Recovering S-Box Design Structures and Quantifying Distances between S-Boxes using Deep Learning

Donggeun Kwon[1], Deukjo Hong[2], Jaechul Sung[3], and Seokhie Hong[1]

[1] School of Cybersecurity, Korea University, Republic of Korea
donggeun.kwon@gmail.com, shhong@korea.ac.kr
[2] Smart Grid Research Center, Department of Computer Sciences and Artificial Intelligence, Jeonbuk National University, Republic of Korea
deukjo.hong@jbnu.ac.kr
[3] Department of Mathematics, University of Seoul, Republic of Korea
jcsung@uos.ac.kr

Abstract. At ASIACRYPT'19, Bonnetain et al. demonstrated that an S-box can be distinguished from a permutation chosen uniformly at random by quantifying the distances between their behaviors. In this study, we extend this approach by proposing a deep learning-based method to quantify distances between two different S-boxes and evaluate similarities in their design structures. First, we introduce a deep learning-based framework that trains a neural network model to recover the design structure of a given S-box based on its cryptographic table. We then interpret the decision-making process of our trained model to analyze which coefficients in the table play significant roles in identifying S-box structures. Additionally, we investigate the inference results of our model across various scenarios to evaluate its generalization capabilities. Building upon these insights, we propose a novel approach to quantify distances between structurally different S-boxes. Our method effectively assesses structural similarities by embedding S-boxes using the deep learning model and measuring the distances between their embedding vectors. Furthermore, experimental results confirm that this approach is also applicable to structures that the model has never seen during training. Our findings demonstrate that deep learning can reveal the underlying structural similarities between S-boxes, highlighting its potential as a powerful tool for S-box reverse-engineering.

Keywords: S-box reverse-engineering · Design structure · Cryptographic tables · Quantifying distances · Deep learning.

1 Introduction

In modern cryptography, block ciphers are designed based on Kerckhoffs' principle [53], which assumes that all aspects of the cipher's specification and design rationale, except for the secret key, are publicly accessible. This principle emphasizes transparency, asserting that cryptographic algorithms should be open to foster trust and enable rigorous scrutiny and cryptanalysis by the community.

Despite this principle, there are several cases where the designers exploit secret structures to construct S-boxes, intentionally hiding these structures or even presenting misleading design rationales. Historically, several standard encryption algorithms, including DES [21], Skipjack [2], Streebog [23], and Kuznyechik [24], have utilized secretly designed S-boxes. When an S-box is provided solely as a lookup table, even with the full cipher specification disclosed, recovering the underlying design rationale of the S-box is a challenging task. In this context, research focused on recovering the hidden design structures behind an S-box becomes indispensable.

S-box reverse-engineering is a critical technique that aims to recover the methodologies used in the construction of an S-box from its lookup table, with the goal of understanding the design rationale intended by its designers [16]. It is of significant importance in cryptanalysis, as it can reveal underlying potential weaknesses and backdoors within the S-box [7,46]. It can also contribute to optimizing implementations. For example, replacing a monolithic lookup table with multiple smaller ones can reduce memory usage, and implementing a bitslicing approach can enhance execution efficiency by computing in parallel [11, 36].

Related Work and Motivation. The concept of anomalies of an S-box was introduced to quantify the distance between the behavior of a given S-box and those of an S-box chosen uniformly at random [16, 19, 46]. These distances are calculated based on the distribution of maximum coefficients in a table such as the Difference Distribution Table (DDT) and the Linear Approximation Table (LAT), reflecting how good the cryptographic properties are. If the value of anomalies for a given S-box significantly differs from that of random S-boxes, it can be inferred that the S-box is not picked at random.

However, since the anomalies consider only the maximum value and its number of occurrences in the tables, they overlook other factors, such as positional information and non-maximum coefficients. This loss of information could be detrimental when an attacker tries to identify S-box design structures. For example, Fig. 1 displays the distribution of the value of the coefficients of the LAT (excluding the first row and column) and the linear anomaly of each S-box. Since the anomaly only reflects the maximum coefficient value of the distribution, ICEBERG's anomaly value is closer to Zorro's than CLEFIA's, even though CLEFIA S_0 is also constructed with SPN. However, a more detailed examination of the lower-magnitude coefficients reveals that Zorro's occurrence distribution does not decrease as smoothly as ICEBERG's, thereby enabling a clear distinction between these two S-boxes. This observation demonstrates the limitations of relying solely on the anomaly to quantify the distance between S-boxes or identify their design structures, and highlights the importance of other coefficient information as additional evidence.

Biryukov et al. proposed a method, called Pollock's pattern recognition [16], to distinguish an S-box from a random permutation and recover its hidden design structure. This method employs a heuristic approach to identify non-random patterns within the visual representation of the cryptographic table of an S-box.

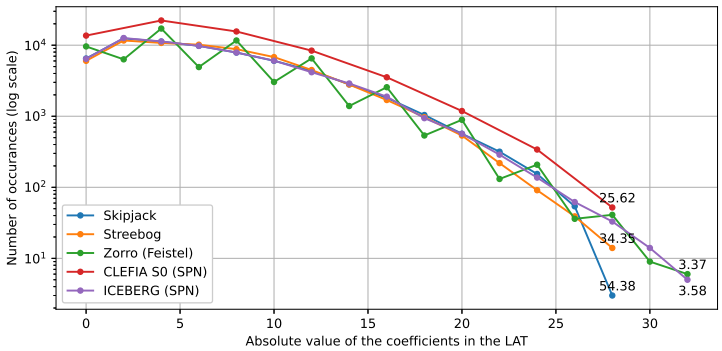


Fig. 1. Coefficients of the LAT and linear anomaly of each S-box.

Since the method relies on human visual and cognitive abilities, its effectiveness depends on an attacker's expertise. We believe that applying deep learning techniques to this pattern recognition can overcome the limitations of human vision. In this study, we present a new approach based on deep learning to extract patterns that are difficult to identify from cryptographic tables.

Our Contribution. In this study, we propose deep learning-based approaches for recovering S-box design structures and quantifying the distance between two distinct S-boxes. First, we introduce a deep learning-based framework for recovering S-box design structures with cryptographic tables. Our experimental results demonstrate that using the framework, our model can predict S-box structures with high accuracy and can be effectively applied to new structures without prior knowledge. Additionally, by analyzing which patterns within the tables contribute to the model's decision, we confirm that the model learns not only the differences between design structure classes but also their similarities. This indicates that our model learns the characteristics of the design structure, rather than merely patterns for classification.

We also experimentally validate the robustness of the deep learning model in recovering S-box structures across various scenarios. Our results show that the model provides valid outcomes for S-boxes used in the wild and also for S-boxes composed of modified structures based on the trained structures. For example, our model successfully identifies both Feistel and Misty structures in S-boxes that incorporate these designs. These results demonstrate the deep learning model's generalization capability to untrained structures. Furthermore, the model can distinguish the number of rounds in S-boxes with untrained structures, highlighting the potential of deep learning in quantifying distances between S-boxes.

Building on these results, we propose a new method to quantify distances between two different S-boxes using embedding vectors extracted from the neural network. By employing metrics such as Euclidean distance and cosine distance in the embedding space, we quantitatively evaluate the similarities between S-

boxes. Our experiments show that similar design structures are positioned closely in the embedding space, while distinct structures are clearly separated. Additionally, by applying the concept of anomaly detection through Support Vector Data Description, we propose a method to measure how far a given S-box lies from a specific structure. Our results demonstrate that the model can identify untrained structures that are similar to trained ones, further confirming the model's capability in capturing structural similarities. These findings underscore that deep learning-based approaches are effective in quantifying structural similarities between S-boxes and can serve as a useful tool in S-box reverse-engineering.

2 Preliminaries

2.1 S-Box and its Structure

An S-box (Substitution-box) is a crucial component of symmetric-key cryptography, performing non-linear operations by substituting each input value with a corresponding output value. It plays a vital role in ensuring the security of modern encryption algorithms, as its presence is essential to achieve cryptographic properties. One notable approach to designing an S-box, introduced in [44], involves borrowing the structure of block ciphers to construct S-boxes from smaller ones. Examples of these constructions can be found in Fig. 2. This methodology remains widely used in the design of S-boxes to this day. The Feistel and Misty structures, validated in block cipher design, underpin the S-boxes used in Zorro [25], Robin [26], and CRYPTON 0.5 [41], and usually consist of 3 to 4 rounds.

An additional block cipher-based design, known as the Substitution-Permutation Network (SPN) structure, is also widely used. We introduce a variation of SPN, denoted Sbp, which employs a bitwise permutation for its permutation layer. In this study, bitwise permutations are limited to permutations that shuffle at least one bit from both the left and right sides. While we refer to the QARMA S-boxes [4] for Sbp, we define a bpSbp structure based on the Midori S-boxes [6]. Block ciphers such as CLEFIA [54], Khazad [9], and ICEBERG [56] typically employ up to 3 rounds in the construction of their S-boxes.

