysis of each experiments are demonstrated in Figure 12.

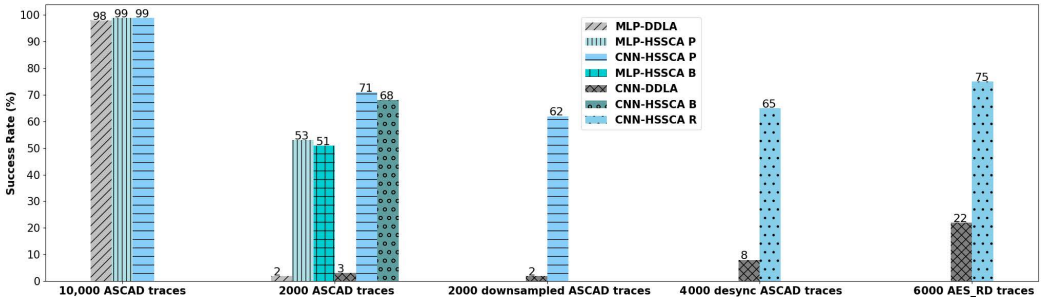

Figure 12. Success rate of DDLA and HSSCA in the experiments.

From the results shown in Figure 12, it can be observed that both the traditional model and the proposed model possess high success rate when the size of dataset is large. However, the traditional model fails to distinguish the correct key almost every time when the number of traces is reduced to 2000. On the contrary, the success rates of the proposed model remains acceptable under the same condition. The situation is similar when the data are downsampled or desynchronized. Moreover, it is concluded in the experiments that when an attack with the proposed model fails, all the curves are mixed so there is not a distinguished incorrect key that could be mistakenly considered as the correct key by the attacker. Under this circumstance, if an analysis with the proposed model fails, the attacker simply has to perform the analysis again and again until a curve is distinguished.

To evaluate the computational efficiency of the proposed model, the average execution time of one side-channel analysis of the traditional DDLA and the proposed HSSCA in the experiments is recorded and illustrated in Figure 13.

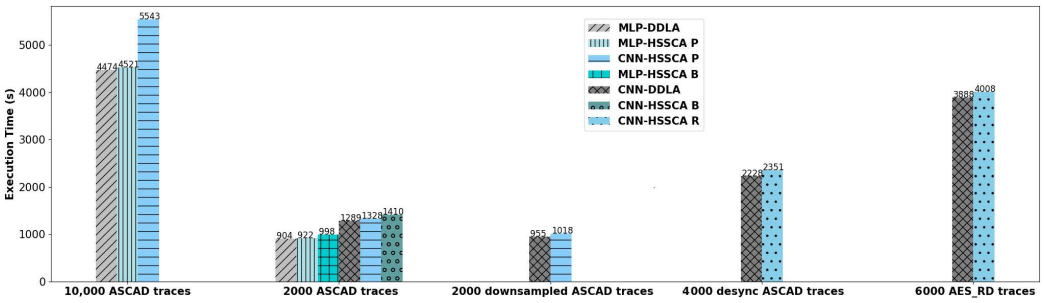

Figure 13. Execution time of DDLA and HSSCA in the experiments.

From Figure 13, it is obvious that the average execution time of HSSCA exceeds that of DDLA because HSSCA consists of two training processes: the training process of autoencoder and classifier. The CNN-HSSCA model is more time-consuming than MLP-HSSCA and DDLA under the same condition because there are more parameters to train. The HSSCA with binary labeling scheme consumes more time because the autoencoder needs to be trained for every hypothetical key. Despite that, the increased time is acceptable because it does not increase dramatically under the same condition and the real-time demand for a side-channel analysis is usually not high in practice.

5. Conclusions

In this paper, we introduce a novel architecture for non-profiled differential side-channel attacks with a hybrid-supervised neural network aiming to adapt the method of DDLA for practical applications. The key feature of this architecture is its ability to reduce power trace noise and enhance critical information using the self-supervised autoencoder.

