DLFA: Deep Learning-based Fault Analysis against Block Ciphers

Anonymous Submission

Abstract. The proliferation of embedded cryptographic devices in the Internet of Things (IoT) ecosystem has elevated the importance of physical security assessments. Although traditional Fault Analysis (FA) methods exhibit significant effectiveness in cryptographic key recovery, their practical application is heavily constrained by rigid mathematical requirements, the demand for precise physical fault injection, and a sensitivity to measurement noise. To address these limitations, this paper proposes Deep Learning-based Fault Analysis (DLFA), a comprehensive attack framework. By decomposing cryptanalysis into tailored feature engineering and neural network classification, DLFA successfully unifies four prominent fault models (i.e., Differential (DFA), Statistical (SFA), Statistical Ineffective (SIFA), and Persistent Fault Analysis (PFA)) under a single data-driven paradigm. Extensive physical evaluations on the SAKURA-G FPGA implementing AES-128 demonstrate that DLFA reduces the data complexity and computational time overhead compared to classical algebraic solvers. More crucially, DLFA exhibits sustained analytical stability against severe physical injection noise, relaxing the stringent hardware requirements for attackers. Finally, we employ the Integrated Gradients (IG) principle to conduct a quantitative attribution analysis, proving that the neural networks autonomously learn valid cryptographic leakages rather than overfitting to experimental artifacts.

Keywords: Deep learning · Fault analysis · Cryptographic security

1 Introduction

To ensure a reliable and efficient computing environment, cryptographic algorithms are implemented in specialized cryptographic devices. The security of these devices relies not only on the mathematical robustness of the cryptographic algorithm but also on the physical security of the cryptographic implementation. Fault Analysis (FA) represents one of the most potent active attack vectors, where an adversary intentionally perturbs the device's operating conditions (e.g., manipulating the power supply voltage [BMM00], varying the environmental temperature [GA03] or utilizing pulsed lasers [CLFT14]) to induce computational errors. The analysis of the resulting faulty outputs often reveals the internal state of the cipher, achieving key recovery with significantly fewer data samples than passive Side-Channel Analysis (SCA).

Despite the efficacy of classical FA methods, they are inherently limited by their reliance on rigid mathematical models and idealized fault assumptions. For instance, Differential Fault Analysis (DFA) [BS97] requires the adversary to solve complex systems of non-linear equations derived from precise fault models (e.g., a strict single-byte flip). Similarly, advanced methodologies like Statistical Fault Analysis (SFA) [FJLT13], Statistical Ineffective Fault Analysis (SIFA) [DEK+18], and Persistent Fault Analysis (PFA) [ZLZ+18] introduce new challenges, heavily depending on massive data accumulation and deep domain expertise to construct manual statistical distinguishers. Furthermore, these traditional analytical solvers lack functional resilience. When confronted with the inherent stochastic noise of real-world physical injections, they frequently experience analytical failures or yield false positives.

To mitigate this physical fragility, researchers have explored the integration of data-driven methodologies, primarily relying on traditional Machine Learning (ML) techniques such as Support Vector Machines (SVM) and Random Forests [HGM+11, LBM15]. However, these classical ML paradigms remain heavily constrained by their dependence on expert-driven, hand-crafted feature engineering to pre-process and linearize data before inference [LBM15]. Furthermore, due to their bounded model capacity, traditional ML classifiers inherently struggle to capture the algebraic non-linearity of cryptographic transformations (e.g., the $GF(2^8)$ S-box mappings) and frequently exhibit performance degradation when exposed to the unstructured, stochastic noise of physical fault injections [PHJ+19].

In contrast, the paradigm shift toward Deep Learning (DL) has revolutionized physical cryptanalysis [MPP16, ZBHV20, PPM+23]. By leveraging deep neural networks as universal function approximators, DL demonstrates an exceptional ability to autonomously extract robust, high-dimensional representation features directly from raw data, entirely bypassing the need for manual feature engineering. Yet, despite its success in breaking protected implementations in passive SCA [KPH+19, BPS+20], the application of DL to the active domain of FA remains fundamentally underexplored. Existing research has primarily utilized DL for SCA evaluations or basic fault leakage assessments [SAB+23], leaving a significant void in the development of universal deep learning frameworks capable of direct key recovery from diverse faulty ciphertexts.

To bridge this gap and overcome the physical fragility of classical methods, this paper proposes Deep Learning-based Fault Analysis (DLFA). DLFA is a comprehensive framework that reframes cryptanalysis from manual algebraic derivation to a statistical pattern recognition task. Through a divide-and-conquer strategy comprising bespoke feature selection and specialized neural architectures (e.g., Algebraic Verifiers, Weight-Shared networks, and Domain-Informed layers), DLFA effectively adapts to the physical leakages of transient, statistical, ineffective, and persistent faults. The specific contributions of this work are as follows:

- **Unified Framework for DL-based Key Recovery.** We propose a generalized DLFA framework that successfully integrates various fault models, including DFA, SFA, SIFA, and PFA. By abstracting the fault mechanism into a classification problem, the framework enables adversaries to bypass complex mathematical derivations and manual equation solving.

- **Specialized Feature Selection and Architectural Design.** Departing from generic hyperparameter tuning, we propose a domain-driven approach to feature engineering and model construction. We develop specialized feature extraction strategies, ranging from differential patterns for DFA to raw sequences for SFA and aggregated histograms for SIFA/PFA. To process these heterogeneous inputs, we design specific architectural primitives, including Embedding layers to handle non-ordinal $GF(2^8)$ values, TimeDistributed layers for parallel byte-wise processing, and custom Algebraic Verifier frameworks. This methodology ensures that the neural network architecture is intrinsically aligned with the underlying cryptographic transformations.

- **Superior Efficiency and Physical Noise Robustness.** Through extensive physical evaluations on the SAKURA-G platform, we demonstrate that DLFA achieves a significant reduction in data complexity compared to classical solvers, such as a 33% reduction in traces for SFA and a 41% reduction for SIFA. More crucially, our framework exhibits resilience to physical noise. While traditional algebraic methods typically experience analytical failures when the fault injection success rate drops below 90%, the proposed models maintain sustained analytical stability even at

a higher noise level, fundamentally lowering the precision requirements for injection equipment.

- **Interpretability via Integrated Gradients.** Acknowledging the black-box criticism of neural networks in security-critical applications, we incorporate the Integrated Gradients (IG) interpretability method into our framework. We provide a detailed attribution analysis generated by the trained models, demonstrating that the networks are learning valid cryptanalytic features, rather than overfitting to dataset artifacts. This contributes to a deeper understanding of how deep learning automates the discovery of cryptanalytic vulnerabilities.

2 Preliminaries

This section establishes the theoretical foundations necessary for understanding the proposed DLFA framework. We begin with an overview of the deep learning architectures utilized in this study, followed by a detailed mathematical treatment of the classical fault analysis methods that underpin our feature extraction strategies. Finally, we review related works to contextualize the novelty of our approach.

2.1 Deep Learning Architectures

Deep learning employs artificial neural networks with multiple processing layers to learn intricate representations of data. In the context of cryptanalysis, these networks function as universal function approximators, capable of mapping physical leakages (e.g., faulty ciphertexts, probability distributions, or differential states) directly to secret keys. To accommodate the diverse data representations across different fault models (e.g., DFA, SFA, SIFA, and PFA), our framework extends beyond basic models, categorizing the utilized neural networks into three fundamental architectural paradigms.