In addition, the Lai-Massey structure [39] and its derivative, the Bridge structure [13], are also used to construct S-boxes. There is another approach that does not adopt block cipher structures, such as a finite field inverse-based S-box. Although inversion-based S-boxes are known for their strong security advantages, they come with the downside of higher implementation costs than block cipher-based S-boxes [27,35]. This approach is widely used in ciphers such as AES [22], ARIA [38], and Camellia [3].

2.2 Cryptographic Tables

The cryptographic characteristics of an S-box can be described using $2^n \times 2^n$ tables. These tables are essential for understanding the cryptographic strength

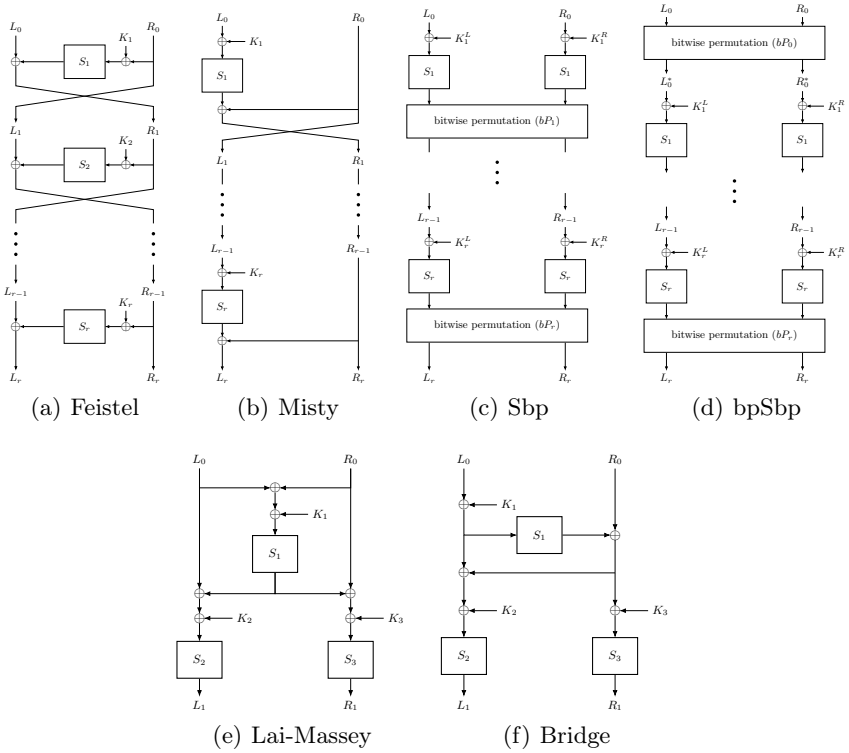


Fig. 2. Block cipher-based S-box construction methods.

of S-boxes and provide insight to assess potential vulnerabilities from different types of cryptanalysis. The definition of the cryptographic tables used in this study is provided below.

DDT (Difference Distribution Table) evaluates the differential properties of an S-box by showing the distribution of output differences for each input difference, which is crucial to assessing resistance to differential cryptanalysis [12]. Its coefficient $DDT_S(\Delta_i, \Delta_o)$ is defined as $\#\{x \in \{0,1\}^n \mid S(x) \oplus S(x \oplus \Delta_i) = \Delta_o\}$, where Δ_i and Δ_o are the input and output differences, respectively. Its maximum value is called the *differential uniformity* except when $\Delta_i = 0$.

LAT (Linear Approximation Table) presents the linear relationships between input and output bits, used in linear cryptanalysis [43] to identify biases that can be exploited. The value of the LAT coefficient $LAT_S(\lambda_i, \lambda_o)$ is $\#\{x \in \{0,1\}^n \mid x \cdot \lambda_i = S(x) \cdot \lambda_o\} - 2^{n-1}$, where λ_i and λ_o are the input and output masks, respectively. Its maximum value is called the *linearity* except when $\lambda_o = 0$.

Cid et al. [20] introduced a new cryptanalysis tool called BCT (Boomerang Connectivity Table), which is related to boomerang attacks and analyzes the S-box by mapping differential paths in both forward and backward directions. The value of the BCT coefficient $BCT_S(\Delta_i, \nabla_o)$ is given by $\#\{x \in \{0,1\}^n \mid S^{-1}(S(x) \oplus$

$\nabla_o) \oplus S^{-1}(S(x \oplus \Delta_i) \oplus \nabla_o) = \Delta_i\}$. Its maximum value is called the *boomerang uniformity* except when $\Delta_i, \nabla_o = 0$.

DLCT (Differential-Linear Connectivity Table), studied in detail by Bar-On et al. [8], combines differential and linear cryptanalysis approaches to provide insight into the susceptibility of ciphers to differential-linear attacks. The value of DLCT coefficient $DLCT_S(\Delta, \lambda)$ is defined as $\#\{x \in \{0,1\}^n \mid S(x) \cdot \lambda = S(x \oplus \Delta) \cdot \lambda\} - 2^{n-1}$, where Δ is a difference, and λ is a mask.

2.3 Anomalies of an S-Box

An anomaly of an S-box is used as a metric to quantify how rare or unlikely it is for an S-box to satisfy the cryptographic properties (e.g., differential uniformity) [16,46]. Let m_F be the maximum value of coefficients in a $2^n \times 2^n$ table T of an S-box F, and o_F be the number of occurrences of the maximum value. The anomaly for the table T can be expressed as $\mathsf{A}(T) = -\log_2(\Pr[m_G \leq m_F \text{ and } o_G \leq o_F])$, where the probability is taken over all G in the same space as F. The anomaly can be estimated as follows:

$$\mathsf{A}(T) = -\log_2 \left(\sum_{k=0}^{o_F} \binom{(2^n-1)^2}{k} \cdot p_{m_F}^k \cdot \left(\sum_{j=0}^{m_F-1} p_j \right)^{(2^n-1)^2-k} \right),$$

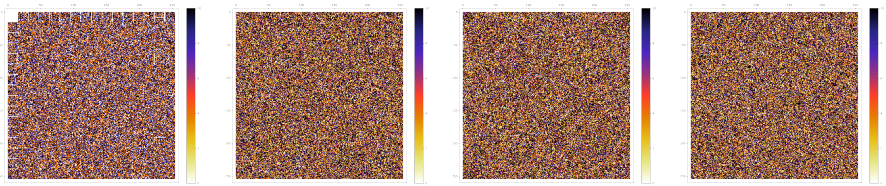
where p_i is the probability that $T(a,b) = |i|$. A high value of the anomaly indicates that the given S-box has a property distinctive enough to be set apart from a random permutation.

The differential and linear anomalies were proposed by Biryukov et al. [16], and the concept of anomalies was more clearly defined in the later study by Perrin [46], and Bonnetain et al. extended its application to boomerang uniformity [19]. This score can demonstrate that the given S-box exhibits unique properties beyond those of a random S-box, thereby providing a potential means of uncovering hidden design structures within S-boxes. However, it can only quantify the distance from randomness and is unable to quantify the distance between two different S-boxes. Also, the anomalies are affected only by the maximum value and its number of occurrences and do not consider other factors, such as non-maximum coefficients and positional information.

2.4 Pollock Representation and Pattern Recognition

Biryukov et al. introduced Pollock's pattern recognition, which transforms an S-box into a visual representation, known as the Pollock representation, and then identifies non-random patterns within the representation [16]. Their experimental results demonstrated that this approach facilitates the differentiation of specifically designed S-boxes from random ones through visual observation. Also, the Pollock representation of LAT inspired new recovery attacks for the 4-round Feistel structure, as proposed in [17, 48].

However, the effectiveness and performance of this method depend on human capabilities in pattern recognition. As the number of rounds in an S-box increases, its Pollock representation begins to resemble randomness more closely, making it challenging to visually discern. An example of this scenario is illustrated in Fig. 3. Despite the S-box of ICEBERG having one more round than that of CLEFIA, it becomes difficult to recognize any pattern that differs from the random S-box. As demonstrated in [16], while the *dents* pattern is easily recognized in CLEFIA, no clear pattern is found in Skipjack, making it indistinguishable from the random S-box. In this situation, human vision alone is insufficient to distinguish a random-looking S-box from a random S-box.



(a) CLEFIA's S_0 (b) ICEBERG's S (c) Skipjack's F (d) Random S-box

Fig. 3. The Pollock representation of the LAT for each S-box.