Experimental results demonstrate that the proposed architecture, utilizing both MLP and CNN implementations, outperforms traditional DDLA in terms of distinction and computational efficiency. This aligns with the notion that the architecture effectively enhances data features. By training data with enhanced features, the architecture maintains the capability to accurately distinguish the correct key even with significantly fewer data. Additional experiments highlight the robustness of the architecture, showing that it can recover the correct key in extreme conditions in practice. For instance, when the amount of data is one-fifth of the amount in general situations or when the data are downsampled to half, the novel architecture still can reveal the correct key. In contrast, DDLA fails to function effectively under such conditions. Additionally, the versatility of the novel architecture is demonstrated by its successful recovery of the correct key from desynchronized traces. An altered version of the architecture specifically designed for data resynchronization proves to be effective through experiments on two datasets in this regard. The reliability and computational efficiency of the proposed model are demonstrated through statistical analysis on the success rate and execution time of it. The results illustrate that the proposed model reaches significantly higher success rate than the traditional DDLA model with a reasonable cost of average execution time.

Author Contributions: Conceptualization, F.K. and K.P.; methodology, F.K., X.W. and K.P.; software, J.Z.; validation, F.K. and K.P.; formal analysis, F.K.; investigation, F.K. and K.P.; resources, H.D.; writing—original draft preparation, F.K. and K.P.; writing—review and editing, J.Z.; supervision, X.W. and H.D.; project administration, H.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All data can be provided upon reasonable request to the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this paper:

CPA	Correlation Power Analysis
DPA	Differential Power Analysis
DDLA	Differential Deep Learning Analysis
AES	Advanced Encryption Standard
SNR	Signal-to-Noise Ratio
HSSCA	Hybrid-Supervised Side-Channel Analysis
MLP	Multilayer Perceptron
CNN	Convolutional Neural Network
DL	Deep Learning
ML	Machine Learning
AI	Artificial Intelligence
FC	Fully Connected
ReLU	Rectified Linear Unit
LSB	Least Significant Bit
RReLU	Randomized leaky Rectified Linear Unit
MSB	Most Significant Bit
MSE	Mean Square Error
PoI	Points of Interest
PCA	Principal Component Analysis