Standard Feedforward Architectures: Multi-Layer Perceptron (MLP) is the foundational feedforward architecture used for processing fixed-size statistical distributions or flattened feature vectors. An MLP consists of an input layer, multiple hidden layers, and an output layer. Mathematically, a single neuron j in layer l computes a weighted sum of its inputs $a_i^{[l-1]}$ followed by a non-linear activation function:

$$a_j^{[l]} = \alpha\left(z_j^{[l]}\right) = \alpha\left(\sum_i w_{ji}^{[l]} a_i^{[l-1]} + b_j^{[l]}\right), \tag{1}$$

where $z_j^{[l]}$ denotes the pre-activation value (i.e., the linear combination of inputs) of the j-th neuron in the l-th layer, while $w_{ji}^{[l]}$ and $b_j^{[l]}$ represent the weight connecting neuron i in the $l-1$ layer and the bias term, respectively. The term $a_j^{[l]}$ signifies the activation output resulting from the non-linear function $\alpha(\cdot)$. The output layer typically employs a softmax function to produce a probability distribution over N possible key candidates:

$$s(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{N} e^{z_k}}. \tag{2}$$

This standard architecture is heavily utilized in our framework for scenarios like SIFA, where the input is a highly abstracted, non-linear probability distribution. Moreover, when these feedforward networks are specifically configured for binary classification tasks to determine the mathematical validity of a given cryptographic state or differential, they function as Algebraic Verifiers. Such verifiers are instrumental in mapping discrete algebraic patterns to their corresponding feasibility within the cipher's transformation space.

Weight-Shared and Parallel Architectures: In classical block ciphers like Advanced Encryption Standard (AES), the internal state is structured as an array of independent bytes processed in parallel. Flattening such a state matrix into a single vector for a standard MLP destroys its spatial structure and unnecessarily inflates the parameter space. To address this, we employ weight-shared and parallel architectures. Given an input sequence of cryptographic features $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T]$, a weight-shared layer applies the exact same transformation f_θ, which is parameterized by weights θ, independently across all T spatial slices:

$$Y = [f_\theta(\mathbf{x}_1), f_\theta(\mathbf{x}_2), \ldots, f_\theta(\mathbf{x}_T)]. \tag{3}$$

This architectural design allows the network to process all bytes of a state matrix simultaneously. In our methodology, this enables the parallel generation of 16 independent sub-key probability distributions, significantly enhancing computational efficiency while preserving the structural integrity of the cipher's state.

Domain-Informed Neural Architectures: While deep learning excels at automated feature extraction, purely data-driven models may struggle to efficiently learn highly specific cryptographic constraints, such as permutation anomalies caused by persistent faults. Inspired by Physics-Informed Neural Networks (PINNs) [RPK19], we introduce Domain-Informed Architectures that embed mathematical priors and cryptographic rules directly into the network's topology.

By introducing customized, non-parametric layers before the trainable layers, the network explicitly incorporates algebraic relationships. For instance, in our PFA models, an alignment layer dynamically realigns a ciphertext frequency histogram $H \in \mathbb{R}^{256}$ based on a specific XOR fault differential Δ. The transformed histogram H' is mathematically defined as:

$$H'_i = H_{i \oplus \Delta} \quad \text{for } i \in \{0, 1, \ldots, 255\}. \tag{4}$$

Embedding such deterministic cryptographic operations directly into the computational graph guides the network's attention to the most relevant features, effectively bridging analytical cryptanalysis with deep learning and accelerating model convergence.

2.2 Classical Fault Analysis

To effectively apply deep learning to fault analysis, one must understand the underlying leakage mechanisms exploited by classical methods. DLFA essentially replaces the manual mathematical derivation and statistical hypothesis testing of these methods with data-driven learning.

Differential Fault Analysis: Differential Fault Analysis (DFA) [BS97] is arguably the most prominent fault analysis technique. It relies on injecting a fault during the encryption process and analyzing the difference between the correct ciphertext C and the faulty ciphertext C^*.

In the case of AES-128, a common attack targets the encryption process by injecting a fault before the MixColumns operation of the 8th or 9th round. For a single byte fault f induced at the input of the 9th round MixColumns, the fault diffuses to four bytes at the output of the round. The relationship between the input differential and the output differential is linear with respect to the MixColumns transformation but becomes non-linear after the SubBytes operation of the 10th round. The adversary calculates the ciphertext differential $\Delta C = C \oplus C^*$. Following the formulation by Piret and Quisquater [PQ03], for each active byte i in the 10th round state, the relationship involving the last round key K_{10} is given by:

$$\delta_i = S^{-1}(C_i \oplus K_{10,i}) \oplus S^{-1}(C_i^* \oplus K_{10,i}), \tag{5}$$

where δ_i is the known fault value propagated through the MixColumns inverse. By collecting a small number of pairs (C, C^*) (typically two are sufficient for AES-128), the adversary can solve this system of equations to recover K_{10}.

Statistical Fault Analysis: Statistical Fault Analysis (SFA) [FJLT13] was developed to address scenarios where the exact impact of the fault is unknown or where observing the correct ciphertext is impossible, which exploits the statistical bias induced by a fault. Fundamentally, a fault can be modeled as a transition of an intermediate variable x to a faulty value x'. This transition invariably disturbs the uniform distribution of the intermediate values. SFA typically utilizes the Squared Euclidean Imbalance (SEI) as a distinguisher. The SEI metric quantifies the distance between the observed distribution of the intermediate value (derived from decrypting the faulty ciphertext with a key guess) and the uniform distribution. The SEI for a key hypothesis $\hat{k}$ is defined as:

$$SEI(\hat{k}) = \sum_{v=0}^{2^s-1} \left(\frac{N_v(\hat{k})}{N} - \frac{1}{2^s} \right)^2, \tag{6}$$

where N is the total number of faulty ciphertexts, $N_v(\hat{k})$ is the count of occurrences where the intermediate value equals v under key hypothesis $\hat{k}$, and s is the bit-width of the variable (e.g., $s = 8$ for AES S-box). The correct key hypothesis yields the maximal SEI because it correctly decrypts the ciphertext to the biased intermediate state, whereas wrong key guesses behave like random permutations.

Statistical Ineffective Fault Analysis: Statistical Ineffective Fault Analysis (SIFA) [DEK+18] represents a paradigm shift in fault analysis by exploiting ineffective faults, where the injected fault fails to alter the computation result (e.g., a Stuck-at-0 fault but the bit is already '0'). While traditional countermeasures might suppress faulty outputs, they often release the correct ciphertext when no error is detected.

Note that a fault was ineffective leaks information about the intermediate value. If a Stuck-at-0 fault on bit b is ineffective, the attacker knows with certainty that bit b was '0'. Consequently, the distribution of the intermediate values for the subset of ineffective fault ciphertexts will be biased. Similar to SFA, SIFA utilizes statistical metrics like SEI or Chi-squared tests to identify the key guess that reveals this non-uniformity. SIFA is particularly powerful because it bypasses many redundancy-based countermeasures that only check for differential errors.

Persistent Fault Analysis: Persistent Fault Analysis (PFA) [ZLZ+18] operates on a distinct fault model where the fault is not transient but modifies the stored constants of the cryptographic algorithm, such as the S-box look-up table. Once injected, the fault persists across multiple encryptions until the device is reset or the table is refreshed.

In AES, if an entry in the S-box $S[x]$ is corrupted to $S^*[x]$, it creates a permutation anomaly. Specifically, suppose the original value v at $S[x]$ is overwritten by a faulty value v'. This leads to two inputs mapping to v', while the original output value v becomes impossible to generate. Consequently, the expected uniform probability distribution of the S-box output Y is disrupted into a biased faulty distribution $Pr[Y^* = y]$, which can be mathematically expressed as:

$$Pr[Y^* = y] = \begin{cases} 0, & \text{if } y = v \quad \text{(Impossible Value)} \\ \frac{2}{256}, & \text{if } y = v' \quad \text{(Faulty Value)} \\ \frac{1}{256}, & \text{otherwise} \end{cases}. \tag{7}$$

PFA exploits this explicit permutation anomaly by analyzing the distribution of ciphertext bytes. The adversary collects ciphertexts and identifies the value that never appears in the distribution. The absence of this value allows the adversary to shortlist possible keys.