3 Deep Learning-based S-Box Structure Recovery

In this section, we introduce a deep learning-based framework for recovering S-box design structures. Our framework comprises two main steps: dataset generation and model training.

3.1 A Framework for S-Box Structure Recovery

Dataset Generation: Constructing S-Boxes and Preprocessing. An S-box design structure includes a construction function (CF), such as Feistel and Misty, and its number of rounds. Each round uses a smaller inner S-box to form its round function. For example, in an 8-bit Feistel S-box, the right 4-bit block is processed by a 4-bit inner S-box, and the resulting output is XORed with the left block. We employ random permutations as inner S-boxes, and each round uses a different permutation. In our experiments, we generate 20,000 S-boxes for each design structure, allocating 16,000 for training, 2,000 for validation, and the remainder for testing.

After generating the S-box datasets, we preprocess the S-boxes by converting them into cryptographic tables, such as the DDT, LAT, BCT, and DLCT. Previous studies focused on the distance between a given S-box and randomly selected S-boxes by examining statistical anomalies derived from these tables [16, 19, 46]. However, since these anomalies capture only human-filtered information—such

as the maximum coefficient and its occurrences—they are not suitable for deep learning-based methods due to significant information loss. Consequently, we do not convert the cryptographic tables into anomalies but instead use the tables themselves as training data. This approach can replace the power of human vision with that of deep learning in Pollock's pattern recognition. In our experiments, we preprocess 8-bit S-boxes into cryptographic tables and then crop each table to a 255×255 size by excluding the first row and column.

Training and Prediction. Following our previous steps, we generate training datasets for each S-box design structure, preparing us to begin training a neural network. During the training phase, we load the S-box datasets corresponding to each design structure, based on the classes we aim to classify. We then assign a label that indicates the design structure of each S-box, grouping together S-boxes with the same structure and separating those with different structures. We implement the label data with one-hot encoding, representing each label as an array. The number of dimensions in each encoded label is determined by the number of classes, and an element of each dimension represents the probability that the input belongs to the corresponding class. Finally, we train the model using the prepared training data along with these labels.

Once training is complete, the trained neural network is used to infer the design rationales of unknown S-boxes. To achieve this, the same preprocessing applied to the training dataset must also be applied to the given S-boxes before they are fed into the trained network. The network then outputs the probabilities of being classified into each class for a given input. Through analyzing the output probabilities, we are able to recover the design structure of the given S-box. The overview of this process can be seen in Fig. 4.

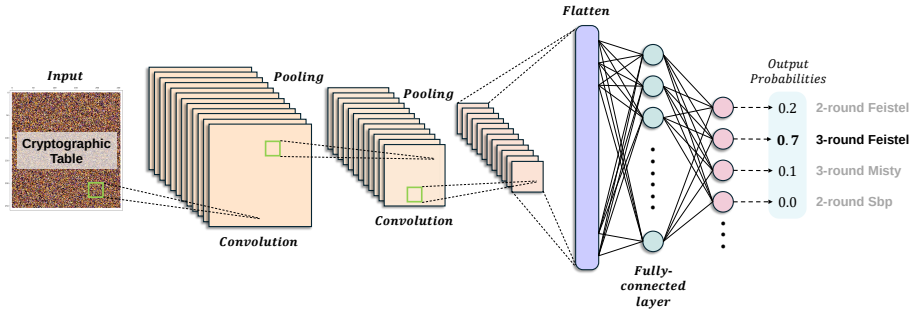


Fig. 4. Overview of the framework for S-box structure recovery.

Neural Network Structure. Since a cryptographic table can be treated as image data, we adopt a convolutional neural network architecture for our neural

network, building upon the approach proposed by Kim et al. [34]. Following the input layer, we attach a batch normalization layer and then add four convolutional blocks. The convolution layer in each block consists of 32 filters with kernel size 3×3, and their weights are initialized using the He normal initializer [30]. Following each convolution layer, we connect a batch normalization layer, a ReLU activation function, and a 2×2 average pooling layer. After flattening, we employ a fully connected layer with the Softmax activation function for the output layer. The number of output nodes equals the number of classes to be classified by the neural network. The cross-entropy function serves as the loss function, and the trainable parameters are updated using the Adam optimizer [37] to minimize the loss, with a batch size of 800.

Experimental Results. To evaluate our S-box structure recovery model performance, we train networks to classify 15 types of S-box design structures: 2 to 4-round Feistel, 2 to 4-round Misty, 1 to 3-round Sbp, 1 to 3-round bpSbp, 1-round Lai-Massey, 1-round Bridge, and an inversion-based structure. We denote the model that classifies given S-boxes into these 15 classes as $\mathtt{DL}^1_{\text{table}}$. We build four models for each cryptographic table—DDT, LAT, BCT, and DLCT—and train each model separately. The left plot in Fig. 5 shows the learning curve of the models, where the x-axis denotes the number of training epochs, and the y-axis denotes the validation accuracy. Our experimental results show that all four models converge within 50 epochs.

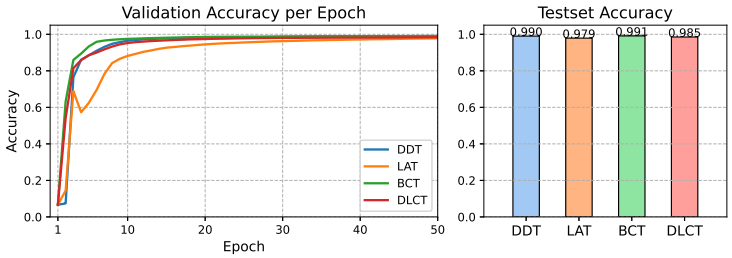


Fig. 5. Results of training history and testset evaluation for each model.

On the right in Fig. 5, the model with BCT achieves the highest accuracy of 0.991, slightly outperforming the others, but all models achieve high accuracy, over 0.979. Our results demonstrate that even though the Bridge structure has not been researched for structural recovery, the model can be seamlessly applied to classify this structure with high accuracy without prior knowledge. This strength contrasts with conventional structural recovery attacks [15–18], which can only target specific structures and require structure-specific pre-analysis, highlighting the broad applicability of our approach to new structures.

3.2 Interpreting the Model's Decision-Making Process

In this section, we aim to provide deeper insights into the input features on which our model bases its decisions. To achieve this goal, we apply an attribution method to score the contribution of each coefficient in the table. While previous methods [45, 55, 58, 61] used in computer vision attribute individual samples effectively, they cannot offer unified attributions across multiple samples within a single class. Therefore, we propose a new attribution method that scores a common attribution for S-boxes constructed with the same design structure.

Definition 1. *Given N samples and the result of element-wise multiplication between the n-th sample and its gradients denoted as G^n, an attribution score $S_{(i,j)}$ for the position (i, j) is defined as follows:*

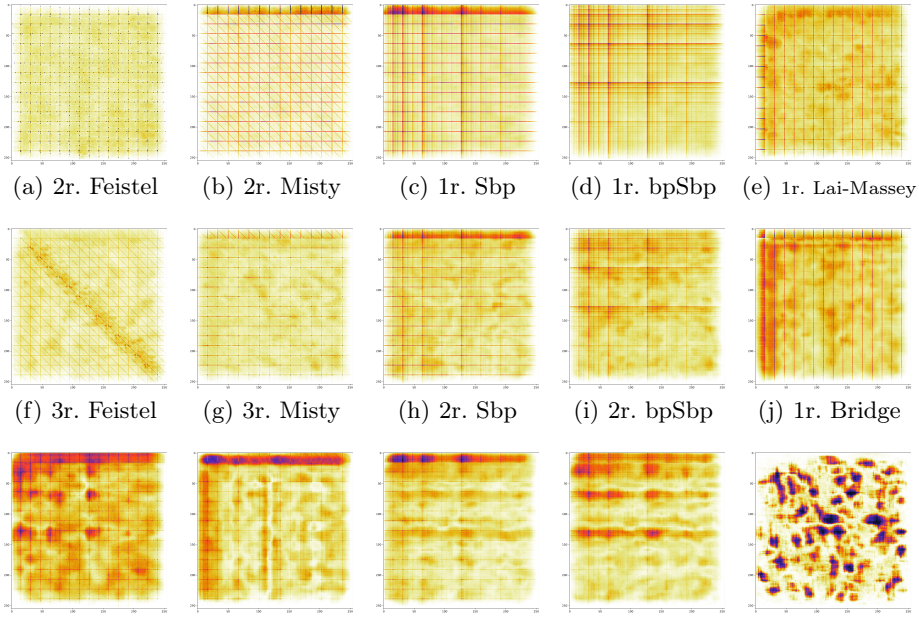
$$S_{(i,j)} = \sum_{n}^{N} s^n_{(i,j)}, \quad s^n_{(i,j)} \equiv \begin{cases} 1 & \text{if } G^n_{(i,j)} \in T^n \cup B^n \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$\textit{where } T^n = \underset{A \subset G^n, |A|=k}{\operatorname{argmax}} \sum_{a \in A} a, \; B^n = \underset{A \subset G^n, |A|=k}{\operatorname{argmin}} \sum_{a \in A} a$$

According to Definition 1, the attribution score is determined by extracting the top k and bottom k positions from G, which is the element-wise product of a given sample and its gradients, and summing these values across samples within the same class. In G, points with a positive value indicate the locations of input features that positively contribute to the model's decisions, while those with a negative value suggest a negative influence. In this definition, both positive evidence (T^n) and negative evidence (B^n) are weighted equally. The attribution map visualizes this evidence, playing a crucial role in illustrating how much the model's decisions depend on specific positions of input features. Examples of the attribution map are shown in Fig. 6. Darker colors represent greater influence, while lighter colors indicate positions with less impact.