References

1. Kocher, P.C. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Proceedings of the Advances in Cryptology—CRYPTO'96: 16th Annual International Cryptology Conference, Santa Barbara, CA, USA, 18–22 August 1996; Springer: Berlin/Heidelberg, Germany, 1996; pp. 104–113.
2. Le, T.H.; Canovas, C.; Clédiere, J. An overview of side channel analysis attacks. In Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security, Tokyo, Japan, 18–20 March 2008; pp. 33–43.
3. Standaert, F.X.; Koeune, F.; Schindler, W. How to compare profiled side-channel attacks? In Proceedings of the Applied Cryptography and Network Security: 7th International Conference, ACNS 2009, Paris-Rocquencourt, Paris, France, 2–5 June 2009; Springer: Berlin/Heidelberg, Germany, 2009; pp. 485–498.
4. Chari, S.; Rao, J.R.; Rohatgi, P. Template attacks. In Proceedings of the Cryptographic Hardware and Embedded Systems-CHES 2002: 4th International Workshop, Redwood Shores, CA, USA, 13–15 August 2002; Revised Papers; Springer: Berlin/Heidelberg, Germany, 2003; pp. 13–28.
5. Rechberger, C.; Oswald, E. Practical template attacks. In *International Workshop on Information Security Applications*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 440–456.
6. Hettwer, B.; Gehrer, S.; Güneysu, T. Applications of machine learning techniques in side-channel attacks: A survey. *J. Cryptogr. Eng.* **2020**, *10*, 135–162. [CrossRef]
7. Schindler, W.; Lemke, K.; Paar, C. A stochastic model for differential side channel cryptanalysis. In Proceedings of the Cryptographic Hardware and Embedded Systems–CHES 2005: 7th International Workshop, Edinburgh, UK, 29 August–1 September 2005; Springer: Berlin/Heidelberg, Germany, 2005; pp. 30–46.
8. Lemke-Rust, K.; Paar, C. Analyzing side channel leakage of masked implementations with stochastic methods. In Proceedings of the Computer Security–ESORICS 2007: 12th European Symposium on Research in Computer Security, Dresden, Germany, 24–26 September 2007; Springer: Berlin/Heidelberg, Germany, 2007; pp. 454–468.
9. Brier, E.; Clavier, C.; Olivier, F. Correlation power analysis with a leakage model. In Proceedings of the Cryptographic Hardware and Embedded Systems—CHES 2004: 6th International Workshop, Cambridge, MA, USA, 11–13 August 2004; Springer: Berlin/Heidelberg, Germany, 2004; pp. 16–29.
10. Sakamoto, J.; Tachibana, K.; Matsumoto, T. Application of Profiled Analysis to ADC-Based Remote Side-Channel Attacks. In Proceedings of the 2023 IEEE 9th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), New York, NY, USA, 6–8 May 2023; pp. 115–121.
11. Kocher, P.; Jaffe, J.; Jun, B. Differential power analysis. In Proceedings of the Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference, Santa Barbara, CA, USA, 15–19 August 1999; Springer: Berlin/Heidelberg, Germany, 1999; pp. 388–397.
12. Maghrebi, H.; Portigliatti, T.; Prouff, E. Breaking cryptographic implementations using deep learning techniques. In Proceedings of the Security, Privacy, and Applied Cryptography Engineering: 6th International Conference, SPACE 2016, Hyderabad, India, 14–18 December 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 3–26.
13. Wouters, L.; Arribas, V.; Gierlichs, B.; Preneel, B. Revisiting a methodology for efficient CNN architectures in profiling attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2020**, *2020*, 147–168. [CrossRef]
14. Lu, X.; Zhang, C.; Cao, P.; Gu, D.; Lu, H. Pay attention to raw traces: A deep learning architecture for end-to-end profiling attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2021**, *2021*, 235–274. [CrossRef]
15. Pfeifer, C.; Haddad, P. Spread: A new layer for profiled deep-learning side-channel attacks. *Cryptol. Eprint Arch.* **2018**, *2018*.
16. Timon, B. Non-profiled deep learning-based side-channel attacks with sensitivity analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2019**, *2019*, 107–131. [CrossRef]
17. Le, T.H.; Clédière, J.; Serviere, C.; Lacoume, J.L. Noise reduction in side channel attack using fourth-order cumulant. *IEEE Trans. Inf. Forensics Secur.* **2007**, *2*, 710–720. [CrossRef]
18. Das, D.; Maity, S.; Nasir, S.B.; Ghosh, S.; Raychowdhury, A.; Sen, S. High efficiency power side-channel attack immunity using noise injection in attenuated signature domain. In Proceedings of the 2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), Mclean, VA, USA, 1–5 May 2017; pp. 62–67.
19. Do, N.T.; Hoang, V.P.; Doan, V.S.; Pham, C.K. On the performance of non-profiled side channel attacks based on deep learning techniques . *IET Inf. Secur.* **2023**, *17*, 377–393. [CrossRef]
20. Cheng, K.; Song, Z.; Cui, X.; Zhang, L. Hybrid denoising based correlation power analysis for AES. In Proceedings of the 2021 IEEE 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), Chongqing, China, 18–20 June 2021; Volume 4, pp. 1192–1195.
21. Tseng, C.C.; Lee, S.L. Minimax design of graph filter using Chebyshev polynomial approximation. *IEEE Trans. Circuits Syst. II Express Briefs* **2021**, *68*, 1630–1634. [CrossRef]
22. Maghrebi, H.; Prouff, E. On the use of independent component analysis to denoise side-channel measurements. In Proceedings of the Constructive Side-Channel Analysis and Secure Design: 9th International Workshop, COSADE 2018, Singapore, 23–24 April 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 61–81.
23. Kwon, D.; Kim, H.; Hong, S. Non-profiled deep learning-based side-channel preprocessing with autoencoders. *IEEE Access* **2021**, *9*, 57692–57703. [CrossRef]

