Improving Differential-Neural Cryptanalysis with Inception

Liu Zhang[1,2], Zilong Wang[1,2], and Baocang Wang[3]

[1] School of Cyber Engineering, Xidian University, Xi'an, China,
liuzhang@stu.xidian.edu.cn,zlwang@xidian.edu.cn
[2] State Key Laboratory of Cryptology, P.O.Box 5159, Beijing 100878, China
[3] State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an, China,bcwang79@aliyun.com

Abstract. In CRYPTO'19, Gohr proposed a new cryptanalysis method by building the differential-neural distinguisher with a neural network. Gohr combined a differential-neural distinguisher with a classical differential, achieving a 12-round (out of 22) key recovery attack on Speck32/64. Bao *et al.* improved the classical differential by generalizing the concept of neutral bits, leading to key recovery attacks for 13-round Speck32/64 and 16-round (out of 32) Simon32/64.

Our primary objective is to enhance the differential-neural distinguisher's capabilities by applying deep learning techniques, focusing on handling more rounds and improving accuracy. We adopt a design inspired by the Inception Block in GoogLeNet to effectively capture information across multiple dimensions, employing multiple parallel convolutional layers with different kernel sizes positioned before the Residual Network. In the case of Speck32/64, our efforts yield accuracy improvements in rounds 6, 7, and 8, enabling the successful training of a 9-round differential-neural distinguisher. As for Simon32/64, we develop a differential-neural distinguisher capable of effectively handling 12 rounds while achieving noteworthy accuracy enhancements in rounds 9, 10, and 11. Additionally, we utilize neutral bits to ensure the required data distribution for launching a successful key recovery attack using multiple-ciphertext pairs as input for the neural network.

Combining these various advancements allows us to considerably reduce the time and data complexity of key recovery attacks on 13-round Speck32/64. In particular, we achieve a successful 14-round key recovery attack by exhaustively guessing a 1-round subkey, marking a significant milestone in differential-neural cryptanalysis. In the case of Simon32/64, we accomplish a groundbreaking 17-round key recovery attack for the first time and reduce the time complexity of the 16-round key recovery attack.

Keywords: Differential-Neural Distinguisher · Inception · Speck · Simon· Key Recovery Attack

1 Introduction

In CRYPTO 2019, Gohr [12] proposed the idea of differential-neural cryptanalysis. The differential-neural distinguisher, trained by the neural network, is introduced as the underlying distinguisher. Bayesian search speeds up key recovery attacks compared to classical differential cryptanalysis. The differential-neural distinguisher can distinguish whether ciphertexts are encrypted by plaintexts that satisfy a specific input difference or by random numbers. However, the current differential-neural distinguisher seems only effective for limited rounds of ciphertext. Therefore, a short high-probability classical differential $\Delta S \rightarrow \Delta P$ is prepended before the differential-neural distinguisher to increase the number of rounds for key recovery attacks.

Gohr [12] showed that the Residual Network [14] could capture the non-randomness of the distribution of output pairs when the input pairs of round-reduced Speck32/64 meet a specific difference. As a result, 6, 7, and 8-round differential-neural distinguishers were trained, and 11, 12-round key recovery attacks for Speck32/64 were achieved by combining a 2-round classical differential. There may be two directions to improve differential-neural cryptanalysis. One is to use a longer classical differential prepended on top of the differential-neural distinguisher. Bao *et al.* [3] generalized the concept of neutral bits and searched for (conditional) simultaneous neutral bit-set with a higher probability for more rounds of the classical differential. Thus, Bao *et al.* devised a new 13-round key recovery attack for Speck32/64 with the same differential-neural distinguisher proposed in [12]. The other is to study the effective differential-neural distinguisher with more rounds. Chen *et al.* [9] and Benamira [5] *et al.* almost simultaneously proposed the method of using multiple-ciphertext pairs instead of single-ciphertext pairs (in Gohr's work) as input of the neural network, both improved the accuracy of the 6, 7-round differential-neural distinguisher of Speck32/64. Bao *et al.* [3] used the Dense Network [16] and the Squeeze-and-Excitation Network [15] to train differential-neural distinguisher, obtained 9, 10, and 11-round differential-neural distinguisher and devised a 16-round key recovery attack for Simon32/64. To obtain more information from the ciphertext, we made some improvements for differential-neural cryptanalysis, as listed below.

