# DLPFA: Deep Learning based Persistent Fault Analysis against Block Ciphers

Yukun Cheng[1], Changhai Ou[1], Fan Zhang[2] and Shihui Zheng[3]

[1] School of Cyber Science & Engineering, Wuhan University, Wuhan, China, ouchanghai@whu.edu.cn

[2] College of Computer Science and Technology, Zhejiang University, China, fanzhang@zju.edu.cn

[3] School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing, China, shihuizh@bupt.edu.cn

**Abstract.** Deep learning techniques have been widely applied to side-channel analysis (SCA) in recent years and shown better performance compared with traditional methods. However, there has been little research dealing with deep learning techniques in fault analysis to date. This article undertakes the first study to introduce deep learning techniques into fault analysis to perform key recovery. We investigate the application of multi-layer perceptron (MLP) and convolutional neural network (CNN) in persistent fault analysis (PFA) and propose deep learning-based persistent fault analysis (DLPFA). DLPFA is first applied to advanced encryption standard (AES) to verify its availability. Then, to push the study further, we extend DLPFA to PRESENT, which is a lightweight substitution–permutation network (SPN)-based block cipher. The experimental results show that DLPFA can handle random faults and provide outstanding performance with a suitable selection of hyper-parameters.

**Keywords:** Deep Learning · Fault Analysis · PFA · AES · PRESENT

## 1 Introduction

### 1.1 Overview

To provide a reliable and efficient computing environment, cryptographic algorithms are implemented in particular cryptographic devices. Security in such devices relies not only on the mathematical security of the cryptographic algorithm, but also on the physical security of the cryptographic implementation. Nowadays, implementation-based attacks have attracted intensive attentions. Such attacks pose a serious threat to cryptographic security, as they can exploit leakage of physical information that is neglected in conventional cryptanalysis.

Depending on the leakage of physical information, implementation-based attacks can be classified into passive and active attacks. Side-channel analysis (SCA) is the representative of passive attacks. The adversary does not interfere with the work of the cryptographic device. The adversary plays the role of an observer and collects key-related information from the power consumption[KJJ99], electromagnetic emanation[GMO01] and other leakages of physical information while the cryptographic device is in operation. There are kinds of SCA to recover the key using different methods, such as simple power analysis (SPA)[Man02], correlation power analysis (CPA)[BCO04] and template analysis (TA)[CRR02]. On the other hand, fault analysis (FA), which is the representative of active attacks, aims to induce cryptographic device faults using some physical methods, such as voltage glitch[BMM00], temperature[GA03] and laser[CLFT14]. The adversary analyzes the faulty outputs of the cryptographic device to recover the secret key. Differential fault analysis (DFA)[BS97] and statistical fault analysis (SFA)[FJLT13] are the most prominent fault analysis over the years. Additionally, statistical ineffective fault analysis (SIFA)[DEK+18] and persistent fault analysis (PFA)[ZLZ+18] also show their tremendous potential as new analysis methods in recent years.

In the traditional FA, the exploitation of the fault leakage depends solely on the analysis method. However, these methods are more concerned with how to successfully recover the key than how to fully exploit the fault leakage. To improve the exploitation of the fault leakage, deep learning techniques are ideal tools as neural networks can learn from the data automatically and fully. However, how to introduce deep learning techniques into FA has not been investigated yet.

In this article, we first apply deep learning techniques to fault analysis to perform key recovery. Given some feature information from the faulty cryptographic implementation, our ultimate goal is to recover the secret key that has been processed internally. We propose a novel method, named deep learning-based persistent fault analysis (DLPFA), to solve this complex problem using a divide-and-conquer approach.

## 1.2 Related Works

The application of neural networks in the field of cryptography was first published in [Riv91]. The combination of machine learning techniques and SCA was first proposed by Backes et al. [BDG+10], who recovered the printed text from a printer using acoustic side channel. Hospodar et al. proposed the first study on the application of machine learning techniques in SCA[HGM+11]. From here, a large number of works have been proposed to break both unprotected[BL12, LPB+15] and protected cryptographic implementations[GHO15, LBM15] using machine learning techniques.

In recent years, more researchers have focused on deep learning techniques, such as multilayer perceptron networks (MLP)[MZVT15, MDM16] and convolutional neural networks (CNN)[PSK+18] to go with the trend in the machine learning community. Of the various SCA, profiled attacks, represented by TA, are the most powerful ones and in some cases can recover the key with only one power trace. These attacks can be composed of two phases: profiling and matching, which are similar to the training phase and test phase in deep learning. On this basis, deep learning techniques have been introduced into SCA as an alternative to TA[KPH+19, CDP17a, BPS+20]. Compared to TA, deep learning-based SCA is more robust to noise and shows comparable or better performance[Tim19].

On the other hand, there is less research on the combination of machine learning or deep learning techniques with FA. Several researchers have introduced FA into deep neural networks for misclassification[LWLX17, BHJ+18]. Moreover, Saha et al. have applied deep learning techniques to the leakage assessment of FA[SAB+20]. However, to the best of our knowledge, the investigation of the application of machine learning or deep learning techniques in the key recovery of FA is a void in the field of cryptography.

## 1.3 Contributions

Our contributions in this article can be summarized as follows:

- To improve the exploitation of the fault leakage, we initiate the first comprehensive study of the application of deep learning based key recovery in the context of fault analysis and propose DLPFA as a novel FA method. The complex key-recovery problem is decomposed into three tractable sub-problems: fault model selection, feature selection and classification.

- We analyze the effect of faults and propose an effective feature selection method. The fault model of PFA is selected as the target model and the probability distributions of the faulty outputs of the S-box, which contains a large amount of relevant information about the key, is selected as the feature. Moreover, we prepare the dataset containing all the cases of faults in PFA fault model.

- We select MLP and CNN as neural network models and comb the attack process. In the training phase, a large dataset is fed to the network to teach it the relevant knowledge of the key. In the attack phase, another dataset is provided to the network to mount the actual attack.
- To reasonably evaluate the potency of our attack, we perform DLPFA on advanced encryption standard (AES), which is the classical substitution–permutation network (SPN)-based block cipher, and PRESENT, which is a lightweight SPN-based cipher. Moreover, we discuss several parametrization options in the experiments and present tuned hyper-parameters. The given hyper-parameters can be viewed as a proposal to help future researchers to make their own choice for the design of new deep learning models.

### 1.4 Organizations

The rest of the article is organized as follows: We begin with the selection of the fault model and features in Section 2. Section 3 introduces the neural network models used in this work and describes the whole process of DLPFA. The experimental results on two different ciphers are presented in Sections 4 and 5. We also discuss the impact of hyper-parameters on the results in these two sections. Finally, we conclude this article with a summary of our attack and an outlook for the future work in Section 6.

## 2 Fault Model and Feature Selection

In this section, we first give a brief introduction to the fault model of PFA selected in this article. We then describe the method of feature selection, which is a necessary preliminary step in deep learning. Finally, the dataset is given to assist in solving the classification problem.

### 2.1 Fault Model

For a systematic study of FA, faults are categorized by effects, such as altering the control flow, skipping commands or changing the storage information. The most typical one in each category is chosen as the representative to build the fault model. The fault model explains the position (e.g. before or after specific operation), the property (e.g. transient, permanent or persistent), the effect of the fault and the ability of the adversary.

In this article, we focus on the fault model of PFA in [ZLZ+18], which can be described as follows:

- The faults are injected into the cryptographic device before the encryption of the block cipher.
- The injected faults randomly corrupt the stored constants of the algorithm (i.e. the elements in the S-box) in the storage of the device.
- The injected faults are persistent, meaning that the faulty constants are used during the encryption until the storage is refreshed or the device is rebooted.
- The adversary is able to feed plaintexts into the faulty device and collect ciphertexts for subsequent fault analysis.

Such a choice of fault model has several advantages. In contrast to traditional fault models, which are mostly transient, the fault model of PFA places minor restrictions on the adversary's capabilities, such as his ability to inject faults within a relatively loose time window. Then, according to the persistence property, the fault can be exploited multiple times with only one injection. Also, note that the faults lead to a non-uniform distribution of the outputs of the S-box. This non-uniform distribution favors feature selection, as will be explained in detail below.