2.3 Related Works

In 1991, Rivest first outlined the relationship between provable machine learning techniques and cryptanalysis in [Riv91]. Hospodar et al. investigated the application of machine learning techniques in SCA [HGM+11]. In recent years, more researchers have focused on deep learning techniques, such as MLP [MZVT15, MDM16] and CNN [PSK+18] to go with the trend in the machine learning community. These techniques have been increasingly adopted in SCA as an alternative to traditional template analysis (TA) [KPH+19, CDP17, BPS+20]. Compared to TA, deep learning-based SCA is more robust to noise and often achieves comparable or superior performance [Tim19].

However, the application of deep learning to fault analysis is comparatively nascent. Prior work has largely focused on peripheral tasks. For instance, Liu et al. [LWLX17] and Breier et al. [BHJ+18] explored using faults to induce misclassification in neural networks, rather than using neural networks for cryptanalysis. Saha et al. [SAB+23] applied deep learning for leakage assessment in fault attacks, identifying vulnerable points. The direct application of DL for key recovery from faulty ciphertexts was explicitly identified as a research gap. This paper addresses this gap, formalizing a framework that leverages the learning capabilities of neural networks to automate and enhance the efficiency of DFA, SFA, SIFA, and PFA.

3 Deep Learning-based Fault Analysis

In this section, we propose a unified framework for Deep Learning-based Fault Analysis (DLFA). Instead of relying on manual derivation of fault propagation equations, DLFA employs neural networks to learn the mapping between the fault-induced leakage features and the secret key.

3.1 Fault Model

To establish a versatile analysis framework, we propose a generalized fault model applicable across various attack scenarios. The fault model is characterized by the following properties:

- **Fault Position and Timing:** Faults can be induced either during the encryption process (e.g., transient faults in DFA/SFA) or before the execution (e.g., persistent faults in PFA). The target is typically the state register or the non-volatile memory storing S-box constants.
- **Fault Persistence:** The model accommodates transient faults, which affect only a single encryption, or persistent faults, which remain active across multiple encryptions until a system reset.
- **Adversary Capability:** We assume a profiled attack scenario where the adversary can manipulate a clone device to collect labeled training data (ciphertexts associated with known keys and faults). For the target device, the adversary is only required to collect faulty ciphertexts.

3.2 General Description of DLFA

The DLFA framework follows a unified methodology based on a divide-and-conquer strategy to resolve the complex task of key recovery. As shown in Fig. 1, the overall workflow is structured into four distinct stages: Injection, Preparation, Training, and Attack, mirroring the profiled attack paradigm commonly found in side-channel analysis but specifically adapted for the active fault induction domain.

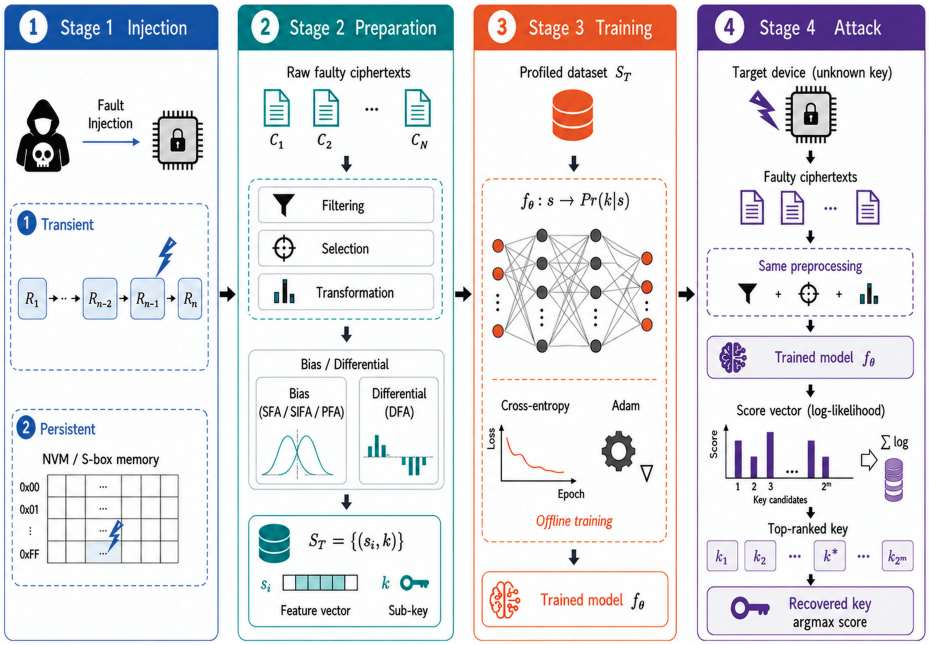


Figure 1: The attack framework of DLFA.

- **Stage 1: Injection:** In this stage, the adversary induces faults into the cryptographic implementation to create analytical leakage. For transient models such as DFA or SFA, faults are typically injected during the execution of the cipher, targeting the penultimate or final rounds to observe immediate computational perturbations. Conversely, for persistent models such as PFA, the fault is injected into the Non-Volatile Memory (NVM) storing S-box constants prior to the encryption phase.

- **Stage 2: Preparation:** The Preparation stage involves feature selection and data preprocessing, which are essential for transforming raw, high-dimensional faulty ciphertexts into a format optimized for neural network consumption. The adversary must extract pertinent features that maximize the correlation with the sub-key while minimizing noise. For statistical models like PFA and SFA, this involves characterizing the fault-induced bias. In contrast, for DFA, the features are constructed from differentials between correct and faulty ciphertext pairs. This stage results in a labeled dataset $\mathcal{S}_T$ consisting of pairs (s_i, k), where s_i represents the feature vector and k denotes the ground-truth sub-key label.

- **Stage 3: Training:** During the Training stage, the adversary selects an appropriate neural network architecture and trains it offline on the profiled dataset $\mathcal{S}_T$. The network seeks to learn a probabilistic function $f_\theta : s \to Pr(k|s)$ that maps input feature vectors to a probability distribution over the key space $\mathcal{K}$. The network parameters θ are updated iteratively using gradient-based optimization algorithms, such as the Adam optimizer, to minimize the categorical cross-entropy loss between the predicted distributions and the true key labels.

- **Stage 4: Attack:** Finally, the Attack stage involves performing fault injection on the target device where the secret key is unknown. A new set of faulty ciphertexts is collected and preprocessed into features using the identical methodology employed in the preparation stage. These samples are then fed into the trained network, which computes a score vector based on the accumulation of log-likelihoods for each key

candidate. The candidate achieving the maximum accumulated score is ultimately identified as the recovered secret key.

3.3 Feature Selection and Architectural Design

To demonstrate the versatility and efficiency of the DLFA framework, we apply it to four prominent fault models in AES-128. Rather than relying on a monolithic network, we tailor the feature engineering and architectural paradigms to the specific physical leakage characteristics of each fault type.

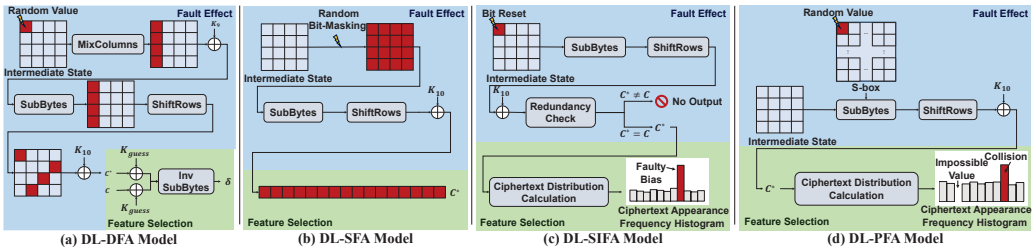


Figure 2: The diagram of fault effect and feature selection of different models.