24. Paguada, S.; Batina, L.; Armendariz, I. Toward practical autoencoder-based side-channel analysis evaluations. *Comput. Netw.* **2021**, *196*, 108230. [CrossRef]
25. Wu, L.; Picek, S. Remove some noise: On pre-processing of side-channel measurements with autoencoders. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2020**, *2020*, 389–415. [CrossRef]
26. Kuroda, K.; Fukuda, Y.; Yoshida, K.; Fujino, T. Practical aspects on non-profiled deep-learning side-channel attacks against AES software implementation with two types of masking countermeasures including RSM. In Proceedings of the 5th Workshop on Attacks and Solutions in Hardware Security, Virtual Event, Republic of Korea, 19 November 2021; pp. 29–40.
27. Kwon, D.; Hong, S.; Kim, H. Optimizing implementations of non-profiled deep learning-based side-channel attacks. *IEEE Access* **2022**, *10*, 5957–5967. [CrossRef]
28. Won, Y.S.; Han, D.G.; Jap, D.; Bhasin, S.; Park, J.Y. Non-profiled side-channel attack based on deep learning using picture trace. *IEEE Access* **2021**, *9*, 22480–22492. [CrossRef]
29. Hu, F.; Wang, H.; Wang, J. Cross subkey side channel analysis based on small samples. *Sci. Rep.* **2022**, *12*, 6254. [CrossRef]
30. Taud, H.; Mas, J. Multilayer perceptron (MLP). In *Geomatic Approaches for Modeling Land Change Scenarios*; Springer International: Cham, Swizerland, 2018; pp. 451–455.
31. O'Shea, K.; Nash, R. An introduction to convolutional neural networks. *arXiv* **2015**, arXiv:1511.08458.
32. Eren, L.; Ince, T.; Kiranyaz, S. A generic intelligent bearing fault diagnosis system using compact adaptive 1D CNN classifier. *J. Signal Process. Syst.* **2019**, *91*, 179–189. [CrossRef]
33. Daemen, J.; Rijmen, V. Reijndael: The advanced encryption standard. *Dr. Dobb's J. Softw. Tools Prof. Program.* **2001**, *26*, 137–139.
34. Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y.; Manzagol, P.A.; Bottou, L. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.* **2010**, *11*, 3371–3408.
35. Vincent, P.; Larochelle, H.; Bengio, Y.; Manzagol, P.A. Extracting and composing robust features with denoising autoencoders. In Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, 5–9 July 2008; pp. 1096–1103.
36. Liu, X.; Zhang, F.; Hou, Z.; Mian, L.; Wang, Z.; Zhang, J.; Tang, J. Self-supervised learning: Generative or contrastive. *IEEE Trans. Knowl. Data Eng.* **2021**, *35*, 857–876. [CrossRef]
37. Benadjila, R.; Prouff, E.; Strullu, R.; Cagli, E.; Dumas, C. Deep learning for side-channel analysis and introduction to ASCAD database. *J. Cryptogr. Eng.* **2020**, *10*, 163–188. [CrossRef]
38. Coron, J.S.; Kizhvatov, I. An efficient method for random delay generation in embedded software. In Proceedings of the Cryptographic Hardware and Embedded Systems-CHES 2009: 11th International Workshop, Lausanne, Switzerland, 6–9 September 2009; Springer: Berlin/Heidelberg, Germany, 2009; pp. 156–170.
39. Do, N.T.; Hoang, V.P.; Doan, V.S. A novel non-profiled side channel attack based on multi-output regression neural network. *J. Cryptogr. Eng.* **2023**, 1–13. [CrossRef]
40. Nomikos, K.; Papadimitriou, A.; Psarakis, M.; Pikrakis, A.; Beroulle, V. Evaluation of Hiding-based Countermeasures against Deep Learning Side Channel Attacks with Pre-trained Networks. In Proceedings of the 2022 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), Austin, TX, USA, 19–21 October 2022; pp. 1–6.