Representatively, attribution maps from $\texttt{DL}^1_{\text{BCT}}$ across 15 different design structures are presented in Fig. 6, with 2,000 test set samples (N) included in each class and k set to 3,251 (5% of 65,025). In the attribution map of the 2-round Feistel structure, the pattern of regularly spaced lines can be observed. These lines are positioned where the lower 4-bit block is fixed to `0b1111` (= 15), and the upper 4-bit block increments by 1 from 0 onwards (e.g., 15, 33, 63, ...). In the attribution map of Misty, a similar pattern is observed, where the horizontal lines align with those of the Feistel pattern, but the vertical lines differ. The vertical lines are positioned where the upper 4-bit block increments by 1 from 1 onwards, and the lower 4-bit block is 1 less than the upper 4-bit block (e.g., 16, 33, 50, ...). On the other hand, in the attribution map of Lai-Massey, the pattern is reversed compared to Misty: the vertical lines align with those of the Feistel pattern, while the horizontal lines differ.

In the attribution map of Sbp, the horizontal lines follow the same pattern as in Feistel, but the vertical lines are positioned where the lower 4-bit block is



(a) 2r. Feistel (b) 2r. Misty (c) 1r. Sbp (d) 1r. bpSbp (e) 1r. Lai-Massey

(f) 3r. Feistel (g) 3r. Misty (h) 2r. Sbp (i) 2r. bpSbp (j) 1r. Bridge

(k) 4r. Feistel (l) 4r. Misty (m) 3r.Sbp (n) 3r. bpSbp (o) Inversion

Fig. 6. The attribution map of each S-box design from $\mathtt{DL}^1_{\mathrm{BCT}}$.

fixed to `0b1111` and the Hamming weight of the upper 4-bit block increments by one (e.g., 15, 31, 63, 127). Lastly, in the attribution map of bpSbp, both the vertical and horizontal lines follow the same spacing as the vertical lines in Sbp, with strong influence observed in areas with lower Hamming weight. It can also be seen that the attribution maps for S-boxes, which are constructed with different numbers of rounds but the same CF, show high contributions at the same positions, although the patterns become less distinct as the rounds deepen.

Interestingly, although we provide only labels indicating that each class is different—without any prior information about the relationships among classes—the deep learning model can uncover similarities between them. For example, even though we merely label 2-round, 3-round, and 4-round Feistel structures as distinct classes, the model recognizes shared features between them, as can be observed in their attribution maps. In other words, even without providing information about relationships among different design structures, our model successfully identifies both similarities and differences across these structures. These findings suggest that the deep learning model does more than simply classify each class; it actively learns the underlying design features of each S-box structure and leverages this understanding to differentiate among the classes.

4 Recovery Results on Untrained Structures

4.1 On Untrained but Derived Structures From Trained Ones

The performance of our model on the trained structures can be demonstrated by the high accuracy presented in Section 3.1. In this section, we further investigate whether our model maintains its effectiveness when applied to actual S-boxes. Table 1 presents the inference results of $\mathtt{DL}^{1}_{\mathrm{BCT}}$ for real-world S-boxes. Our model accurately predicts the design of S-boxes based on Inversion. There

Table 1. Prediction results on actual S-boxes.

Cipher	Design Type	Prediction (Output Probability)		
		Top 1	Top 2	Top 3
AES [22]	Inversion	Inversion (1.000)	1r bpSbp (0.000)	2r Sbp (0.000)
ARIA S_2 [38]	Inversion	Inversion (1.000)	1r bpSbp (0.000)	2r Sbp (0.000)
Camellia S_1 [3]	Inversion	Inversion (1.000)	1r bpSbp (0.000)	2r Sbp (0.000)
Camellia S_2	Inversion	Inversion (1.000)	1r bpSbp (0.000)	2r Sbp (0.000)
Camellia S_3	Inversion	Inversion (1.000)	1r bpSbp (0.000)	2r Sbp (0.000)
Camellia S_4	Inversion	Inversion (1.000)	1r bpSbp (0.000)	2r Sbp (0.000)
CLEFIA S_1 [54]	Inversion	Inversion (1.000)	1r bpSbp (0.000)	2r Sbp (0.000)
DBlock [60]	Inversion	Inversion (1.000)	1r bpSbp (0.000)	2r Sbp (0.000)
SEED S_0 [40]	Inversion	Inversion (1.000)	1r bpSbp (0.000)	2r Sbp (0.000)
SEED S_1	Inversion	Inversion (1.000)	1r bpSbp (0.000)	2r bpSbp (0.000)
Midori SSb_0 [6]	1r bpSbp	1r bpSbp (1.000)	2r bpSbp (0.000)	1r Lai-Massey (0.000)
Midori SSb_1	1r bpSbp	1r bpSbp (1.000)	2r bpSbp (0.000)	1r Bridge (0.000)
Midori SSb_2	1r bpSbp	1r bpSbp (0.992)	1r Sbp (0.005)	2r bpSbp (0.004)
Midori SSb_3	1r bpSbp	1r bpSbp (1.000)	2r bpSbp (0.000)	1r Lai-Massey (0.000)
QARMA Σ_0 [4]	1r Sbp	1r Sbp (1.000)	2r Sbp (0.000)	1r bpSbp (0.000)
QARMA Σ_1	1r Sbp	1r Sbp (1.000)	1r bpSbp (0.000)	2r Sbp (0.000)
QARMA Σ_2	1r Sbp	1r Sbp (1.000)	2r Sbp (0.000)	1r bpSbp (0.000)
QARMAv2 [5]	1r Sbp	1r Sbp (1.000)	2r Sbp (0.000)	1r bpSbp (0.000)
CLEFIA S_0 [54]	2r SMP†	3r bpSbp (0.506)	3r Sbp (0.470)	4r Feistel (0.015)
CRYPTONv1 S_0 [42]	2r SMP†	3r bpSbp (0.692)	3r Sbp (0.179)	4r Feistel (0.093)
CRYPTONv1 S_1	2r SMP†	3r bpSbp (0.723)	3r Sbp (0.173)	4r Feistel (0.085)
CRYPTONv1 S_2	2r SMP†	3r bpSbp (0.687)	3r Sbp (0.147)	4r Feistel (0.138)
CRYPTONv1 S_3	2r SMP†	3r bpSbp (0.696)	3r Sbp (0.174)	4r Feistel (0.114)
Enocoro-80 [59]	2r SMP†	3r bpSbp (0.732)	3r Sbp (0.258)	4r Misty (0.009)
Twofish q_0 [51]	2r SMP†	3r bpSbp (0.583)	4r Feistel (0.412)	3r Sbp (0.004)
Twofish q_1	2r SMP†	3r bpSbp (0.695)	4r Feistel (0.238)	3r Sbp (0.059)
ICEBERG [56]	3r Sbp‡	3r Sbp (0.864)	4r Misty (0.069)	3r bpSbp (0.035)
KHAZAD [9]	3r Sbp‡	3r bpSbp (0.845)	3r Sbp (0.144)	4r Feistel (0.007)
SKINNY 8bit [10]	1r Generalized Feistel	1r bpSbp (0.938)	1r Sbp (0.053)	2r Sbp (0.006)
CS-cipher [57]	3r Feistel	3r Feistel (1.000)	1r Lai-Massey (0.000)	2r bpSbp (0.000)
CRYPTONv0.5 S_0 [41]	3r Feistel	4r Feistel (0.718)	3r Feistel (0.282)	1r Lai-Massey (0.000)
CRYPTONv0.5 S_1	3r Feistel	4r Feistel (0.831)	3r Feistel (0.169)	1r Lai-Massey (0.000)
Lilliput-AE [1]	3r Feistel	3r Feistel (0.997)	4r Feistel (0.002)	2r bpSbp (0.000)
Scream v3 [29]	3r Feistel	3r Feistel (0.999)	4r Feistel (0.001)	1r Lai-Massey (0.000)
Robin (iScream) [26, 28]	3r Feistel	2r bpSbp (0.730)	1r Bridge (0.257)	3r bpSbp (0.012)
Fantomas [26]	3r $\mathrm{Misty}_{(5,3,5)}$	2r bpSbp (0.984)	2r Sbp (0.015)	1r Lai-Massey (0.001)
Zorro [25]	4r Feistel Linear Mix	4r Feistel (0.857)	3r bpSbp (0.070)	3r Sbp (0.063)
FLY (Littlun-1) [33]	1r Lai-Massey	1r Lai-Massey (1.000)	3r Feistel (0.000)	2r bpSbp (0.000)
Fox [31]	1r Lai-Massey	1r Lai-Massey (0.705)	4r Feistel (0.280)	3r Misty (0.008)
MD2 [32]	Random	3r bpSbp (0.535)	4r Feistel (0.293)	3r Sbp (0.153)
newDES [52]	Random	3r bpSbp (0.875)	3r Sbp (0.077)	4r Feistel (0.041)
Turing [49]	Random	3r bpSbp (0.466)	4r Feistel (0.285)	3r Sbp (0.236)