3.3.1 DL-based Differential Fault Analysis (DL-DFA)

Fault Effect and Information Analysis: DFA is the classical methodology for transient fault analysis. As shown in Fig. 2(a), a *single-byte-random* fault is injected prior to the MixColumns operation in the ninth round. This fault propagates through the linear and non-linear layers of the cipher, resulting in a four-byte error pattern in the tenth-round state. Although the fault disrupts the state, its propagation follows strict algebraic laws within the Galois Field $GF(2^8)$. The core information lies in whether a specific ciphertext differential satisfies the linear equations of the AES MixColumns transformation.
Feature Selection and Rationale: The feature vector for DL-DFA is the differential value calculated after the inverse S-box operation, denoted as δ. Specifically, for a correct-faulty ciphertext pair (C, C^*) and a hypothesized key candidate K_{guess}, the feature is defined as $\delta = S^{-1}(C \oplus K_{guess}) \oplus S^{-1}(C^* \oplus K_{guess})$. We select this specific feature extraction method because it maps the computational perturbation directly back to the output of the MixColumns transformation. By extracting δ, we allow the network to bypass the severe non-linearity of the S-box, enabling it to focus exclusively on verifying the linear algebraic constraints imposed by the diffusion layer.
Network Architecture: We employ an Algebraic Verifier architecture. It utilizes an Embedding Layer to map the discrete bytes of δ into a dense vector space, followed by an MLP that acts as a distinguisher. This allows the model to learn the underlying $GF(2^8)$ properties and efficiently filter out key candidates that produce algebraically invalid differentials.

3.3.2 DL-based Statistical Fault Analysis (DL-SFA)

Fault Effect and Information Analysis: SFA relaxes the DFA requirement for correct-faulty ciphertext pairs by exploiting the statistical bias induced by faults. As shown in Fig. 2(b), in our AES implementation, the fault is a *random-bit-masking* operation applied simultaneously to the entire 16-byte intermediate state at the input of the tenth-round S-box. This wide-area fault forces bits across all state bytes toward zero, disrupting the expected uniform distribution and creating a biased 16-byte ciphertext distribution that is strongly correlated with the entire round key.

Feature Selection and Rationale: For DL-SFA, we extract the raw ciphertext bytes as the input features. We utilize this direct feature extraction method to strictly preserve the spatial correlation and the unaggregated statistical leakage of the entire 16-byte state. By feeding raw sequences to the network, we ensure no subtle multi-dimensional dependencies are lost, providing the model with the complete data necessary to autonomously learn and quantify the underlying statistical imbalance (e.g., the Squared Euclidean Imbalance).
Network Architecture: We implement a Weight-Shared Architecture using TimeDistributed layers. This architecture is specifically chosen because the fault affects all 16 bytes in parallel. By sharing weights across all 16 spatial bytes, the network learns a universal feature extractor for statistical bias. This parallel design significantly reduces the parameter count and enables the model to recover all 16 sub-key bytes simultaneously in a single forward pass.

3.3.3 DL-based Statistical Ineffective Fault Analysis (DL-SIFA)

Fault Effect and Information Analysis: SIFA targets implementations with redundancy-based countermeasures (e.g., Dual Modular Redundancy) by exploiting ineffective faults. As shown in Fig. 2(c), our SIFA model targets a single specific byte of the intermediate state after the ninth round. A *bit-reset* fault is induced on this specific byte, but the ciphertext is only released if the fault is ineffective (i.e., the bit was already zero), passing the redundancy check. This results in a safe-error condition where the surviving ciphertexts possess an extreme statistical bias in that targeted intermediate byte.
Feature Selection and Rationale: The features for DL-SIFA are the normalized frequency histograms of the surviving ciphertexts. Because the AES cipher operates on 8-bit bytes, any intermediate target byte can only take one of 256 discrete values ranging from 0 to 255. To capture the statistical bias, we construct a histogram containing exactly 256 slots, where each slot records the occurrence frequency of a specific byte value. We extract these histograms because SIFA relies entirely on identifying non-uniformity within a subset of correct computations. Aggregating these relatively rare ineffective fault occurrences into a 256-dimensional probability distribution creates a highly concentrated characteristics of the bias. This targeted extraction strips away irrelevant noise and allows the network to efficiently evaluate the statistical hypotheses for each key candidate.
Network Architecture: A Standard Deep MLP is utilized for SIFA. Given that the fault is isolated to a single byte and the input is a fixed-size 256-element statistical vector, a high-capacity feedforward network is perfectly suited to magnify the subtle non-uniformity in the surviving distribution.

3.3.4 DL-based Persistent Fault Analysis (DL-PFA)

Fault Effect and Information Analysis: : PFA is a highly efficient model where a fault persists in the S-box constants across multiple encryptions. As shown in Fig. 2(d), a *single-byte-random* fault is injected into the non-volatile memory (e.g., Flash or EEPROM) storing the S-box constants prior to encryption. This corruption creates a specific permutation anomaly: the original value at the corrupted index becomes an impossible value (i.e., probability 0), while the faulty value creates a collision (i.e., doubled probability).
Feature Selection and Rationale: We extract the ciphertext probability distribution as the primary feature, alongside an estimated fault offset parameter. Similar to SIFA, this distribution is structured as a 256-element vector, recording the appearance frequency of every possible byte value (from 0 to 255) observed in the faulty ciphertexts. This statistical representation is strictly utilized because the key-dependent leakage in PFA is encoded entirely in the altered frequencies of the ciphertext bytes. Specifically, this 256-element format perfectly captures the position of the impossible value (i.e., the specific

slot within the distribution that drops to a zero-count), which is the primary indicator for key recovery.

Network Architecture: We propose a Domain-Informed Architecture featuring a custom Histogram Alignment Layer followed by a 1D Convolutional Neural Network (CNN) and an MLP. This specialized architecture is designed to provide the network with additional domain knowledge, specifically the fault position and the fault value. As demonstrated in [ZZJ+20], both pieces of information can be accurately obtained through Maximum Likelihood Estimation (MLE). Following alignment, the 1D Convolutional layers act as highly optimized topological feature extractors, specifically scanning the distribution for localized probability impossible values and collisions. Finally, a funnel-shaped MLP interprets these extracted patterns to accurately predict the secret key, effectively accelerating model convergence on complex statistical distributions.

4 Experimental Evaluation

4.1 Experimental Setup

To validate the practical feasibility of the DLFA framework in real-world scenarios, we selected AES-128 as the primary evaluation target. The cryptographic algorithm was implemented on the SAKURA-G platform, a widely recognized standard FPGA board dedicated to hardware security research. Specifically, we synthesized a serial hardware architecture of AES-128. The serial design executes the state byte transformations over multiple clock cycles, which not only reflects resource-constrained IoT environments but also provides a wider temporal window for highly precise, targeted fault injections.

To benchmark the performance of the proposed DLFA framework, we selected the seminal and classical methodologies for each respective fault model (e.g., Piret and Quisquater's DFA [PQ03], Minimal mean Hamming weight-based SFA [FJLT13], SEI-based SIFA [DEK+18], and the original PFA [ZLZ+18]) as our comparative baselines. While highly specialized heuristic solvers and recent optimizations [XZY+21, BSR+22] exist for specific scenarios, these classical works establish the fundamental theoretical and data complexity limits for their respective attack vectors. By comparing DLFA against these canonical baselines, we aim to demonstrate a fundamental paradigm shift—proving that a unified, automated deep learning architecture can consistently surpass or match the theoretical limits established by decades of manual mathematical derivation, rather than merely evaluating marginal algorithmic improvements within the classical paradigm.

To accurately reproduce the diverse fault effects required by our models, we utilized distinct physical fault injection mechanisms tailored to each specific scenario. For transient fault models (DFA, SFA, and SIFA), we employed voltage fault injection. By introducing precisely timed, rapid voltage drops to the FPGA's core supply, we intentionally violated the setup and hold times of the flip-flops, successfully inducing the required single-byte, multi-byte, and bit-reset errors during the cipher's execution. Conversely, we applied the Rowhammer technique [KDK+14] to permanently corrupt the stored cryptographic constants for the persistent fault model. By repeatedly accessing specific DRAM rows where the AES S-box is loaded, we induced electrical charge leaks that caused unintended bit flips, mirroring the permutation anomalies required for PFA.