## 2.2 Feature Selection

Before solving the classification problem, feature selection is used as a preliminary step in deep learning, which filters out and preprocesses the given raw data. The significance of feature selection can be mainly represented in two aspects. On one hand, this step filters out erroneous features and thus improves the accuracy of the neural network. On the other hand, the features containing larger relevant pieces of information are extracted, which reduce the computational cost of the neural network.

In the traditional SCA, power traces carry a large amount of relevant information about the key, and are thus usually used to recover the key. Moreover, from the data format perspective, power traces can be treated like the two-dimensional image, which has been widely studied in the field of deep learning. Therefore, in the deep learning-based SCA, the adversary can directly select features from the power traces with some statistical methods, such as Pearson correlation coefficient or principal component analysis.

Unfortunately, in the field of FA, the problem of feature selection is more complicated. The effects of faults are various, leading to a variety of data used in different fault analysis methods. For example, DFA performs an analysis based on the differential values between the correct and faulty ciphertexts which are generated from the same plaintext. SFA takes advantage of the bias caused by the fault to recover the key. Moreover, the adversary of PFA pays extra attention to the distribution of ciphertexts after a large number of encryptions. It is difficult to find a uniform criterion for feature selection.

There is, however, one thing that all faults have in common. From the circuit point of view, the effect of all faults can be seen as a change of a particular intermediate variable from a correct value $x$ to a faulty value $x^{'}$. The distribution of the intermediate variable is disrupted with an abnormal bias by the change. Actually, this fact has already been exploited in the case of SFA[FJLT13] and SIFA[DEK+18]. In both studies, the adversary evaluates the difference between the uniform distribution, which is expected in the fault-free scenario, and the practical distribution, which is operating with a byte-level fault, using a metric named Squared Euclidean Imbalance (SEI), which can be described as follows:

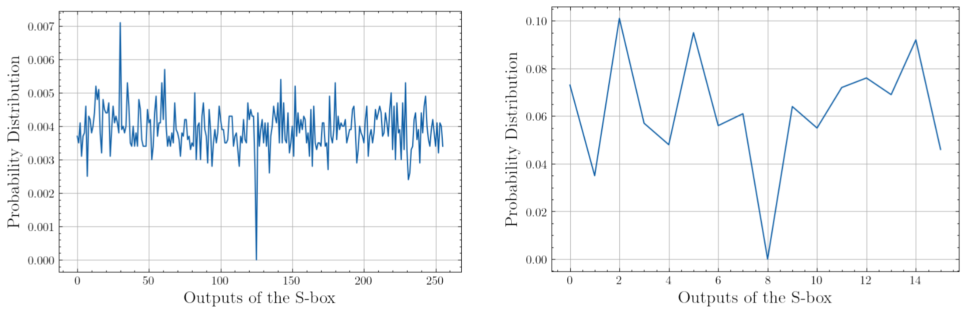
$$SEI(\hat{k}) = \sum_{\delta=0}^{2^s-1} \left( \frac{\#\left\{i \mid x = \delta\right\}}{N} - \frac{1}{2^s} \right)^2, \tag{1}$$

where $\hat{k}$ denotes the hypothesis of the key and $s$ denotes the bit length of the intermediate variable $x$. Thus the number of different values of $x$ can be expressed as $2^s$. We use $\frac{\#\{i|x=\delta\}}{N}$ and $\frac{1}{2^s}$ to denote the practical distribution of $x$ and the uniform distribution, respectively. The SEI of each $\hat{k}$ is computed as the sum of squared difference between the practical and the uniform distribution in all values of $x$. In the fault-free scenario, with the increase of encryption, the two distributions are almost equal thus the SEI of each $\hat{k}$ is close to 0. In the faulty scenario, for the wrong hypothesis of the key, each value of $x$ occurs randomly in the same probability. However, for the correct hypothesis of the key, with the bias caused by the fault, the distribution of $x$ becomes non-uniform, which leads to the increase of the SEI. Hence, the hypothesis that the key has the highest SEI can be considered correct.

Reviewing the fault model used in this article, we focus on the SPN-based block ciphers (e.g. AES and PRESENT) and the fault is injected into the S-box of the cipher. Hence, the output of the S-box can be considered as the intermediate variable affected by the fault. Due to the avalanche effect, for the correct encryption, the distribution of the output of the S-box converges to a uniform distribution, which is consistent with the SEI requirement. Unfortunately, the SEI is merely a calculated value that is too brief to be directly selected as the feature. However, the feature selection problem can be solved with the help of the concept of SEI. Note that the distribution of the intermediate variable in the fault-free scenario is a constant (i.e. $\frac{1}{2^s}$), which is associated only with the bit length of the intermediate variable. Thus, from the information theoretic point of view, the key-related information is contained in the practical distribution of the intermediate variable. That is, the adversary can select the practical distribution of the intermediate variables as the features.

## 2.3 Details of the Dataset

At the completion of feature selection, the dataset is prepared for the training and test of the neural network in the next stage. As mentioned above, we select the distribution of the output of the S-box of the SPN-based block cipher as the features. More specifically, the distribution of the output of the S-box in the last round is selected to reduce the computational overhead. The key of the last round is selected as the label for each group of features. Due to the nature of the block cipher, the features and labels are processed in bytes or nibbles. For the sake of simplicity, the term sample is used to denote a set of features. The examples of the sample on each cipher are given in Fig. 1.



(a) Example on AES (b) Example on PRESENT

**Figure 1:** Examples of the samples.

In addition to this, other relevant information, such as the plaintext, the ciphertext and the master key, can also be selected as the metadata. The efficiency of the neural network in deep learning techniques can be confirmed by the mere use of the labels. However, since these data are necessary to check for the correctness of the features and labels, the metadata are helpful for the adversary.

In consideration of the storage, all above data are stored in the current version 5 of the Hierarchical Data Format (HDF5), which is a cross-platform data storage format capable of accommodating extremely large, complex, high-dimensional, and heterogenous data. The structure of the dataset is shown in Fig. 2.

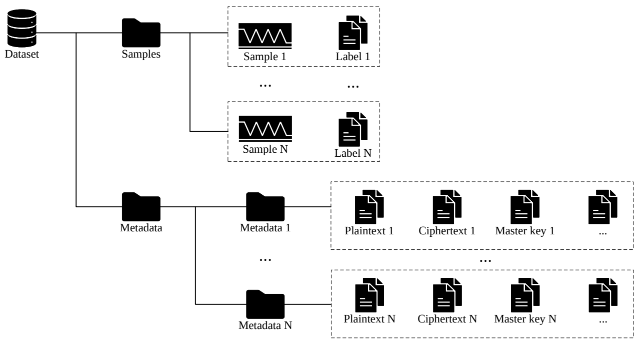


**Figure 2:** Structure of the dataset.

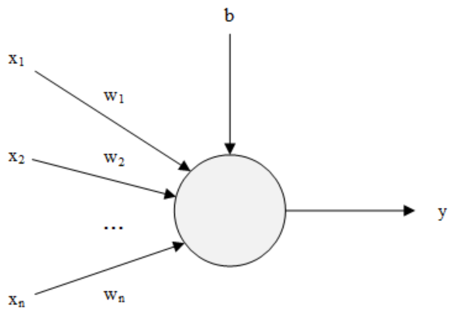
## 3 Deep Learning-Based Fault Analysis

With all our preparation behind us, we present our DLPFA in this section. The deep learning techniques used in this article are MLP and CNN. We first introduce the basic information about the two neural network models. The method of how to implement FA using deep learning techniques is shown later. Finally, we provide a general description of DLPFA in detail.