r stands for the number of rounds.
† This structure adopts a matrix-based permutation layer.
‡ Without the final permutation.

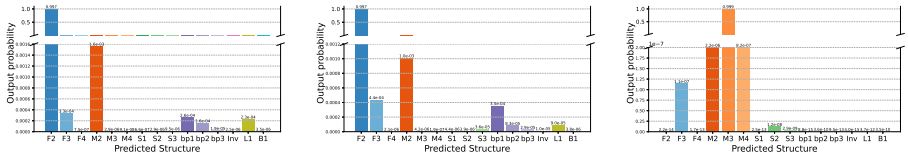
are instances of accurate predictions for other structures; however, in some cases, the construction function is correct but its number of rounds is not. It is supposed to stem from the difference between the structures that our model is trained on and the actual structures in the S-boxes. For trained structures, such as Midori and QARMA, since the model is trained on exactly the same structures as those in the actual structures, the inference results are very accurate, including their number of rounds.

Based on the results in Table 1, it can be observed that our model achieves higher accuracy when the actual S-box structure closely resembles the trained structures. Although the model is not perfectly accurate in some cases, it still infers similar design structures. To further analyze this, we examine how our model infers S-boxes with modified structures that are derived from those it is trained on. We construct new S-boxes by combining the Feistel and Misty structures to design slight variations of known design structures. Each S-box consists of three rounds, where each round follows either Feistel or Misty. As a result, we consider a total of six new structures, denoted as `FFM` (Feistel-Feistel-Misty), `FMF`, `FMM`, `MFF`, `MFM`, and `MMF`. Moreover, to focus solely on structural changes, we employ the same 4-bit inner S-boxes for the six structures. In other words, we use distinct inner S-boxes for each round while employing the same S-box in the corresponding rounds across all structures. The prediction results for these structures are shown in Table 2. It can be seen that if Feistel is used in the first round, our model classifies it as Feistel, whereas if Misty is used, the model classifies it as Misty.

Table 2. Prediction results for the 8 design structures.

Design Structure	**Feistel-**				**Misty-**			
	`FFF`	`FFM`	`FMF`	`FMM`	`MFF`	`MFM`	`MMF`	`MMM`
Predicted Class	3-round Feistel	2-round Feistel	2-round Feistel	2-round Feistel	3-round Misty	2-round Misty	3-round Misty	3-round Misty

According to the results in Section 3.2, our model does not simply memorize the inputs; rather, it learns the unique patterns of each design structure and uses similarities and differences to distinguish them. Consequently, in a case such as `FMM`, it can be conjectured that the next highest output probability (after Feistel) would belong to Misty, and this conjecture is indeed confirmed by Fig. 7. Fig. 7 shows the output probabilities for three representative cases: `FFM`, `FMM`, and `MFF`. As shown in Fig. 7, our model infers, with high confidence, the structure employed in the first round of each S-box. It also can be observed that, except the structure of the first round, the second-highest class corresponds to the other structure that comprises the given S-box. These findings suggest that the model can detect not only the main structure but also other hidden structures within an S-box.

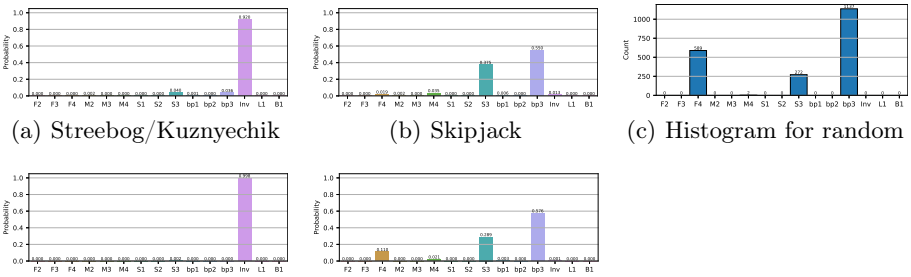


(a) `FFM` (b) `FMM` (c) `MFF`

Fig. 7. Detailed visualization of output probabilities for `FFM`, `FMM` and `MFF`.

4.2 On Untrained and Unknown Structures

Despite the high likelihood that the design structures of Streebog's π and Skipjack's F are not included in our training datasets, we can infer the design similarity between these S-boxes and related S-boxes. The inference results for the S-boxes of Streebog and Skipjack can be seen in Figs. 8(a) and 8(b), respectively. We believe that these inference results indicate the probability of classification into the closest class among the structures that our model is trained on, rather than directly predicting the design structure itself. For instance, according to Table 1, our model inferred the S-boxes generated randomly, like the S-boxes of MD2, Turing and newDES, as 3-round bpSbp. This implies that our model considers the random S-boxes to be the closest to the 3-round bpSbp among the trained structures. To verify this, we aggregate the prediction results for 2,000 random S-boxes and show the histogram in Fig. 8(c).



(a) Streebog/Kuznyechik (b) Skipjack (c) Histogram for random

(d) BelT (e) Anubis

Fig. 8. Prediction results on unknown S-boxes from $\mathtt{DL}^1_{\mathrm{BCT}}$.

Given that a trained neural network produces similar outputs for inputs with close features, it can be hypothesized that comparing the output probabilities of S-boxes can help infer the design similarities between them. According to the existing studies in [47,48], S-boxes of BelT and Anubis are presumed to have design structures similar to the S-boxes of Streebog and Skipjack, respectively. The inference results for the S-boxes can be found in Figs. 8(d) and 8(e), confirming our hypothesis. These results strongly imply that the Streebog S-box does not

belong to a random family, contrary to the claims of Russian cryptographers, and this conclusion aligns with the claims made in the previous study using the concept of anomalies [19]. However, similar to previous research, our model demonstrates its limitation in identifying design structures. Nevertheless, these findings indicate that our model can estimate the similarity between S-boxes, even for untrained structures.

4.3 Recovering the Number of Rounds of Untrained Structures

As demonstrated in Section 4.2, the model faces limitations in recovering structures it has not been trained on. *If no prior information about the design of an S-box is available, is it possible to estimate at least its number of rounds?* To address this question, we investigate whether the number of rounds in an S-box can be inferred without training its design structure. Our framework can be easily adapted to perform different types of attacks, depending on how the training dataset is structured. To recover the number of rounds, we present a round identification model that classifies S-boxes constructed with different numbers of rounds but the same CF. In our experiment, we denote this model as $\mathtt{DL}^2_{\text{table}}$, which is trained to classify S-boxes constructed with 2 to 7-round Misty structures.

Each element in a neural network's output for a sample represents a probability that the sample belongs to each class, and the class with the highest probability becomes the predicted class by the model. When the model is a binary classification model, we can display the probability of being classified into the first class, i.e. the first element of output, however, it is not possible for a multi-class classification model. We introduce a new method for visualizing the output probabilities of the round identification model. When output probabilities of an n-class model are denoted as $(p_0, p_1, ..., p_{n-1})$, we display the following value:

$$y = y_{pred} - \sum_{i<y_{pred}} p_i + \sum_{j>y_{pred}} p_j, \text{ where } y_{pred} = \underset{i}{\operatorname{argmax}}\, p_i \tag{2}$$

In Fig. 9, the x-axis represents the index of the sample, showing 2,000 samples each from 2-round Feistel to 1-round Bridge in order, and the y-axis represents the inference results according to equation (2). As observed in Fig. 9, samples with design structures not considered in the training process, such as Feistel, Sbp, and bpSbp, are not classified by rounds accurately, but the trend can be seen where, akin to stairs, the classification gradually shifts to higher rounds as the number of rounds increases. Considering that the 1-round Lai-Massey and Bridge contain three inner S-boxes, even without prior training on these structures, the model infers them somewhat correctly as either 3-round or 4-round that biased towards 3-round. These findings suggest that, despite not being trained on the design structures of given S-boxes, our model has the potential to effectively distinguish their number of rounds.