The experiments are conducted on an ordinary computer with the following configuration: an Intel Core i5-12400 CPU at 3.20 GHz, 32 GB of DDR5 memory, a 1 TB Samsung SSD-980-PRO drive, and an Nvidia GeForce RTX 3060 GPU. The development language used is Python (version 3.9.7), and the development environment is Jupyter Notebook (version 6.4.5). The deep learning neural network implementations are developed using the Keras library (version 2.9.0) as the frontend and TensorFlow (version 2.9.1) as the backend. CUDA (version 11.7) is employed to accelerate the training process.

4.2 Deep Learning Model Configurations

As detailed in Table 1, the neural network architectures are highly customized to accommodate the distinct feature representations and attack objectives of each fault model.

Table 1: Summary of Deep Learning Architectures for DLFA

Feature	DL-DFA	DL-SFA	DL-SIFA	DL-PFA
Input Shape	4-byte differential (δ)	16-byte raw ciphertext	256-element histogram	256-element hist + Offset
Primary Layer	Embedding ($256 \to 64$)	Embedding ($256 \to 128$)	Dense (Standard)	Custom Alignment Layer + Reshape
Hidden Layers	2 Dense Layers	2 Time-Distributed Dense	3 Dense Layers	2 Conv1D + 2 Dense Layers
Units per Layer	512, 256	512, 512	1024, 1024, 1024	32, 64 (Conv) / 512, 256 (Dense)
Activation	ReLU	ReLU	ReLU	ReLU
Regularization	BatchNorm, Dropout (0.2)	BatchNorm, Dropout (0.2)	BatchNorm, Dropout (0.1)	BatchNorm, Dropout (0.2)
Output Layer	Dense (1, Sigmoid)	$16 \times$ Dense (256, Softmax)	Dense (256, Softmax)	Dense (256, Softmax)
Optimizer	Adam	Adam	Adam	Adam
Loss Function	Binary Crossentropy	Sparse Categorical Crossentropy	Sparse Categorical Crossentropy	Sparse Categorical Crossentropy

For models processing discrete cryptographic bytes (i.e., DL-DFA and DL-SFA), we incorporate initial Embedding layers to capture the complex, non-linear algebraic properties of the Galois Field. DL-DFA functions as a binary algebraic verifier, utilizing Binary Crossentropy and a Sigmoid output to distinguish valid from invalid differentials. In contrast, DL-SFA employs a weight-shared topology featuring Time-Distributed Dense layers, outputting 16 parallel Softmax distributions to recover the entire cipher state simultaneously.

Conversely, for models processing statistical features (i.e., DL-SIFA and DL-PFA), the networks directly ingest 256-element histograms. DL-SIFA requires a high-capacity Standard MLP with three wide hidden layers to reliably magnify subtle statistical biases. DL-PFA, on the other hand, utilizes a Domain-Informed hybrid design, replacing the standard input layer with a Custom Alignment Layer to physically align the histogram based on the estimated fault offset. This is subsequently processed by two 1D Convolutional layers to extract local topological anomalies, and finally fed into a funnel-shaped MLP for subsequent classification.

To mitigate overfitting and ensure robust generalization against the inherent noise of physical fault injections, all models integrate Batch Normalization and Dropout layers, alongside standard non-linear activation functions.

4.3 Evaluation Metrics

To quantitatively assess the efficiency and robustness of the proposed DLFA framework, we adopt Guessing Entropy (GE) and Success Rate (SR), which are widely utilized in the side-channel and fault analysis community, as the evaluation metrics.

Guessing Entropy measures the average computational effort required by an adversary to brute-force the remaining key candidates after applying the analytical model. Formally, for a given sub-key search space $\mathcal{K}$, let $S(k)$ denote the accumulated likelihood score output by the neural network for a specific candidate $k \in \mathcal{K}$ over a set of injected faults. The rank of the correct secret key k^*, denoted as $\mathrm{Rank}(k^*)$, is mathematically defined by the number of false candidates that yield a score strictly greater than that of the correct key, which can be expressed as:

$$\mathrm{Rank}(k^*) = 1 + \left|\{k \in \mathcal{K} \mid S(k) > S(k^*)\}\right|, \tag{8}$$

where $|\cdot|$ denotes the cardinality of the set. Thus, a rank of 1 implies that $S(k^*)$ is the absolute maximum among all candidates. Given N independent attack experiments, the GE is defined as the expected rank over all N experiments:

$$GE = \frac{1}{N}\sum_{i=1}^{N} \mathrm{Rank}_i(k^*). \tag{9}$$

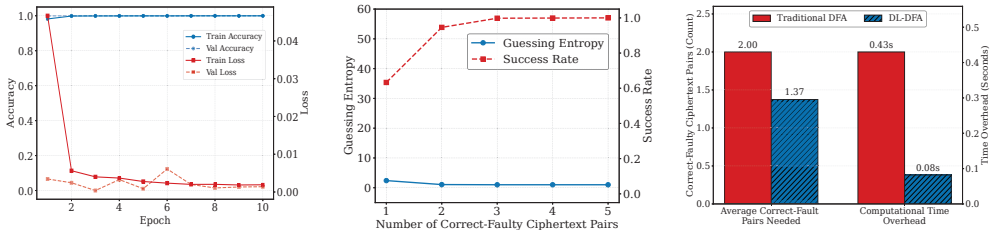
A GE converging to 1 indicates that the correct key is consistently ranked first, meaning the attack has fully converged and the key is recovered without any additional exhaustive search.

While GE provides a continuous measure of the attack's convergence trend, the Success Rate evaluates the absolute success probability of the attack. It is defined as the empirical probability that the correct sub-key k^* is ranked exactly first (i.e., $\mathrm{Rank}(k^*) = 1$) among all candidates. Using the same notation for N independent attacks, the SR can be mathematically expressed using the indicator function $\mathbb{I}(\cdot)$, which equals 1 if the condition is true and 0 otherwise:

$$SR = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(\mathrm{Rank}_i(k^*) = 1). \tag{10}$$

4.4 Evaluation of DL-DFA

We first evaluate the performance of DL-DFA. The objective is to verify whether the deep learning model can autonomously learn the linear constraints of the AES diffusion layer and successfully utilize them to recover the correct sub-key.



(a) Training Performance. (b) Attack Efficiency. (c) Comparative Analysis.

Figure 3: Evaluation Results of DL-DFA.

Training Performance: As shown in Fig. 3(a), the training and validation metrics over 10 epochs highlight the learning capability of the embedding-based Algebraic Verifier. The model exhibits rapid convergence. Within the first epoch, both the training and validation accuracies approach 100%, while the loss metrics plummet to near zero. This convergence indicates that the Embedding layer effectively projects the discrete, non-linear cryptographic bytes into a continuous space, allowing the subsequent MLP layers to effortlessly map the valid MixColumns differential properties within $GF(2^8)$. The stable, overlapping curves of the training and validation sets further confirm the absence of overfitting, ensuring the model's robustness against noise.

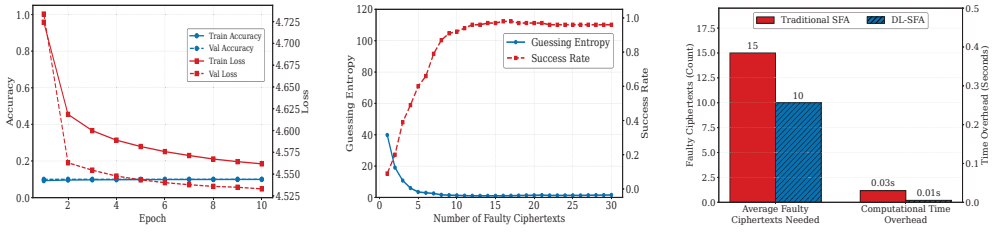
Attack Efficiency: The experimental results shown in Fig. 3(b) demonstrate a highly efficient key recovery process. With only a single correct-faulty ciphertext pair, the model is already capable of drastically reducing the search space, bringing the GE down to approximately 2.5 and achieving an SR of roughly 60%. When the adversary provides exactly 2 correct-faulty ciphertext pairs, the GE converges perfectly to 1, indicating that the correct key is ranked first on average.