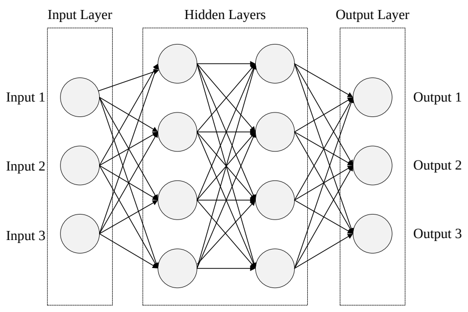
### 3.1 MLP and CNN

MLP and CNN are two widely used neural network models in deep learning techniques. Both of them fall into supervised learning, where a neural network is trained on data labeled with the corresponding labels. When the training phase is complete, the neural network is able to give the prediction for another set of inputs based on the rank of some indicators, such as probability or likelihood score.

MLP is a classical artificial neural network, inspired by the human nervous system. The artificial neuron (i.e. perceptron), whose basic structure is shown in Fig. 3(a), is the basic element of the network. In a mathematical sense, an artificial neuron consists of an affine transformation nested in an activation function, which can be defined as follows:



(a) Structure of single artificial neuron



(b) Structure of a four-layer MLP

**Figure 3:** Structure of the artificial neuron and MLP.

$$y = \alpha(\vec{w} \cdot \vec{x} + b), \tag{2}$$

where $\vec{x}$, $y$, $\alpha$ denote the input, output, and activation function of the artificial neuron, respectively. The weight and bias parameters of the affine transformation are denoted as $\vec{w}$ and $b$. Naturally, a single artificial neuron is powerless to deal with complex problems. Therefore, MLP is designed by joining artificial neurons together to adapt to the complicated situations. As shown in Fig. 3(b), the output of one layer is connected to the following layer as input. Let $\lambda$ denote the fully-connected layer, which is the set of the affine transformation with the same input, then the model of MLP $\mathcal{M}$ can be defined as follows:

$$\mathcal{M}(\vec{x}) = s \circ \lambda_n \circ \alpha_{n-1} \circ \lambda_{n-1} \circ ... \circ \alpha_1 \circ \lambda_1(\vec{x}). \tag{3}$$

Note that $s$ denotes the last activation function which is the special one. The activation functions from $\lambda_1$ to $\lambda_{n-1}$ have a variety of options, such as sigmoid or ReLU. However, $s$ is invariably chosen from the softmax function or its variants, which can be written as:

$$s(z_j) = \frac{e^{z_j}}{\sum_{i=1}^{C} e^{z_i}}, \tag{4}$$

where $C$ denotes the number of classes of the network's output. The softmax function renders the output of the network as a probability distribution of all classes. The classification problem can be solved by choosing the class with the highest one or several probabilities as the final output of the neural network model.

Compared to MLP, CNN is better at learning the internal representations of two-dimensional images. On one hand, CNN inherits the hierarchical structure from MLP. On the other hand, CNN broadens the type of layers to improve the efficiency of the network. In general, a CNN starts from the convolutional layer followed by the activation function. The convolutional layer extracts high-level abstract features from the input through the filters, and the activation function introduces non-linearities to increase the goodness-of-fit of the network. Then, the pooling layer is introduced to limit the number of neurons. The convolutional layer, the activation function and the pooling layer form the main block of the CNN, which is reused in the network until desired output. Moreover, the batch normalization layer is sometimes used as an option to improve the generalization ability of the network. After that, the flatten layer compresses the data into one dimension, and some fully-connected layers followed by a softmax function at the end of the network are added to summarize the global information and produce the final output. The structure of CNN is given in Fig. 4.

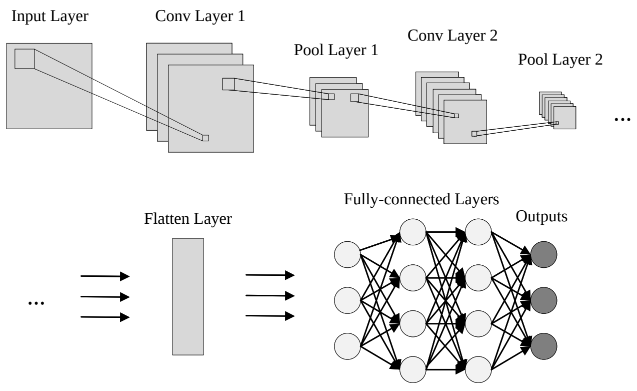


**Figure 4:** Structure of CNN.

Similarly, the model of CNN $\mathcal{C}$ can be characterized as follows:

$$\mathcal{C} : s \circ [\lambda]^{n_1} \circ [\rho \circ \alpha \circ \iota]^{n_2}, \tag{5}$$

where $\iota$ denotes the convolutional layer and $\rho$ denotes the pooling layer. Moreover, we recall that $\alpha$, $\lambda$ and $s$ denote the activation function, the fully-connected layer and the softmax function, respectively.

## 3.2 Combination of Fault Analysis and Deep Learning

To combine fault analysis and deep learning, our attack can be composed of two phases: the training phase and the attack phase. Thus, two independent datasets, the training set and the attack set, are prepared for the analysis. As mentioned above, the training set $\mathcal{S}_T$ of size $N_T$ consists of a quantity of independent identically distributed samples and the corresponding labels, which can be described as:

$$\mathcal{S}_T : \{(\vec{s_i}, k_i) | 1 \leq i \leq N_T\}, \tag{6}$$

where $\vec{s_i}$ denotes the sample, and $k_i$ denotes the label due to it represents the last round key in our dataset. In the training phase, the adversary aims to construct a generative model. The parameters of the network, denoted by $\theta$, will be updated iteratively to fit the estimation $g_k$ of every $k$ from the key space $\mathcal{K}$. The model $M_\theta$ can be described with the following conditional probability distribution function:

$$M_\theta : \{g_k : (\vec{s}, k) \to \Pr[\vec{S} = \vec{s} | K = k]\}_{k \in \mathcal{K}}, \tag{7}$$

where the upper-case letter $\vec{S}$ and $K$ denote the random variables over the set of $\vec{s}$ and $k$, respectively.

After the training phase, the adversary is able to perform the fault analysis in the attack phase with the attack set $\mathcal{S}_A$, which can be described as:

$$\mathcal{S}_A : \{(\vec{s_i}) | 1 \leq i \leq N_A\}. \tag{8}$$

Unlike the training set, the label (i.e. the key) of the attack set is unknown to the adversary. The goal of the attack phase is to recover the key from the dataset. Since the generative model has been constructed, the adversary needs to determine which of the key candidates is the most likely correct key. This problem is normally solved via the maximum likelihood method, which calculates a likelihood score $d_{\mathcal{S}_A}[k]$ for every key candidate $k \in \mathcal{K}$ with the help of Bayes' theorem as follows:

$$\begin{aligned} d_{\mathcal{S}_A}[k] &= \prod_{i=1}^{N_A} \Pr[K = k | \vec{S} = \vec{s_i}] \\ &= \prod_{i=1}^{N_A} \frac{\Pr[\vec{S} = \vec{s_i} | K = k]}{f_{\vec{S}}(\vec{s_i})} \times f_K(k), \end{aligned} \tag{9}$$

where $f_{\vec{S}}$ and $f_K$ denote the probability distribution function of $\vec{S}$ and $K$ respectively. The key candidate with the highest $d_{\mathcal{S}_A}[k]$ can be selected as the correct key.

## 3.3 General Description of DLPFA

Consistent with the previous analysis, we can give a general description of DLPFA here. To perform DLPFA, the following steps are required for the adversary:

**Injection:** At the beginning, the adversary injects the persistent fault into the S-box of the block cipher prior to the encryption. The faulty S-box will be used in the following encryption, and the effect of the fault will persist until the cryptographic device is rebooted. The persistent fault does not affect a single output of the S-box, but all outputs of the S-box in all rounds. However, we only focus on the effect of the fault in the last round.

**Preparation:** When the injection is complete, the next task of the adversary is to prepare the dataset. Similar to TA, we consider that the adversary is able to manipulate the faulty cryptographic device (or a cloned device) for encryption. For one sample in the dataset, the adversary performs encryption several times with the same key and random plaintexts. The metadata are collected to compute the distribution of the output of the S-box in the last round as features and the key of the last round as the label. The dataset is divided into the training set and attack set in a certain proportion. Although the labels in the attack set are useless for the adversary, we retain them to evaluate the performance of the attack.