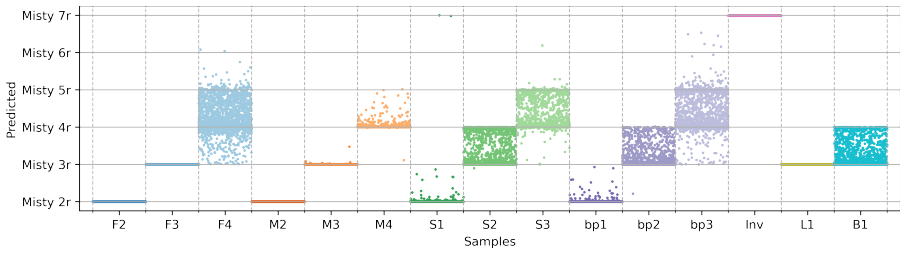


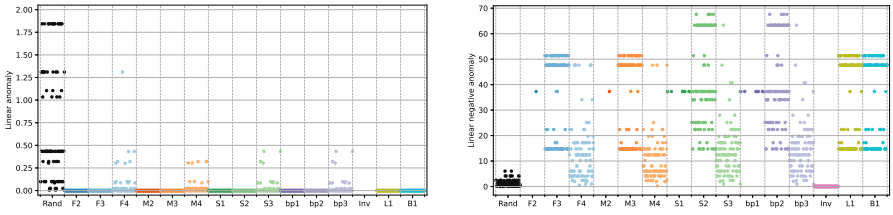
Fig. 9. Inference results for the 15 design structures from the $\mathtt{DL}^2_{\mathrm{BCT}}$ model trained only with the Misty structures.

5 Quantifying Distances Between S-boxes

The concept of anomalies of an S-box is introduced to quantify how rare an S-box is with respect to certain properties, thereby measuring how far an S-box deviates from randomly selected ones. However, while this method can reveal whether an S-box is non-randomly generated, it is less suited to evaluating the similarity or distance between two different S-boxes. To address this issue, we propose new approaches that quantify distances using deep learning-based embeddings.

5.1 Revisiting the Conventional Method: Anomaly of an S-Box

First, before introducing a deep learning-based approach, we briefly revisit the anomaly of an S-box. Fig. 10 displays the linear anomalies of random S-boxes and those of S-boxes constructed with 15 different design structures, and each class contains 100 samples. In Fig. 10(a), it seems that since the inner S-boxes of each S-box are chosen uniformly at random without considering any properties, it is hard to distinguish them from random S-boxes using positive anomalies. In this case, we can consider applying the concept of negative anomalies proposed by Bonnetain et al. [19]. However, as seen in Fig. 10(b), even with negative anomalies, distinguishing 4-round Feistel, Misty, 3-round Sbp, and bpSbp from random S-boxes remains challenging in some instances.



(a) Positive anomaly (b) Negative anomaly

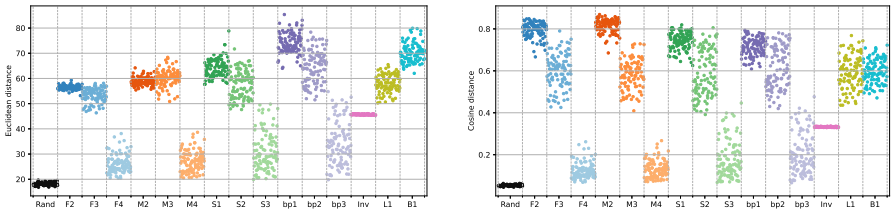
Fig. 10. Linear anomalies of S-boxes constructed with each design structure.

Additionally, the anomalies of Sbp and bpSbp S-boxes with the same inner S-boxes are identical. This is because affine-equivalent S-boxes [14] have the same values of anomalies. For example, given a 1-round Sbp S-box S and a bitwise permutation P, the differences between the LAT of S and $S \circ P$ (1-round bpSbp) lie solely in the rearrangement of coefficients, rather than their values. Consequently, distinguishing between these S-boxes based on their anomalies becomes infeasible. These results in Figs. 1 and 10 highlight the limitations of the anomalies of an S-box in identifying S-box design structures.

5.2 Embedding S-Boxes using Multi-Class Classification Model

According to the results in Section 3.2, our model does not merely distinguish the design structures of S-boxes but also recognizes their similarities. Building on this observation, we propose a method that extracts each S-box's embedding vector from our trained classification model, then quantifies the distance between these embeddings. Concretely, we use the output of the fully-connected layer immediately before the final softmax layer in our neural network as the embedding vector. Since the neural network learns to separate each class distinctly, it naturally forms clustered distributions in the embedding space based on class. Consequently, a model with high classification accuracy tends to learn an embedding space where each class is well separated. By leveraging these embedding vectors, we quantify the distance between two S-boxes using various metrics such as the Euclidean (L_2) and the cosine distances.

We verify our proposed method by extracting embedding vectors for each S-box from our trained $\mathtt{DL}^1_{\mathrm{DDT}}$ model and then measuring their distances. Fig. 11 shows how far each structured S-box is from random S-boxes, using both the L_2 distance and the cosine distance. We set the mean embedding vector of 2,000 random S-boxes as the centroid of the random S-box embedding vectors. Each point in Fig. 11 corresponds to a single S-box, and its position on the y-axis indicates its distance from the centroid of random S-boxes.



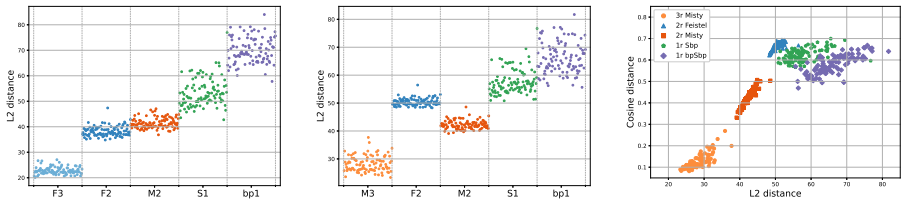
(a) L_2 distance (b) Cosine distance

Fig. 11. Distances from the center of random permutations to each S-box.

As shown in Fig. 11, it can be observed that the design structures become closer to the centroid of random S-boxes as the number of rounds increases.

This trend appears in both L_2 distance and cosine distance, and it demonstrates that as the S-boxes become closer to random, the distances also decrease. These observations suggest that similar to the conventional method, the distance to random S-boxes can be quantified by measuring the distance from the centroid of their embedding vectors.

Our method can also be applied to quantify distances between S-boxes with different structures. Figs. 12(a) and 12(b) illustrate the distances from the centroids of 3-round Feistel and 3-round Misty, respectively, to S-boxes constructed with 2-round Feistel, 2-round Misty, 1-round Sbp and 1-round bpSbp. In Fig. 12(a), the closest S-boxes to 3-round Feistel are the 2-round Feistel S-boxes, followed by the 2-round Misty S-boxes, and then the 1-round Sbp and bpSbp S-boxes, in that order. Similarly, Fig. 12(b) shows that 2-round Misty is closest to 3-round Misty, with Feistel as the next closest structure. These results confirm that S-boxes with more similar designs lie nearer in the embedding space, highlighting the effectiveness of our embedding-based approach for evaluating structural similarity.



(a) From 3r. Feistel (b) From 3r. Misty (c) Both From 3r. Misty

Fig. 12. Distances from the center of each structure to each S-box.

Fig. 12(c) displays the results of both the L_2 distance and cosine distance from the centroids of 3-round Misty. In both distance metrics, the structure closest to 3-round Misty is 2-round Misty, which shares the same construction function. When considering only the L_2 distance, it is difficult to distinguish between the 1-round Sbp and bpSbp structures due to significant overlap; by contrast, with the cosine distance, it becomes visually evident that these structures can be differentiated. This indicates that using multiple distance metrics in the S-box embedding space helps effectively distinguish between different structures.

5.3 Embedding S-Boxes using One-Class Classification Model

In order to train the classification model used in the embedding method introduced in Section 5.2, a large amount of training data for the design structures to be identified is required. However, in real-world scenarios, the number of samples for unknown S-boxes is limited, and no prior information about their designs is available, making it challenging to apply supervised or semi-supervised learning

models. Moreover, the multi-class classification model-based embedding method does not explicitly define the mapping of each sample in the embedding space. The multi-class classification model is trained solely to predict correct labels, thus there is no strong guarantee that distances between embedding vectors reflect structural similarity.