Comparative Analysis: To benchmark our approach, we quantitatively compared DL-DFA against traditional DFA [PQ03] over 1000 independent physical experiments. As illustrated in Fig. 3(c), traditional DFA strictly requires two correct-faulty ciphertext

pairs to mathematically resolve the non-linear equations. Remarkably, DL-DFA reduces this data complexity to just 1.37 pairs. This demonstrates that the Algebraic Verifier can extract sufficient analytical confidence from a single fault pair to uniquely identify the correct key, surpassing traditional theoretical limits. Furthermore, while traditional DFA relies on sequential heuristic solving that consumes 0.43 seconds per attack, DL-DFA leverages highly parallelized GPU matrix multiplications to slash the execution time to a mere 0.08 seconds. These empirical results firmly validate that our comprehensive attack framework fundamentally lowers both the physical injection threshold and the computational overhead.

4.5 Evaluation of DL-SFA

Then, we evaluate the performance of DL-SFA. The objective is to determine whether the network can autonomously extract and amplify subtle, multi-byte statistical biases from raw faulty ciphertexts to recover the secret key.



(a) Training Performance. (b) Attack Efficiency. (c) Comparative Analysis.

Figure 4: Evaluation Results of DL-SFA.

Training Performance: Unlike the deterministic algebraic nature of DFA, SFA relies on exploiting weak statistical imbalances (e.g., AND masking biases) embedded within noisy physical measurements. This fundamental difference is reflected in the training metrics. As shown in Fig. 4(a), the validation accuracy plateaus at approximately 10%. While seemingly low for a standard classification task, this is actually a strong indicator of successful learning: a single raw ciphertext only provides a fractional statistical hint about the key, yet 10% is astronomically higher than the random guessing baseline of a single byte ($1/256 \approx 0.39\%$). Furthermore, the smooth, monotonic decrease in validation loss from 4.725 to 4.525 over 10 epochs confirms that the weight-shared network successfully captures the universal statistical distribution of the fault leakage without overfitting to the physical noise.

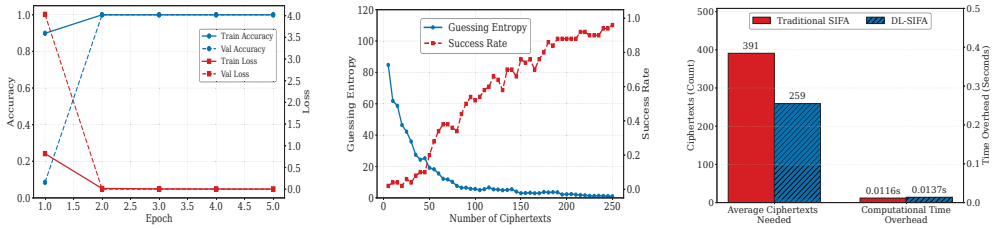
Attack Efficiency: To evaluate the practical key recovery efficiency, we utilize the trained network to process new batches of faulty ciphertexts during the attack phase. Because a single trace only provides a weak bias, the network accumulates likelihood scores across multiple traces. The experimental results in Fig. 4(b) illustrate a statistical convergence: as the number of available traces increases, the weak single-trace signals are rapidly amplified. The GE drops precipitously from near 40 down to 1, while the SR climbs steeply. The network is capable of reliably achieving an SR of nearly 100% and a GE of 1 with approximately 10 to 15 faulty ciphertexts, successfully recovering all 16 sub-key bytes simultaneously.

Comparative Analysis: To benchmark efficiency, we compared DL-SFA against traditional SFA [FJLT13], which manually calculates the Minimal mean Hamming weight, over 1,000 experiments. As Fig. 4(c) illustrates, traditional SFA requires an average of 15 faulty ciphertexts, while DL-SFA achieves recovery with only 10 traces, marking a 33% reduction in data complexity. More impressively, benefiting from highly parallelized GPU matrix inference, DL-SFA not only significantly reduces the physical fault acquisition cost but

also outperforms traditional methods in computational time overhead (0.01 s vs. 0.03 s). These results demonstrate that our deep learning attack framework achieves superiority over traditional cryptanalysis in both data efficiency and execution speed.

4.6 Evaluation of DL-SIFA

In this subsection, we evaluate the performance of the Standard Deep MLP architecture tailored for DL-SIFA. The objective is to verify the network's ability to extract and magnify the highly sparse statistical biases hidden within the normalized frequency histograms of surviving ciphertexts.



(a) Training Performance. (b) Attack Efficiency. (c) Comparative Analysis.

Figure 5: Evaluation Results of DL-SIFA.

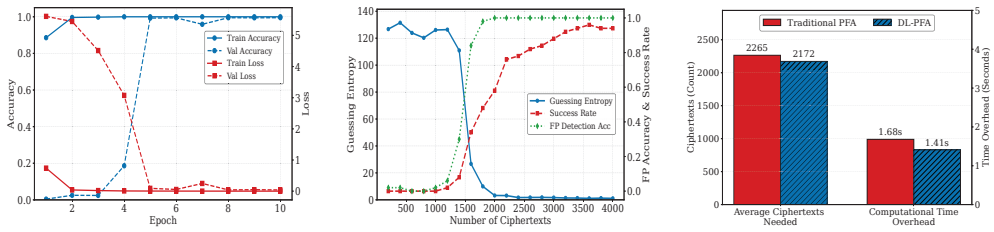
Training Performance: Fig. 5(a) demonstrates the effectiveness of transforming raw ineffective faults into structured 256-element frequency histograms. As depicted in the training metrics, the validation accuracy starts at approximately 35% but experiences a rapid convergence to 100% by the second epoch. Concurrently, the validation loss drops abruptly from over 4.0 to near zero. This convergence proves that once the rare surviving traces are aggregated into a structured probability distribution, a high-capacity feedforward network can effortlessly isolate the key-dependent safe-error fingerprint without succumbing to background noise.

Attack Efficiency: Evaluating the practical key recovery efficiency reveals the inherent physical challenges of ineffective fault analysis. Because the attacker must filter out and utilize only the ciphertexts that survive the hardware redundancy checks, SIFA fundamentally demands a larger data pool than DFA or SFA. Nevertheless, the experimental results in Fig. 5(b) show a stable convergence process. As the number of collected surviving ciphertexts increases toward 250, the GE smoothly descends from over 80 down to 1, while the SR climbs steadily, reaching nearly 100%. The deep learning model successfully navigates the data sparsity, consistently recovering the targeted key byte within approximately 200 to 250 ciphertexts.

Comparative Analysis: To quantify the efficiency gain, we benchmarked DL-SIFA against traditional SIFA methodologies [DEK+18] over 1,000 independent physical attack experiments. Traditional SIFA requires an average of 391 surviving ciphertexts to reliably construct a distinguishable statistical hypothesis. In stark contrast, DL-SIFA significantly reduces this data complexity by approximately 34%, requiring an average of only 259 ciphertexts. Regarding computational execution, traditional SIFA requires 0.0116 seconds, whereas the deep MLP executes in 0.0137 seconds. This fractional overhead of approximately 2 milliseconds is negligible in the hardware security evaluations, where the precise physical induction and collection of hundreds of safe-errors is exceptionally time-consuming. These findings confirm that DL-SIFA provides a massive optimization in overcoming physical data acquisition barriers.

4.7 Evaluation of DL-PFA

Finally, we evaluate the Domain-Informed hybrid architecture designed for DL-PFA. The primary objective is to demonstrate how embedding cryptographic prior knowledge enables the deep learning model to accurately identify permutation anomalies (i.e., impossible values) caused by persistent faults.