**Training:** After that, the adversary needs to select the model and train the neural network. The parameters of the network are initialized with random values. These parameters are then updated to ensure that the predictions of the network are close to the labels. The loss function and the optimizer are usually used as hyper-parameters to aid training. The former measures the difference between the prediction and the label, and the latter determines the range of parameters to update based on the difference. The choice of the hyper-parameters will be discussed later, as it has a huge impact on the performance of the network.

**Attack:** Eventually, the adversary is able to use the trained network to mount a practical attack. One input of the attack set is presented to the network at a time, and the network computes a score vector containing the likelihood scores of all the key candidates. The candidate with the highest score is considered as the right key under attack. The adversary can evaluate the performance of the attack based on the metric, which is similar to the loss function but has no effect on the update of the parameters.

## 4 Experiments on AES-128

The experiments of DLPFA on AES are presented in this section. We first introduce the setup of our experiments. A brief overview of AES is then provided. The experiments are divided by the level of restriction on the position and value of the fault. We investigate the impact of the choice of the hyper-parameters on the training phase and present the experimental results on different datasets individually. Moreover, we separately discuss the experiments on the dataset with the highest level of restrictions due to their undesired performance.

### 4.1 Setups

The experiments are implemented at the software level on an ordinary computer with the following configuration: IntelCore i5-12400 CPU at 3.20 GHz, 32-GB DDR5 memory, 1-TB Samsung SSD-980-PRO drive and Nvidia GeForce RTX 3060 GPU. The development language is Python (version 3.9.7) and the development environment is Jupyter Notebook (version 6.4.5).

The fault injection of our experiments is realized by means of simulation. The persistent faults are injected by altering the value of the corresponding element in the S-box. For simplicity, a serial cryptographic implementation is considered in our experiments. However, our attack can also be applied to the parallelized cryptographic implementation as the fault scale is byte-level.

Our implementations of the deep learning neural network are developed with Keras library (version 2.9.0) as the frontend and Tensorflow library (version 2.9.1) as the backend. CUDA (version 11.7) is used to accelerate the training process. The entire computation of the tuning process takes approximately 300 hours.

## 4.2 Brief Overview of AES

AES is a widely used block cipher, which was published by NIST in 2001 as a standard for symmetric encryption. It comes with a block size of 128 bits and three standard key sizes of 128, 192 and 256 bits. In this article, we focus on the 128-bit variant AES-128, which consists of 10 rounds.

The round function of AES-128 operates on a $4 \times 4$-byte matrix (i.e. state matrix) and includes four byte-oriented operations: AddRoundKey (AK), SubBytes (SB), ShiftRows (SR) and MixColumns (MC). AK combines the state with the round key via bitwise exclusive-or. SB substitutes the state byte by byte with the S-box. SR cyclically shifts each row of the state a different offset. MC mixes each column of the state by multiplying it with a fix matrix. Note that the MC operation is missing in the last round based on the specification. Moreover, a KeyExpansions operation is applied to generate round keys from the master key using a key generation schedule.

## 4.3 Position and Value of the Fault

The position and value of the fault are the two requirements of the original PFA. In the DLPFA, we ignore these requirements and assume that the adversary has no knowledge of these information. However, the position and value of the fault can affect our attack in another way. The randomization of the position and value implies a variety of faults in the dataset. Given a fixed-size dataset, the more types of faults there are in the dataset, the less information each fault provides for the training of the neural network.

To evaluate our attack in stages, we prepare four different datasets representing four levels of restriction on the position and value of the fault. The first dataset, named FPFV dataset, assumes that the position and value of the fault are fixed. That is, all samples in the dataset are collected with one fixed fault. The FPRV and RPFV datasets are then prepared for the case where the position or value of the fault is random, respectively. Finally, in the RPRV dataset, we assume that the position and value of the fault are random, which is the highest level of restriction. Note that the dataset with higher level restriction is closer to the real situation, hence the experimental results in the RPRV dataset can best reflect the capabilities of our attack.

The initial size of each dataset is 50, 000, where the training set size is 45, 000 and the attack set size is 5, 000. We tune the hyper-parameters and show the experimental results for the four datasets separately.

## 4.4 Choice of the Hyper-Parameters

The discussion of tuning the hyper-parameters can be divided into two parts. We first focus on the parameters that define the network architecture (i.e. architecture parameters). Then the parameters that affect the training phase (i.e. training parameters) are discussed in the second part.

In the experiments of MLP model, three parameters are selected to characterize the architecture of MLP: the number of layers, the number of units in each layer and the activation function. The parameterization of an MLP architecture can be described as $\mathrm{MLP}_{arch}(n_{layers}, n_{units}, function)$. In the same way, we select the number of epochs, the batch size and the optimizer as the three training parameters and use $\mathrm{MLP}_{train}(n_{epochs}, \mathit{batch\_size}, \mathit{optimizer})$ to denote the parameterization of a training procedure.

The strategy of selecting the hyper-parameters for CNN model is inherited from the research in MLP context for fair comparison. However, the number of hyper-parameters of CNN model is too large for an exhaustive test of all the possible configurations. Hence, some representative parameters are selected to be tuned and the others are set up utilizing the work of Benadjila et al. in [BPS+20]. We consider a convolutional layer followed by a pooling layer as a block. Then the number of blocks, the number of fully-connected layers and the activation functions are selected as architecture parameters, which can be described as $\text{CNN}_{arch}(n_{blocks}, n_{FC\_layers}, function)$. Moreover, we fix the optimizer to Adam and select the number of epochs and the batch size as training parameters with the description of $\text{CNN}_{train}(n_{epochs}, batch\_size)$.

## 4.5 Evaluation Metrics

To evaluate the performance of our attack, we choose guessing entropy (GE) [SMY09] as the metric in this article. GE is a widely used metric in SCA, which can give information about the trend of the attack and the size of data required for a successful attack. Compared with traditional deep learning metrics like accuracy, GE can give a more fair evaluation of the attack as it assesses the attack by the mean rank of the secret key rather than the success rate of the attack, which gives a more objective assessment of the failed attack.

As mentioned above, for an attack set $\mathcal{S}_A$ of size $N_T$ with the right key $k^*$, the adversary is able to calculate the likelihood score $d_{\mathcal{S}_A}[k]$ for every key candidate $k$. Then the adversary can sort the scores from the largest to the smallest to get a rank vector. GE calculates the mean position of the right key $k^*$ in the rank vector, which can be described as:

$$\mathbf{GE} = \underset{\mathcal{S}_A}{\mathbb{E}}\left[|\{k \in \mathcal{K} | d_{\mathcal{S}_A}[k] > d_{\mathcal{S}_A}[k^*]\}|\right]. \tag{10}$$

## 4.6 Experiments of MLP Model

For the sake of fairness, we tune the architecture parameters with the fixed training parameters $\text{MLP}_{train}(40, 150, Adam)$, which denotes 40 epochs, batch size 150 and the Adam optimizer. Then in the second part of the experiment, different training parameters are tested with the tuned architecture parameters.

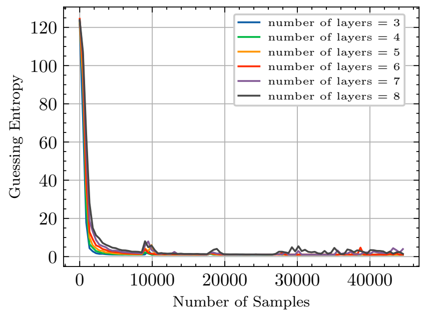
**Choice of Architecture Parameters:** We start the experiment with an evaluation on the number of layers. We fix the number of units at 200 and select the eLU as the activation function. Then MLP model is trained with different $n_{layers} \in \{3, \cdots 8,\}$. As shown in Fig. 5, the 3-layer MLP has the lowest GE on the first three datasets. However, due to the confusion in the results, we are unable to derive anything from the experiments on the RPRV dataset.

The number of units in each layer is then evaluated. We train a 3-layer MLP model with the eLU activation function and different $n_{units} \in \{50, 100, \cdots, 250, 300\}$. As Fig. 6 shows, our attack has a better performance on the first three datasets with the increase of $n_{units}$. However, the larger value of $n_{units}$ will lead to higher computational complexity of the model. Taking the trade-off between efficiency and complexity into account, a compromise of $n_{units}$ is 150.