To overcome these limitations, we apply Deep SVDD (Support Vector Data Description) [50] as a one-class classification model, which is one of the unsupervised anomaly detection methods. The primary goal of Deep SVDD is to map input data into a latent space and minimize the distance between normal data and the center. Samples that deviate significantly from the center are classified as anomalies. The loss function for Deep SVDD is defined as follows:

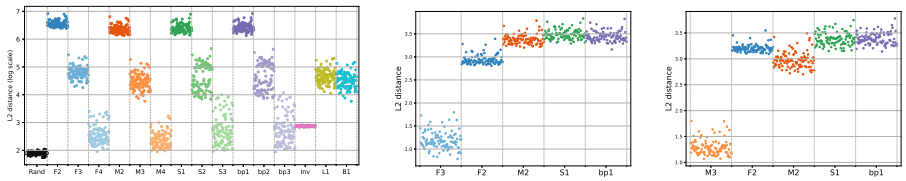
$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \|\phi(x_i; \theta) - c\|^2 + \lambda \|\theta\|^2$$

where $\phi(x_i; \theta)$ represents the neural network that maps the input data x_i into a latent space, c is the center of the latent space, θ is the set of parameters of the neural network, and λ is a hyperparameter controlling the weight decay. The first term in the loss function is designed to minimize the average L_2 distance between normal data points and the center c, so that normal data points cluster around the center. The second term represents weight decay regularization, which penalizes overly large parameter values to control model complexity and mitigate overfitting.

The Deep SVDD model is trained on only one normal class. After training, it can assess the similarity between an input S-box and the class it was trained on, since embeddings from other classes are generally farther from the center in the latent space. This approach functions as an anomaly detection method, where data points far from the normal class are considered anomalies. While it is similar to the existing approach, namely the anomaly of an S-box, it differs in that the distance is not solely determined by the maximum value in a cryptographic table but rather takes into account all the coefficients in the table. Furthermore, since not only random S-boxes but also an S-box design structure can be set as a normal class, our approach can quantify the distance between a given S-box and a specified structure. By constructing individual Deep SVDD models for known structures, the similarity of a given S-box to the trained structures can be effectively evaluated.

To validate the proposed method, we conduct experiments with three Deep SVDD models. Each model is trained on 2,000 samples from a single class: Random S-box, 3-round Feistel, and 3-round Misty. The results of each model are shown in Figs. 13(a), 13(b), and 13(c). As seen in Fig. 13(a), even though the model is trained solely on Random S-boxes, the structures with different rounds are distinguishable, which is consistent with the patterns observed in Fig. 11. Figs. 13(b) and 13(c) demonstrate that S-boxes with similar structures are closer in distance compared to those with different structures, consistent with the observations in Figs. 12(a) and 12(b). Despite the model being trained on only one class with fewer samples, the results align with those in Section 5.2,

where models are trained on all classes. This consistency further supports the effectiveness of the proposed approach. However, as observed in Figs. 13(b) and 13(c), the separation between similar structures and others is less evident than in the classification model trained on various structures.



(a) Random S-box (b) 3r. Feistel (c) 3r. Misty

Fig. 13. Results from the Deep SVDD models that were trained on each structure.

6 Conclusions

In this study, we introduce the deep learning-based framework for recovering the design structures of S-boxes. Our framework trains a neural network to classify the design structure of a given S-box based on its cryptographic table. Leveraging our framework, we propose new methods to quantify distances not only from random S-boxes but also between different S-boxes. For example, while a designer may intuitively infer that the Feistel structure is closer in distance to the Misty structure than to the Sbp structure, our methods provide a quantitative basis for such assessments using deep learning techniques. Our approach offers broader applicability compared to conventional anomaly-based methods and can be seamlessly applied to newly designed structures. Consequently, it is expected to provide significant insights for cryptanalysts and designers in S-box reverse-engineering. Furthermore, our findings indicate that deep learning can enhance cryptanalysis in ways beyond serving as a neural distinguisher.

Acknowledgment. This work was supported by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korea Government (MSIT) under Grant RS-2023-00229028.

References

1. Adomnicai, A., Berger, T.P., Clavier, C., Francq, J., Huynh, P., Lallemand, V., Le Gouguec, K., Minier, M., Reynaud, L., Thomas, G.: Lilliput-ae: a new lightweight tweakable block cipher for authenticated encryption with associated data. Submitted to NIST Lightweight Project pp. 1–58 (2019)

2. Agency, N.S.A.N.S.: Skipjack and kea algorithm specifications (1998), https://csrc.nist.gov/presentations/1998/skipjack-and-kea-algorithm-specifications
3. Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., Tokita, T.: Camellia: A 128-bit block cipher suitable for multiple platforms—design andanalysis. In: Selected Areas in Cryptography: 7th Annual International Workshop, SAC 2000 Waterloo, Ontario, Canada, August 14–15, 2000 Proceedings 7. pp. 39–56. Springer (2001)
4. Avanzi, R.: The qarma block cipher family. almost mds matrices over rings with zero divisors, nearly symmetric even-mansour constructions with non-involutory central rounds, and search heuristics for low-latency s-boxes. IACR Transactions on Symmetric Cryptology pp. 4–44 (2017)
5. Avanzi, R., Banik, S., Dunkelman, O., Eichlseder, M., Ghosh, S., Nageler, M., Regazzoni, F.: The qarmav2 family of tweakable block ciphers. IACR Transactions on Symmetric Cryptology **2023**(3), 25–73 (2023)
6. Banik, S., Bogdanov, A., Isobe, T., Shibutani, K., Hiwatari, H., Akishita, T., Regazzoni, F.: Midori: A block cipher for low energy. In: Advances in Cryptology–ASIACRYPT 2015: 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29–December 3, 2015, Proceedings, Part II 21. pp. 411–436. Springer (2015)
7. Bannier, A., Bodin, N., Filiol, E.: Partition-based trapdoor ciphers. Cryptology ePrint Archive, Paper 2016/493 (2016), https://eprint.iacr.org/2016/493, https://eprint.iacr.org/2016/493
8. Bar-On, A., Dunkelman, O., Keller, N., Weizman, A.: DLCT: A new tool for differential-linear cryptanalysis. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 313–342. Springer, Heidelberg (May 2019). https://doi.org/10.1007/978-3-030-17653-2_11
9. Barreto, P., Rijmen, V.: The khazad legacy-level block cipher. Primitive submitted to NESSIE **97**(106) (2000)
10. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The skinny family of block ciphers and its low-latency variant mantis. In: Advances in Cryptology–CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II 36. pp. 123–153. Springer (2016)
11. Biham, E.: A fast new des implementation in software. In: Fast Software Encryption: 4th International Workshop, FSE'97 Haifa, Israel, January 20–22 1997 Proceedings 4. pp. 260–272. Springer (1997)
12. Biham, E., Shamir, A.: Differential cryptanalysis of des-like cryptosystems. Journal of CRYPTOLOGY **4**, 3–72 (1991)
13. Bilgin, B., De Meyer, L., Duval, S., Levi, I., Standaert, F.X.: Low and depth and efficient inverses: a guide on s-boxes for low-latency masking. IACR Transactions on Symmetric Cryptology **2020**(1), 144–184 (2020)
14. Biryukov, A., De Canniére, C., Braeken, A., Preneel, B.: A toolbox for cryptanalysis: Linear and affine equivalence algorithms. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 33–50. Springer, Heidelberg (May 2003). https://doi.org/10.1007/3-540-39200-9_3
15. Biryukov, A., Leurent, G., Perrin, L.: Cryptanalysis of Feistel networks with secret round functions. In: Dunkelman, O., Keliher, L. (eds.) SAC 2015. LNCS, vol. 9566, pp. 102–121. Springer, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-319-31301-6_6