(a) Training Performance. (b) Attack Efficiency. (c) Comparative Analysis.

Figure 6: Evaluation Results of DL-PFA.

Training Performance: As depicted in Fig. 6(a), DL-PFA model exhibits a distinct two-stage convergence process. During the initial four epochs, the validation accuracy remains low, indicating that the network is adjusting its weights to process the 256-element histograms. However, at the 5th epoch, the validation accuracy experiences a sharp leap to nearly 100%, accompanied by a rapid drop in validation loss to near zero. Following a minor fluctuation at the 7th epoch, both metrics stabilize. This concrete learning curve demonstrates that once the 1D Convolutional layers successfully locate the core statistical anomalies (i.e., the impossible values and collisions) within the distributions, the hybrid network achieves rapid, optimal convergence without overfitting to the physical measurement noise.

Attack Efficiency: The practical attack efficiency of PFA heavily relies on the precision of the initial Fault Position (FP) estimation. The experimental results in Fig. 6(b) clearly illustrate this dependency. The FP Detection Accuracy (green dotted line) acts as the critical prerequisite, requiring the accumulation of approximately 1,800 ciphertexts for the Maximum Likelihood Estimation to consistently achieve 100% accuracy. Once this physical parameter is accurately anchored, the GE drops precipitously, converging to 1 around 2,200 to 2,400 ciphertexts. Concurrently, the SR steadily climbs, achieving near-perfect recovery around 3,500 to 4,000 traces.

Comparative Analysis: To provide a holistic benchmark, we compared DL-PFA against traditional mathematical PFA [ZZJ+20] over 1,000 physical experiments. As illustrated in Fig. 6(c), traditional PFA requires an average of 2,265 ciphertexts per attack. Remarkably, DL-PFA improves this data efficiency, successfully reducing the required ciphertexts to an average of only 2,172 (an approximate 4.1% reduction). More importantly, the deep learning framework demonstrates an advantage in execution speed. While traditional PFA consumes 1.68 seconds per attack due to sequential heuristic solving, DL-PFA leverages optimized 1D Convolutional feature extraction and highly parallelized GPU inference to reduce the execution time to 1.41 seconds. The experimental results establish that our framework achieves superiority, universally reducing both the data acquisition requirements and the computational time overhead across all evaluated fault models.

4.8 Robustness against Physical Noise

Traditional cryptanalytic methods are mathematically rigorous, but this rigor is heavily predicated on highly idealized fault assumptions. Classical algebraic and statistical solvers strictly demand pristine, noise-free fault data to converge. However, in practical hardware

evaluation environments, physical fault injection (e.g., voltage glitching or electromagnetic induction) is an inherently stochastic process [BBKN12]. Due to factors such as target clock jitter, thermal drift, and parasitic capacitance, adversaries cannot guarantee the precision of the fault injection time window, nor can they perfectly control the resulting fault effects to exactly match theoretical models (e.g., inducing a strict single-byte error) [MDH+14]. When a physical glitch inadvertently causes a multi-byte disruption or misses the critical temporal window, traditional mathematical equations typically become unsolvable or yield false positives. In contrast, deep learning methods inherently reframe cryptanalysis as a statistical pattern recognition task. By projecting complex, noisy physical leakages into a high-dimensional continuous feature space, neural networks focus on capturing the underlying data distributions and implicit cryptographic features rather than rigidly enforcing brittle algebraic rules. Consequently, deep learning classifiers exhibit a natural tolerance to partial data corruption and measurement artifacts. They function as advanced non-linear filters, effectively mitigating physical noise and extracting the surviving valid fault signals even from imperfect injections.

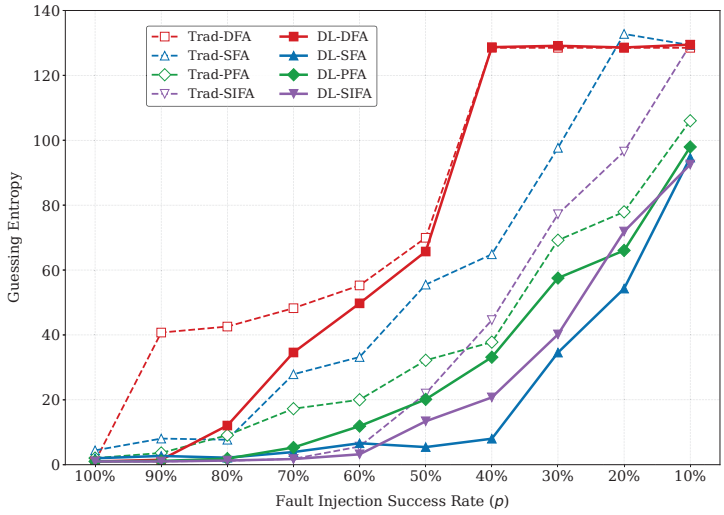
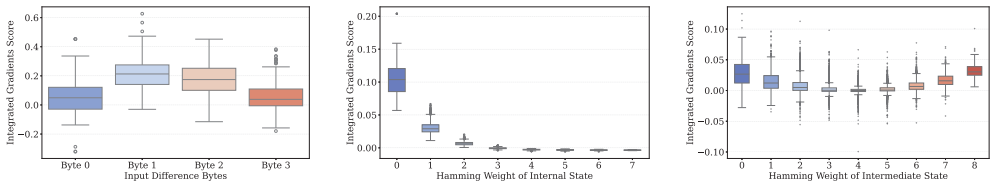


Figure 7: Evaluation Results with different Fault Injection Success Rate p.

To quantitatively evaluate the resilience of our framework against these real-world imperfections, we introduced the Fault Injection Success Rate (p) as a variable parameter, representing the proportion of ideal faults versus physical noise within the collected datasets. As illustrated in Fig. 7, under idealized conditions ($p = 100\%$), both paradigms successfully recover the key. However, as the injection success rate deteriorates, the performance of traditional methodologies experiences analytical failures. Traditional DFA, in particular, exhibits an surge in GE as soon as p drops below 90%, rendering the attack practically infeasible under moderate noise. In contrast, the proposed DLFA framework exhibits sustained analytical stability. Benefiting from the inherent generalization and noise-suppression capabilities of deep neural networks, our models manage to maintain a relatively low GE even when the effective fault rate drops to the extreme bounds of 40% to 50%. This robust performance establishes that our comprehensive attack framework relaxes the stringent requirements for ultra-precise physical injection equipment, rendering it superior for real-world hardware security evaluations.

5 Interpretability of DLFA Framework

While the empirical results in Section 4 establish the superior key recovery efficiency and robust noise tolerance of our DLFA framework, deep learning models are frequently criticized for their inherent black-box nature. In the context of hardware security evaluation, achieving a high success rate is insufficient; it is equally critical to prove that the neural networks are capturing authentic cryptographic leakages rather than merely overfitting to device-specific noise or experimental artifacts. To verify that the neural networks have autonomously learned physically meaningful cryptographic properties, we employ Integrated Gradients (IG) [STY17], an axiomatic Explainable AI (XAI) technique, to conduct quantitative attribution analysis. Unlike heuristic visualization methods, IG evaluates feature importance by calculating the integral of gradients along a straight path from a neutral baseline to the actual input tensor. By applying IG, we can quantify the attribution scores across a diverse spectrum of input features, ranging from ciphertext differentials and statistical histograms to the Hamming weights (HW) of intermediate leakages, thereby establishing an objective basis for evaluating the model's interpretability.



(a) Attribution Analysis of DL-DFA.

(b) Attribution Analysis of DL-SFA.

(c) Attribution Analysis of DL-SIFA.

Figure 8: Attribution Analysis Results of DL-DFA, DL-SFA and DLSIFA based on Integrated Gradients.