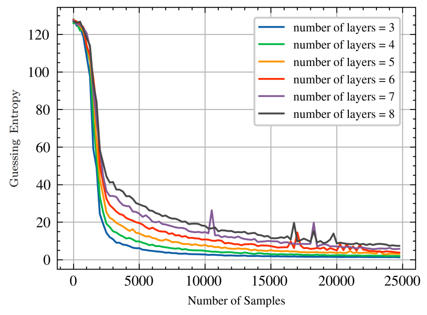
The last architecture parameter we tune is the activation function. We select three activation functions as candidates: sigmoid, tanh and eLU. The first two are common deep learning activation functions and the last one is a modified version of ReLU, which has achieved remarkable performance in the field of image recognition. The experimental results in Fig. 7 show that each candidate performs well and eLU has a slight advantage on the complicated dataset.

Note that the experimental results on the first three datasets are consistent. Hence, we present the experimental results on the RPFV dataset as a representative in the following study. Moreover, we temporarily put aside the experiments on the RPRV dataset due to its poor performance and discuss the issue with this dataset in the last subsection.

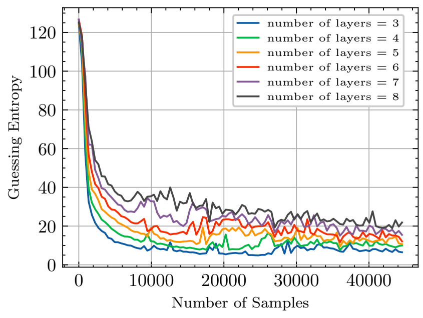
**Choice of Training Parameters:** When the choice of architecture parameters is finished, the training parameters are tuned with the determined architecture parameters



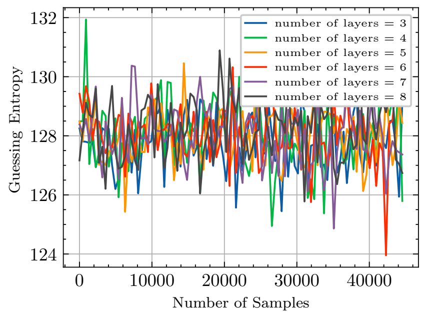
(a) Experiments on the FPFV dataset



(b) Experiments on the FPRV dataset

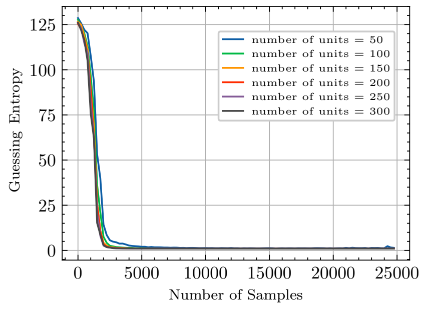


(c) Experiments on the RPFV dataset

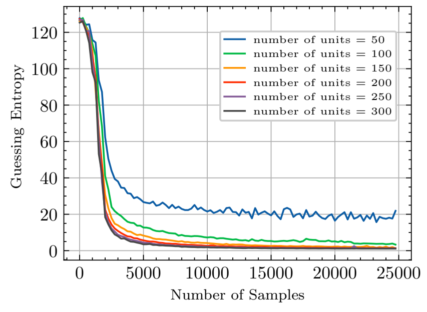


(d) Experiments on the RPRV dataset

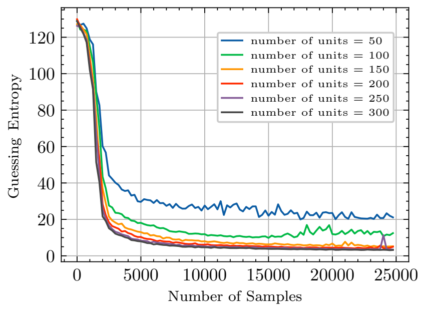
**Figure 5:** GE of MLP$_{arch}(n_{layer}, 200, eLU)$ trained with MLP$_{train}(40, 150, Adam)$.



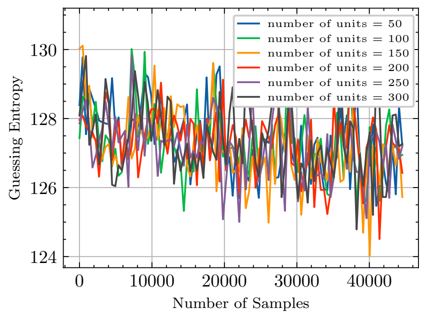
(a) Experiments on the FPFV dataset



(b) Experiments on the FPRV dataset

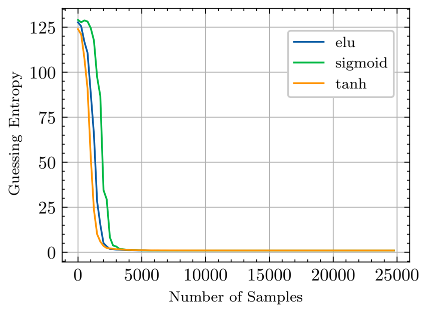


(c) Experiments on the RPFV dataset

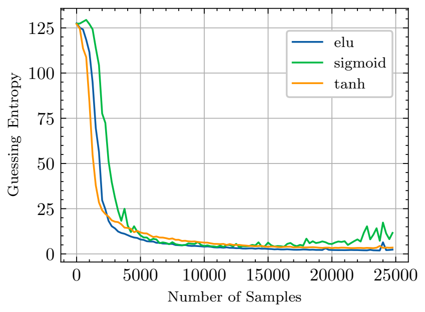


(d) Experiments on the RPRV dataset

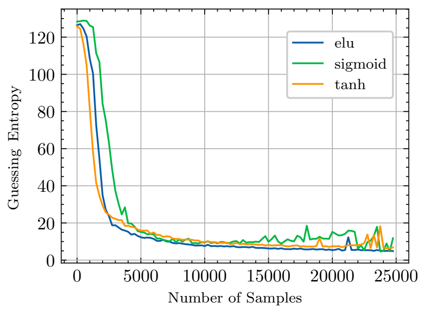
**Figure 6:** GE of MLP$_{arch}(3, n_{units}, eLU)$ trained with MLP$_{train}(40, 150, Adam)$.



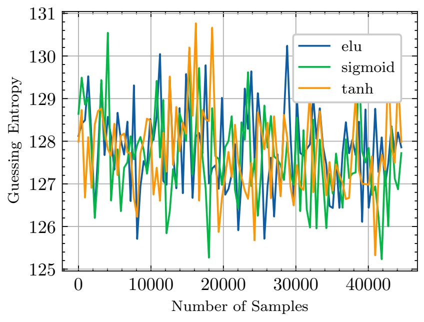
(a) Experiments on the FPFV dataset



(b) Experiments on the FPRV dataset



(c) Experiments on the RPFV dataset



(d) Experiments on the RPRV dataset

**Figure 7:** GE of MLP$_{arch}(3, 150, function)$ trained with MLP$_{train}(40, 150, Adam)$.

MLP$_{arch}(3, 200, eLU)$. Initially, we fix the batch size at 150 and the optimizer at Adam to select the best value for the number of epochs. Based on the experimental results in Fig. 8(a), we choose $n_{epochs} = 40$ as a good trade-off between efficiency and computation time.

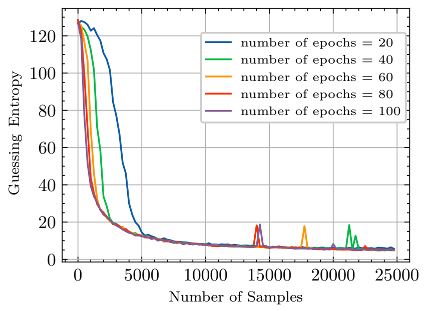
The next training parameter we tune is the batch size. As shown in Fig. 8(b), 100 is a sound choice of batch size due to its efficiency in small data size and stability in large data size.