16. Biryukov, A., Perrin, L.: On reverse-engineering S-boxes with hidden design criteria or structure. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part I. LNCS, vol. 9215, pp. 116–140. Springer, Heidelberg (Aug 2015). https://doi.org/10.1007/978-3-662-47989-6_6
17. Biryukov, A., Perrin, L., Udovenko, A.: Reverse-engineering the S-box of Streebog, Kuznyechik and STRIBOBr1. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part I. LNCS, vol. 9665, pp. 372–402. Springer, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49890-3_15
18. Biryukov, A., Shamir, A.: Structural cryptanalysis of SASAS. Journal of Cryptology **23**(4), 505–518 (Oct 2010). https://doi.org/10.1007/s00145-010-9062-1
19. Bonnetain, X., Perrin, L., Tian, S.: Anomalies and vector space search: Tools for S-box analysis. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part I. LNCS, vol. 11921, pp. 196–223. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-34578-5_8
20. Cid, C., Huang, T., Peyrin, T., Sasaki, Y., Song, L.: Boomerang connectivity table: A new cryptanalysis tool. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 683–714. Springer, Heidelberg (Apr / May 2018). https://doi.org/10.1007/978-3-319-78375-8_22
21. Coppersmith, D.: The data encryption standard (des) and its strength against attacks. IBM journal of research and development **38**(3), 243–250 (1994)
22. Dworkin, M., Barker, E., Nechvatal, J., Foti, J., Bassham, L., Roback, E., Dray, J.: Advanced encryption standard (aes) (2001-11-26 2001). https://doi.org/https://doi.org/10.6028/NIST.FIPS.197
23. Federal Agency for Technical Regulation and Metrology: GOST R 34.11-2012: hash function. Tech. rep. (2013), available at https://www.rfc-editor.org/rfc/rfc6986.html
24. Federal Agency for Technical Regulation and Metrology: GOST R 34.12-2015: Block Cipher Kuznyechik. Tech. rep. (2016), available at https://www.rfc-editor.org/rfc/rfc7801.html
25. Gérard, B., Grosso, V., Naya-Plasencia, M., Standaert, F.X.: Block ciphers that are easier to mask: How far can we go? In: Cryptographic Hardware and Embedded Systems-CHES 2013: 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings 15. pp. 383–399. Springer (2013)
26. Grosso, V., Leurent, G., Standaert, F.X., Varıcı, K.: Ls-designs: Bitslice encryption for efficient masked software implementations. In: Fast Software Encryption: 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers 21. pp. 18–37. Springer (2015)
27. Grosso, V., Leurent, G., Standaert, F.X., Varici, K.: LS-designs: Bitslice encryption for efficient masked software implementations. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 18–37. Springer, Heidelberg (Mar 2015). https://doi.org/10.1007/978-3-662-46706-0_2
28. Grosso, V., Leurent, G., Standaert, F.X., Varici, K., Durvaux, F., Gaspar, L., Kerckhof, S.: Scream & iscream side-channel resistant authenticated encryption with masking. Submission to CAESAR (2014)
29. Grosso, V., Leurent, G., Standaert, F.X., Varici, K., Journault, A., Durvaux, F., Gaspar, L., Kerckhof, S.: Scream side-channel resistant authenticated encryption with masking. CAESAR submission (2015)
30. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision. pp. 1026–1034 (2015)

31. Junod, P., Vaudenay, S.: Fox: a new family of block ciphers. In: Selected Areas in Cryptography: 11th International Workshop, SAC 2004, Waterloo, Canada, August 9-10, 2004, Revised Selected Papers 11. pp. 114–129. Springer (2005)
32. Kaliski, B.: Rfc1319: The md2 message-digest algorithm (1992)
33. Karpman, P., Grégoire, B.: The littlun s-box and the fly block cipher. In: Lightweight Cryptography Workshop (2016)
34. Kim, D.h., Kim, S., Hong, D., Sung, J., Hong, S.: An study on the analysis of design criteria for s-box based on deep learning. Journal of the Korea Institute of Information Security & Cryptology **30**(3), 337–347 (2020)
35. Kim, H., Jeon, Y., Kim, G., Kim, J., Sim, B.Y., Han, D.G., Seo, H., Kim, S., Hong, S., Sung, J., Hong, D.: A new method for designing lightweight s-boxes with high differential and linear branch numbers, and its application. IEEE Access **9**, 150592–150607 (2021)
36. Kim, H., Jeon, Y., Kim, G., Kim, J., Sim, B.Y., Han, D.G., Seo, H., Kim, S., Hong, S., Sung, J., et al.: Pipo: A lightweight block cipher with efficient higher-order masking software implementations. In: Information Security and Cryptology–ICISC 2020: 23rd International Conference, Seoul, South Korea, December 2–4, 2020, Proceedings 23. pp. 99–122. Springer (2021)
37. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
38. Kwon, D., Kim, J., Park, S., Sung, S.H., Sohn, Y., Song, J.H., Yeom, Y., Yoon, E.J., Lee, S., Lee, J., et al.: New block cipher: Aria. In: International conference on information security and cryptology. pp. 432–445. Springer (2003)
39. Lai, X., Massey, J.L.: A proposal for a new block encryption standard. In: Damgård, I. (ed.) EUROCRYPT'90. LNCS, vol. 473, pp. 389–404. Springer, Heidelberg (May 1991). https://doi.org/10.1007/3-540-46877-3_35
40. Lee, H., Lee, S., Yoon, J., Cheon, D., Lee, J.: Rfc 4269: The seed encryption algorithm (2005)
41. Lim, C.H.: Crypton: A new 128-bit block cipher. NIST AES Proposal (1998)
42. Lim, C.H.: A revised version of crypton: Crypton v1. 0. In: International Workshop on Fast Software Encryption. pp. 31–45. Springer (1999)
43. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT'93. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (May 1994). https://doi.org/10.1007/3-540-48285-7_33
44. Matsui, M.: New block encryption algorithm MISTY. In: Biham, E. (ed.) FSE'97. LNCS, vol. 1267, pp. 54–68. Springer, Heidelberg (Jan 1997). https://doi.org/10.1007/BFb0052334
45. Montavon, G., Samek, W., Müller, K.R.: Methods for interpreting and understanding deep neural networks. Digital signal processing **73**, 1–15 (2018)
46. Perrin, L.: Partitions in the S-box of Streebog and Kuznyechik. IACR Trans. Symm. Cryptol. **2019**(1), 302–329 (2019). https://doi.org/10.13154/tosc.v2019.i1.302-329
47. Perrin, L., Udovenko, A.: Exponential s-boxes: a link between the s-boxes of BelT and Kuznyechik/Streebog. IACR Trans. Symm. Cryptol. **2016**(2), 99–124 (2016). https://doi.org/10.13154/tosc.v2016.i2.99-124, https://tosc.iacr.org/index.php/ToSC/article/view/567
48. Perrin, L.P.: Cryptanalysis, reverse-engineering and design of symmetric cryptographic algorithms. Ph.D. thesis, Unilu-University of Luxembourg, Luxembourg, Luxembourg (2017)
49. Rose, G.G., Hawkes, P.: Turing: A fast stream cipher. In: International Workshop on Fast Software Encryption. pp. 290–306. Springer (2003)

50. Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S.A., Binder, A., Müller, E., Kloft, M.: Deep one-class classification. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 4393–4402. PMLR (10–15 Jul 2018), https://proceedings.mlr.press/v80/ruff18a.html
51. Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., Ferguson, N.: Twofish: A 128-bit block cipher. NIST AES Proposal **15**(1), 23–91 (1998)
52. Scott, R.: Wide-open encryption design offers flexible implementations. Cryptologia **9**(1), 75–91 (1985)
53. Shannon, C.E.: Communication theory of secrecy systems. The Bell system technical journal **28**(4), 656–715 (1949)
54. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-bit blockcipher clefia. In: Fast Software Encryption: 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, Revised Selected Papers 14. pp. 181–195. Springer (2007)
55. Shrikumar, A., Greenside, P., Shcherbina, A., Kundaje, A.: Not just a black box: Learning important features through propagating activation differences. arXiv preprint arXiv:1605.01713 (2016)
56. Standaert, F.X., Piret, G., Rouvroy, G., Quisquater, J.J., Legat, J.D.: Iceberg: An involutional cipher efficient for block encryption in reconfigurable hardware. In: Fast Software Encryption: 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004. Revised Papers 11. pp. 279–298. Springer (2004)
57. Stern, J., Vaudenay, S.: Cs-cipher. In: Fast Software Encryption, 5th International Workshop, FSE '98, Paris, France, March 23-25, 1998, Proceedings. Lecture Notes in Computer Science, vol. 1372, pp. 189–205. Springer (1998). https://doi.org/10.1007/3-540-69710-1_13
58. Sundararajan, M., Taly, A., Yan, Q.: Axiomatic attribution for deep networks. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 3319–3328. PMLR (06–11 Aug 2017), https://proceedings.mlr.press/v70/sundararajan17a.html
59. Watanabe, D., Ideguchi, K., Kitahara, J., Muto, K., Furuichi, H., Kaneko, T.: Enocoro-80: A hardware oriented stream cipher. In: 2008 Third International Conference on Availability, Reliability and Security. pp. 1294–1300. IEEE (2008)
60. Wu, W., Zhang, L., Yu, X.: The dblock family of block ciphers. Science China. Information Sciences **58**(3), 1–14 (2015)
61. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2921–2929 (2016)