As shown in Fig. 8(a), the IG score distribution across the input difference bytes indicates that the DL-DFA model does not treat the four active bytes of the MixColumns output as equivalent generic features. The neural network autonomously assigns significantly higher median IG scores to Byte 1 and Byte 2 compared to Byte 0 and Byte 3. This distinct asymmetry reveals that the network successfully learns and exploits the algebraic structural imbalances within the AES diffusion layer and the Maximum Distance Separable (MDS) matrix. The Algebraic Verifier dynamically allocates higher attribution weights to specific byte differences that yield greater algebraic information entropy, thereby mathematically accelerating the filtration of candidate keys.

As illustrated in Fig. 8(b), the IG scores concerning the HW of the internal state exhibit a severe skew toward low HW values, characterized by an exponential decline. The highest attribution weights are heavily concentrated at HW = 0. This phenomenon aligns perfectly with the AND-mask fault model of SFA (i.e., forcing specific bits to zero), which imprints a strong zero-tendency statistical bias on the underlying data. By assigning exceptionally high IG scores to low HW, it is evident that DL-SFA precisely anchors onto the highly distinguishable statistical imbalances inherently induced by AND-mask faults.

Fig. 8(c) shows that the IG scores for DL-SIFA regarding the HW of the intermediate state display a characteristic U-shaped distribution, being high at the extremes and low in the middle. This feature exposes the model's mastery of the statistical essence of SIFA. For a random 8-bit byte, the HW inherently follows a binomial distribution, with the probability density peaking near HW = 4. Consequently, at HW = 4, there is a massive sample overlap between the true state distribution of the correct key and the random distribution of incorrect keys, resulting in an extremely low Signal-to-Noise Ratio (SNR). Conversely, at the extreme positions of HW = 0 and HW = 8, the natural

probability is minimal, making the safe-error bias carried by the surviving ciphertexts highly prominent. By autonomously assigning the highest IG scores to these extreme HWs, the network objectively demonstrates that it has acquired the optimal strategy for extracting cryptographic SNR from the binomial distribution.

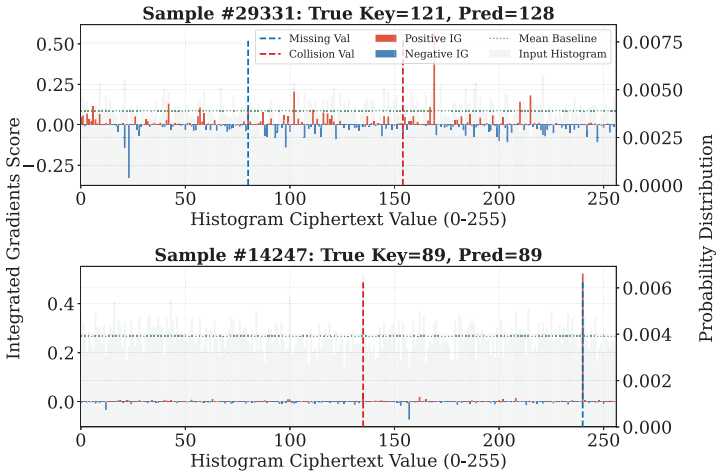


Figure 9: Attribution Analysis Results of DL-PFA based on Integrated Gradients.

The IG distribution of DL-PFA on normalized histograms in Fig. 9 provides transparency into the network's fault identification logic. In samples where the key is successfully recovered, the network generates sharp positive IG spikes that perfectly intersect with the theoretically expected missing value and collision value. This proves that, facilitated by the Custom Alignment Layer, the network comprehends the core prior knowledge of persistent faults: utilizing absolute statistical characteristics to deduce cryptographic permutation anomalies. Furthermore, in mispredicted samples, the IG mapping visually exposes the cause of the model's failure: severe noise fabricated a statistical valley resembling a missing value in an incorrect interval, thereby misleading the network's decision.

6 Conclusion

This paper introduced DLFA, a comprehensive deep learning framework designed to automate and optimize fault analysis against block ciphers. By decomposing the cryptanalytic process into tailored feature engineering and specialized neural network classification, we successfully unified four distinct fault models (DFA, SFA, SIFA, and PFA) under a singular, data-driven paradigm. Our physical evaluations on AES-128 demonstrated that DLFA substantially mitigates the rigid limitations of classical cryptanalysis. The framework universally reduces the data complexity across all evaluated fault models, extracting sufficient analytical confidence from exceptionally sparse data. We proved that deep learning classifiers function as advanced non-linear filters, maintaining sustained analytical stability and successfully recovering keys even when physical injection noise causes traditional algebraic solvers to fail. Furthermore, the application of Integrated Gradients provided interpretability, confirming that the DLFA architectures intrinsically learn valid cryptographic structures rather than overfitting to stochastic noise.

Looking forward, the architectural flexibility and noise-tolerance of the DLFA framework open promising avenues for future research. A primary focus will be extending this methodology to evaluate the physical vulnerability of Post-Quantum Cryptography (PQC) algorithms [RCDB24]. As cryptographic implementations grow increasingly complex,

shifting from manual mathematical derivation to deep learning-driven pattern recognition will become an indispensable strategy for next-generation hardware security assessments.

References

[BBKN12] Alessandro Barenghi, Luca Breveglieri, Israel Koren, and David Naccache. Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures. *Proc. IEEE*, 100(11):3056–3076, 2012.

[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.

[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.

[BSR+22] Nasour Bagheri, Sadegh Sadeghi, Prasanna Ravi, Shivam Bhasin, and Hadi Soleimany. SIPFA: statistical ineffective persistent faults analysis on feistel ciphers. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(3):367–390, 2022.

[CDP17] 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.

[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.

[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.

[KDK+14] Yoongu Kim, Ross Daly, Jeremie S. Kim, Chris Fallin, Ji-Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. In *ACM/IEEE ISCA 2014*, pages 361–372. IEEE Computer Society, 2014.

[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.

[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.

[MDH+14] Nicolas Moro, Amine Dehbaoui, Karine Heydemann, Bruno Robisson, and Emmanuelle Encrenaz. Electromagnetic fault injection: towards a fault model on a 32-bit microcontroller. *CoRR*, abs/1402.6421, 2014.

[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.

[MPP16] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. In Claude Carlet, M. Anwar Hasan, and Vishal Saraswat, editors, *Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings*, Lecture Notes in Computer Science, pages 3–26. Springer, 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.

[PHJ+19] Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni. The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(1):209–237, 2019.

[PPM+23] Stjepan Picek, Guilherme Perin, Luca Mariot, Lichao Wu, and Lejla Batina. Sok: Deep learning-based physical side-channel analysis. *ACM Comput. Surv.*, 55(11):227:1–227:35, 2023.

[PQ03] Gilles Piret and Jean-Jacques Quisquater. A differential fault attack technique against SPN structures, with application to the AES and KHAZAD. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*, Lecture Notes in Computer Science, pages 77–88. Springer, 2003.

[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.

[RCDB24] Prasanna Ravi, Anupam Chattopadhyay, Jan-Pieter D'Anvers, and Anubhab Baksi. Side-channel and fault-injection attacks over lattice-based post-quantum schemes (kyber, dilithium): Survey and new results. *ACM Trans. Embed. Comput. Syst.*, 23(2):35:1–35:54, 2024.

[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.

[RPK19] Maziar Raissi, Paris Perdikaris, and George E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.*, 378:686–707, 2019.

[SAB+23] Sayandeep Saha, Manaar Alam, Arnab Bag, Debdeep Mukhopadhyay, and Pallab Dasgupta. Learn from your faults: Leakage assessment in fault attacks using deep learning. *J. Cryptol.*, 36(3):19, 2023.

[STY17] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, Proceedings of Machine Learning Research, pages 3319–3328. PMLR, 2017.

[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.

[ZBHV20] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Methodology for efficient CNN architectures in profiling attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1):1–36, 2020.

[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.

[ZZJ+20] Fan Zhang, Yiran Zhang, Huilong Jiang, Xiang Zhu, Shivam Bhasin, Xinjie Zhao, Zhe Liu, Dawu Gu, and Kui Ren. Persistent fault attack in practice. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(2):172–195, 2020.