Finally, we evaluate the effect of the optimizer on the performance of the neural network. We provide three options: RMSprop, SGD and Adam. The learning rate for the first two is fixed to $10^{-5}$ and the last one is an adaptive optimizer which can dynamically adjust the learning rate by itself. The experimental results in Fig. 8(c) show that the choice of optimizer has a significant effect on GE. We manage to obtain good results with the Adam optimizer.

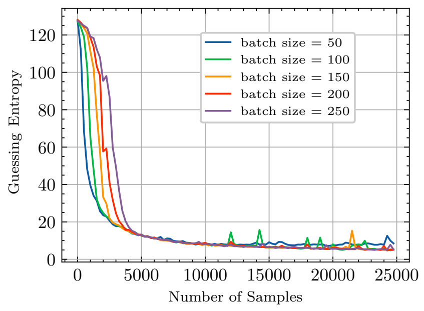
In summary, approximately 25, 000 training samples are sufficient to perform a successful attack with these architecture and training parameters. Moreover, these tuned hyper-parameters are used for reference in the rest of the experiments in this article.

## 4.7 Experiments of CNN Model

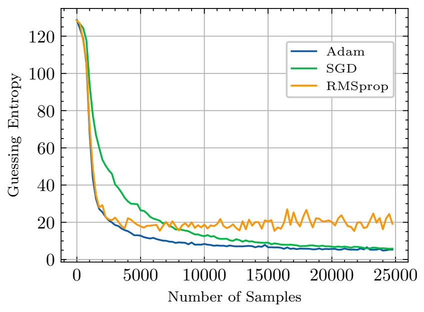
Due to the large search space of the hyper-parameters of CNN model, the goal of the following experiments is not to find the optimal value, but rather to obtain a better value that leads to satisfactory results. Before starting the experiment, some initial configurations should be set up following the experience of the predecessors. We initially fix the kernel size of the convolutional layer at 11. Then the number of filters in the $i$-th convolutional layer is set to $64 \times 2^{i-1}$ with an upper limit of 512. We choose the average pooling as the pooling method and set the padding option to the same padding. Ultimately, the number of units in a fully-connected layer is fixed at 4096.



(a) Experiments of $n_{epochs}$



(b) Experiments of batch size



(c) Experiments of optimizer

**Figure 8:** GE of MLP$_{train}(n_{epochs}, batch\_size, optimizer)$ trained with MLP$_{arch}(3, 150, eLU)$ for different $n_{epochs}$, batch size and optimizer.

We present the benchmark parameters of CNN model below. We start the experiment with the architecture parameters CNN$_{arch}(1, 1, eLU)$ and the training parameters CNN$_{train}(40, 100)$. First, we are concerned with the experiments on the RPRV dataset. The results in Fig. 9 show that CNN model cannot handle this dataset better than MLP model. Therefore, we continue to investigate the effect of the hyper-parameters from the experimental results on the RPFV dataset.

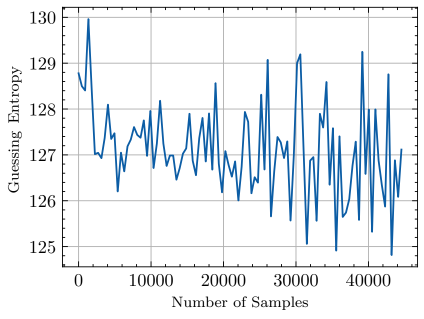
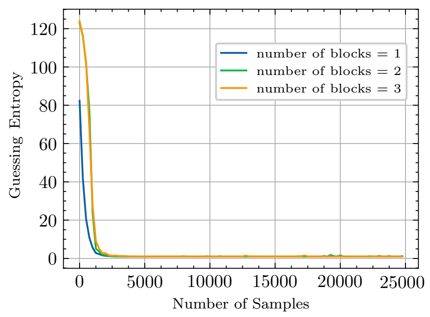


**Figure 9:** Experimental results of CNN model on the RPRV dataset.

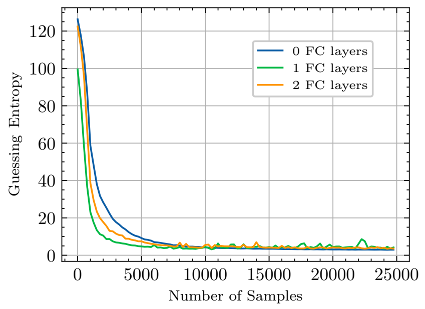
**Choice of Architecture Parameters:** We first evaluate the impact of the number of blocks. As Fig. 10(a) shows, the efficiency of the model is dimly impacted by $n_{blocks}$. One block is sufficient for CNN model to mount a successful attack. Hence, we set the number of blocks to 1 and fix the number of filters in the convolutional layers at 64.

We then investigate the effect of the number of fully-connected layers. Unexpectedly, Fig. 10(b) shows that the network without fully-connected layers performs equally well. This fact can be explained by the functionality of the fully-connected layer. The fully-connected layer integrates and reduces the dimensionality of the characteristics extracted by the network. Since the dataset is lower dimensional and the network model is compact, the fully-connected layer is unnecessary for our experiments. However, the best results are obtained with one fully-connected layer. We recommend keeping the value $n_{FC\_layers} = 1$ to improve the performance of our CNN model.

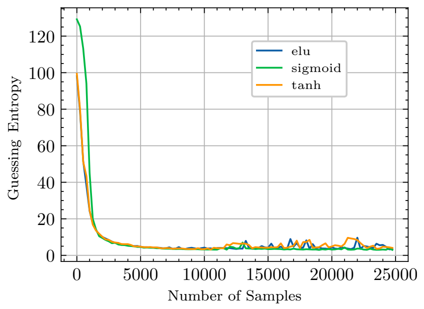
Finally, the effect of the activation function is evaluated. Results given in Fig. 10(c) show that all the activation functions are suitable for our attack. We select eLU as it has less overhead than the other two functions.



(a) Experiments of $n_{blocks}$



(b) Experiments of $n_{FC\_layers}$



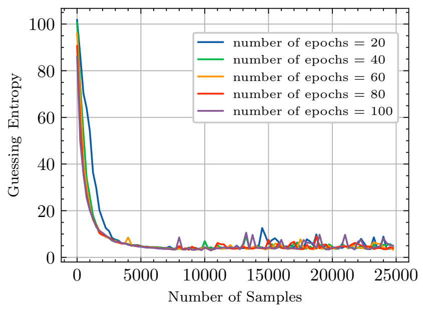
(c) Experiments of activation function

**Figure 10:** GE of $\text{CNN}_{arch}(n_{blocks}, n_{FC\_layers}, function)$ trained with $\text{CNN}_{train}(40, 100)$ for different $n_{blocks}$, $n_{FC\_layers}$ and activation function.

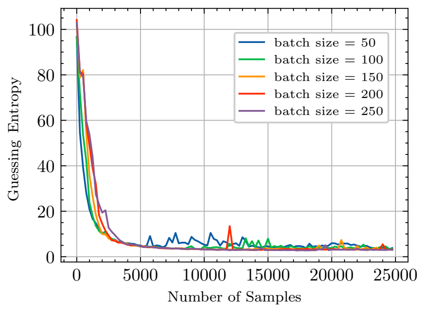
**Choice of Training Parameters:** We first investigate the impact of the number of epochs on the model efficiency. The experimental results are plotted in Fig. 11(a). Similar to MLP model, the efficiency of the model increases when the number of epochs increases. However, the experiments need to be performed in a reasonable amount of time. We select 40 as an appropriate value of $n_{epochs}$.

Then we evaluate the performance of CNN model with different values of the batch size. Fig. 11(b) shows that each batch size can perform well with a large size of training set. However, when the size of training set is smaller than 5, 000, GE decreases more steeply for smaller batch size. Therefore, we recommend fixing the batch size to 50 to obtain decent global results.

The experimental results show that our CNN model with tuned hyper-parameters is efficient for the fault analysis. However, since the optimization of the hyper-parameters is not our goal, we believe that more appropriate parameters may be found to obtain better results in the following studies.



(a) Experiments of $n_{epochs}$



(b) Experiments of batch size

**Figure 11:** GE of CNN$_{train}(n_{epochs}, batch\_size)$ trained with CNN$_{arch}(1, 1, eLU)$ for different $n_{epochs}$ and batch size.

## 4.8 Discussion about the Experiments on the RPRV Dataset

The experiments demonstrate that MLP and CNN models can mount successful attacks as expected on the FPFV, FPRV and RPFV datasets. Unfortunately, these models perform poorly on the RPRV dataset. We attribute the failure to the increased randomness of the dataset. The randomness can be roughly quantified in terms of the possibility of the fault. Reviewing the configuration of the dataset, there is only one possibility of the fault in the FPFV dataset as the position and value of the fault are fixed. According to the characteristics of AES, the faults of the S-box have 256 random values and 256 random positions. Thus, there are 256 possible faults in the FPRV and RPFV datasets and $65,536$ possible faults in the RPRV dataset.

Considering the training set size of 45, 000 in the experiments, the network is hardly to learn enough knowledge for all the possibilities of faults in the RPRV dataset. To overcome this challenge, we expand the training set size to more than 800, 000. The experimental results of CNN model are plotted in Fig. 12. The decreasing trend of GE confirms that the increase of training set is beneficial for our attack in the RPRV dataset.

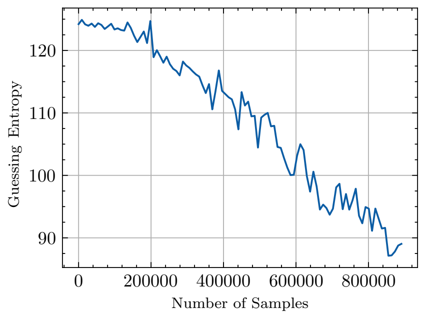


**Figure 12:** Experimental results of CNN model on the expanded RPRV dataset.

To further validate our point of view, we extend our attack to the lightweight SPN-based cipher in the next section. We choose PRESENT as the target due to its pint-sized S-box.

## 5 Experiments on PRESENT

DLPFA has been successfully applied on AES-128 to the first three datasets. In this section, our attack is extended to PRESENT to undertake a further study of its performance on the RPRV dataset. Moreover, we end this section with some comparisons of DLPFA and related works.

## 5.1 Brief Overview of PRESENT

PRESENT is an AES-like lightweight block cipher which was proposed in [BKL+07]. It is designed with area and power constraints and is more suitable for extremely resource-constrained environments.

PRESENT consists of 31 rounds and comes with a block size of 64 bits. Two key lengths of 80 and 128 bits are supported. We focus on the version with 128-bit key in this article. Compared to AES, PRESENT has a similar round function consisting of AddRoundKey, sBoxlayer and pLayer. However, PRESENT uses a 4-bit to 4-bit S-box to substitute these nibbles in the state matrix. Smaller S-box decreases the possibilities of the fault.

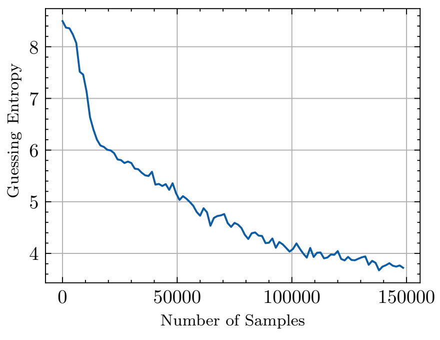
## 5.2 Preparation of the Dataset

The dataset is reprepared for the experiments on PRESENT. We locate our attack in the last round of the cipher as before. However, due to the 4-bit S-box, the features and labels are processed in nibbles. Since the effectiveness of DLPFA on the first three datasets has already been demonstrated, we only focus on the experiments on the RPRV dataset.

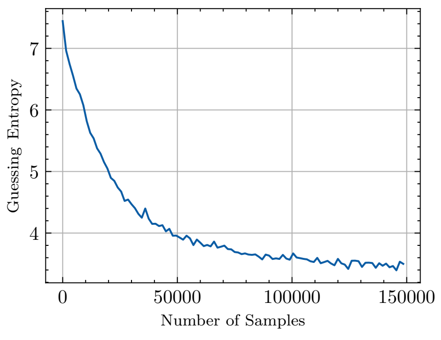
The size of the dataset is enlarged to increase the amount of information contained. We prepare 150, 000 samples for the training set and 15, 000 samples for the attack set.

## 5.3 Experiments on the RPRV Dataset

We start the experiments of MLP and CNN models with the hyper-parameters tuned in the previous section. As shown in Fig. 13, both models have decent results on the RPRV dataset. GE eventually approximately converges to 3, which implies the existence of an average of three possible key candidates. The adversary can easily recover the key with a brute force search.



(a) Experiments of MLP model



(b) Experiments of CNN model

**Figure 13:** Experimental results of MLP and CNN model on PRESENT cipher.

Then we tune the hyper-parameters of both models. We search for parameters from a wider range and the tuned hyper-parameters are summarized in Tables 1 and 2. We recommend the more sophisticated models to improve the efficiency in the training phase.

**Table 1:** Summary of the tuned hyper-parameters of MLP model

| Hyper-Parameter | Reference | Range | Choice |
|---|---|---|---|
| Architecture Parameters | | | |
| Layers | $n_{layer}$ | {1,2,3,4,5} | 5 |
| Units | $n_{units}$ | {50,100, $\cdots$ ,350,400} | 400 |
| Activation Function | - | eLU, Sigmoid, Tanh | eLU |
| Training Parameters | | | |
| Epochs | $n_{epochs}$ | {10,15, $\cdots$ ,55,60} | 20 |
| Batch Size | - | {50,100, $\cdots$ ,450,500} | 500 |
| Optimizer | - | Adam, SGD, RMSprop | Adam |

**Table 2:** Summary of the tuned hyper-parameters of CNN model

| Hyper-Parameter | Reference | Range | Choice |
|---|---|---|---|
| Architecture Parameters | | | |
| Blocks | $n_{blocks}$ | {1,2,3,4} | 4 |
| FC Layers | $n_{FC\_layers}$ | {0,1,2} | 1 |
| Activation Function | - | eLU, Sigmoid, Tanh | eLU |
| Training Parameters | | | |
| Epochs | $n_{epochs}$ | {10,20, $\cdots$ ,40,50} | 20 |
| Batch Size | - | {50,100, $\cdots$ ,450,500} | 500 |

### 5.4 Comparisons of DLPFA and Related Works

At the end of this section, we present some comparisons of DLPFA and related works to give a thorough analysis of our attack. We first compare our attack with classical transient fault analysis like DFA and SFA. Our attack has a more relaxed time window for fault injection thanks to the persistent fault model. Moreover, at least one fault is sufficient to mount our attack, but the transient analysis usually requires multiple faults from multiple injections.

Then we compare our attack with the persistent fault-based attacks like PFA and EPFA[XZY$^+$21]. Indeed, all the faulty ciphertexts used in the analysis of these persistent fault-based attacks are collected from the same fault, which corresponds to the case of the FPFV dataset in our attack. However, DLPFA is able to perform successful attacks on the FPRV, RPFV and RPRV datasets, which have higher level of restriction. That is, our attack is closer to the actual situation.

The final comparison is between DLPFA and deep learning-based SCA, since they both belong to the applications of neural networks in the field of cryptography. The dataset in the deep learning-based SCA consists of the power traces, which contain the computation of the entire or some rounds of the cipher and are collected from multiple encryptions. The adversary has to solve the problems of the choice of Points-of-Interest (POIs) [ZZY$^+$14] and the synchronization of the traces [CDP17b]. However, thanks to our feature selection method, there is no need to preprocess the dataset of DLPFA. This greatly reduces the work of the adversary.

## 6 Conclusions

In this article, we conduct the first thorough study on the application of deep learning based key recovery in the field of fault analysis. One of the main motivations for using deep learning techniques is their outstanding results in side-channel analysis. We propose DLPFA as a novel FA method and mount successful attacks with MLP and CNN models on several classical and modern block ciphers. In particular, we discuss the choice of hyper-parameters and present the tuned parameterization options as a reference for the design of new models. Since the current state of deep learning theory does not yet provide a clear basis for such analysis, we believe that having a methodology is a first necessary step that opens the way for further research in this domain.

For future work, it might be interesting to improve the performance of DLPFA on the RPRV dataset. Another potential future direction is to change the fault model and seek more efficient feature selection methods. Moreover, how to break protected cryptographic implementations is also an interesting topic for future work.

## References

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156, pages 16–29. Springer, 2004.

[BDG+10] Michael Backes, Markus Dürmuth, Sebastian Gerling, Manfred Pinkal, and Caroline Sporleder. Acoustic side-channel attacks on printers. In *19th USENIX Security Symposium, Washington, DC, USA, August 11-13, 2010, Proceedings*, pages 307–322. USENIX Association, 2010.

[BHJ+18] Jakub Breier, Xiaolu Hou, Dirmanto Jap, Lei Ma, Shivam Bhasin, and Yang Liu. Practical fault attack on deep neural networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 2204–2206. ACM, 2018.

[BKL+07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: an ultra-lightweight block cipher. In *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727, pages 450–466. Springer, 2007.

[BL12] Timo Bartkewitz and Kerstin Lemke-Rust. Efficient template attacks based on probabilistic multi-class support vector machines. In *Smart Card Research and Advanced Applications - 11th International Conference, CARDIS 2012, Graz, Austria, November 28-30, 2012, Revised Selected Papers*, volume 7771, pages 263–276. Springer, 2012.

[BMM00] Ingrid Biehl, Bernd Meyer, and Volker Müller. Differential fault attacks on elliptic curve cryptosystems. In *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, volume 1880, pages 131–146. Springer, 2000.

[BPS+20] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. Deep learning for side-channel analysis and introduction to ASCAD database. *J. Cryptogr. Eng.*, 10(2):163–188, 2020.

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, volume 1294, pages 513–525. Springer, 1997.

[CDP17a] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures - profiling attacks without pre-processing. In *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529, pages 45–68. Springer, 2017.

[CDP17b] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures - profiling attacks without pre-processing. In *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529, pages 45–68. Springer, 2017.

[CLFT14] Franck Courbon, Philippe Loubet-Moundi, Jacques J. A. Fournier, and Assia Tria. Adjusting laser injections for fully controlled faults. In *Constructive Side-Channel Analysis and Secure Design - 5th International Workshop, COSADE 2014, Paris, France, April 13-15, 2014. Revised Selected Papers*, volume 8622, pages 229–242. Springer, 2014.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523, pages 13–28. Springer, 2002.

[DEK$^+$18] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Stefan Mangard, Florian Mendel, and Robert Primas. SIFA: exploiting ineffective fault inductions on symmetric cryptography. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):547–572, 2018.

[FJLT13] Thomas Fuhr, Éliane Jaulmes, Victor Lomné, and Adrian Thillard. Fault attacks on AES with faulty ciphertexts only. In *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography, Los Alamitos, CA, USA, August 20, 2013*, pages 108–118. IEEE Computer Society, 2013.

[GA03] Sudhakar Govindavajhala and Andrew W. Appel. Using memory errors to attack a virtual machine. In *2003 IEEE Symposium on Security and Privacy (S&P 2003), 11-14 May 2003, Berkeley, CA, USA*, pages 154–165. IEEE Computer Society, 2003.

[GHO15] Richard Gilmore, Neil Hanley, and Máire O'Neill. Neural network based attack on a masked implementation of AES. In *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2015, Washington, DC, USA, 5-7 May, 2015*, pages 106–111. IEEE Computer Society, 2015.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162, pages 251–261. Springer, 2001.

[HGM$^+$11] Gabriel Hospodar, Benedikt Gierlichs, Elke De Mulder, Ingrid Verbauwhede, and Joos Vandewalle. Machine learning in side-channel analysis: a first study. *J. Cryptogr. Eng.*, 1(4):293–302, 2011.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology*

*Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666, pages 388–397. Springer, 1999.

[KPH+19] Jaehun Kim, Stjepan Picek, Annelie Heuser, Shivam Bhasin, and Alan Hanjalic. Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(3):148–179, 2019.

[LBM15] Liran Lerman, Gianluca Bontempi, and Olivier Markowitch. A machine learning approach against a masked AES - reaching the limit of side-channel attacks with a learning model. *J. Cryptogr. Eng.*, 5(2):123–139, 2015.

[LPB+15] Liran Lerman, Romain Poussier, Gianluca Bontempi, Olivier Markowitch, and François-Xavier Standaert. Template attacks vs. machine learning revisited (and the curse of dimensionality in side-channel analysis). In *Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers*, volume 9064, pages 20–33. Springer, 2015.

[LWLX17] Yannan Liu, Lingxiao Wei, Bo Luo, and Qiang Xu. Fault injection attack on deep neural network. In *2017 IEEE/ACM International Conference on Computer-Aided Design, ICCAD 2017, Irvine, CA, USA, November 13-16, 2017*, pages 131–138. IEEE, 2017.

[Man02] Stefan Mangard. A simple power-analysis (SPA) attack on implementations of the AES key expansion. In *Information Security and Cryptology - ICISC 2002, 5th International Conference Seoul, Korea, November 28-29, 2002, Revised Papers*, volume 2587, pages 343–358. Springer, 2002.

[MDM16] Zdenek Martinasek, Petr Dzurenda, and Lukas Malina. Profiling power analysis attack based on MLP in DPA contest V4.2. In *39th International Conference on Telecommunications and Signal Processing, TSP 2016, Vienna, Austria, June 27-29, 2016*, pages 223–226. IEEE, 2016.

[MZVT15] Zdenek Martinasek, Ondrej Zapletal, Kamil Vrba, and Krisztina Trasy. Power analysis attack based on the MLP in DPA contest v4. In *38th International Conference on Telecommunications and Signal Processing, TSP 2015, Prague, Czech Republic, July 9-11, 2015*, pages 154–158. IEEE, 2015.

[PSK+18] Stjepan Picek, Ioannis Petros Samiotis, Jaehun Kim, Annelie Heuser, Shivam Bhasin, and Axel Legay. On the performance of convolutional neural networks for side-channel analysis. In *Security, Privacy, and Applied Cryptography Engineering - 8th International Conference, SPACE 2018, Kanpur, India, December 15-19, 2018, Proceedings*, volume 11348, pages 157–176. Springer, 2018.

[Riv91] Ronald L. Rivest. Cryptography and machine learning. In *Advances in Cryptology - ASIACRYPT '91, International Conference on the Theory and Applications of Cryptology, Fujiyoshida, Japan, November 11-14, 1991, Proceedings*, volume 739, pages 427–439. Springer, 1991.

[SAB+20] Sayandeep Saha, Manaar Alam, Arnab Bag, Debdeep Mukhopadhyay, and Pallab Dasgupta. Leakage assessment in fault attacks: A deep learning perspective. *IACR Cryptol. ePrint Arch.*, page 306, 2020.

[SMY09] François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479, pages 443–461. Springer, 2009.

[Tim19] Benjamin Timon. Non-profiled deep learning-based side-channel attacks with sensitivity analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):107–131, 2019.

[XZY+21] Guorui Xu, Fan Zhang, Bolin Yang, Xinjie Zhao, Wei He, and Kui Ren. Pushing the limit of PFA: enhanced persistent fault analysis on block ciphers. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 40(6):1102–1116, 2021.

[ZLZ+18] Fan Zhang, Xiaoxuan Lou, Xinjie Zhao, Shivam Bhasin, Wei He, Ruyi Ding, Samiya Qureshi, and Kui Ren. Persistent fault analysis on block ciphers. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):150–172, 2018.

[ZZY+14] Yingxian Zheng, Yongbin Zhou, Zhenmei Yu, Chengyu Hu, and Hailong Zhang. How to compare selections of points of interest for side-channel distinguishers in practice? In *Information and Communications Security - 16th International Conference, ICICS 2014, Hong Kong, China, December 16-17, 2014, Revised Selected Papers*, volume 8958, pages 200–214. Springer, 2014.