Our contribution. The contributions of this work include the following:

- We have developed an enhanced differential-neural distinguisher by modifying the network architecture. To achieve this, we have introduced an Inception module composed of multiple-parallel convolutional layers before the Residual Network. Incorporating the Inception module is to capture a broader range of information across various dimensions within the ciphertext pairs. We have also made some adjustments to the convolutional kernel size according to the round functions of the cipher. As a result of these improvements, we have observed enhanced accuracy in the 6, 7, and 8-round differential-neural distinguishers, and we have trained a new 9-round differential-neural distinguisher for Speck32/64. Similarly, we have improved the accuracy of the 9, 10, and 11-

round distinguishers and developed a new 12-round differential-neural distinguisher for SIMON32/64. Tables 2 and 5 present the results of the differential-neural distinguisher for SPECK32/64 and SIMON32/64, respectively.

- To execute a key recovery attack successfully, each ciphertext pair in multiple-ciphertext pairs acquired through classical differentials must exhibit the same difference. Taking inspiration from Gohr's work on the combined response of the differential-neural distinguisher, we leverage neutral bits with a probability of one to generate multiple-plaintext pairs. Subsequently, we encrypt these multiple-plaintext pairs, resulting in the generation of multiple-ciphertext pairs.

- We successfully implemented several key recovery attacks using the improved differential-neural distinguisher with a classical differential. We reduce the time and data complexity of the key recovery attack for the 13-round SPECK32/64. In particular, we successfully implemented a 14-round key recovery attack by exhaustively guessing a 1-round subkey, marking the first instance of a 14-round key recovery attack in differential-neural cryptanalysis. For SIMON32/64, we can implement a 17-round key recovery attack using deep learning methods for the first time. In addition, we reduce the time complexity of the 16-round key recovery attack. Detailed experimental comparisons are shown in Table 1. Source codes are available in `https://github.com/CryptAnalystDesigner/NeuralDistingsuisherWithInception.git`.

Organization. Section 2 gives the preliminary on the distinguisher model, the key recovery attack process, and generalized neutral bits. Section 3 introduces the training method and the result for SPECK32/64. We introduce data generation, experimental environment, and complexity calculation in Section 4. Section 5 presents the details of our 13, 14-round key recovery attacks for SPECK32/64. Section 6 introduces the training method and the result for SIMON32/64. Section 7 exhibits the details of our 16, 17-round key recovery attacks for SIMON32/64. We conclude the paper in Section 8.

2 Preliminary

2.1 Brief Description of SPECK32/64 and SIMON32/64

Let ω be the word size (the number of bits of a word), and the block size can be denoted as L bits, where $L = 2\omega$. Let (x_r, y_r) be the left and right branches of a state after encryption of r rounds, k_i the subkey of i rounds. Denote the bitwise XOR by $\oplus$, the addition modulo 2^ω by $\boxplus$, the bitwise AND by $\cdot$, the bitwise right rotation by $\ggg$, and the bitwise left rotation by $\lll$.

SPECK32/64 and SIMON32/64 are members in the lightweight block cipher family SPECK and SIMON [4]. The round function (out of 22) of SPECK32/64 takes a 16-bit subkey k_i and a state consisting of two 16-bit words (x_i, y_i) as

Table 1. Summary of key recovery attacks on SPECK32/64 and SIMON32/64

Target	r	Dist.	Conf.	Time	Data	Succ. Rate	Key Space	Ref.
SPECK 32/64	13	$\mathcal{DD}$	1+8+4	2^{57}	2^{25}	–	2^{64}	[10]
		$\mathcal{DD}$	1+8+2+2	$2^{50.16}$	$2^{31.13}$	63%	2^{64}	[8]
		$\mathcal{DD}$	2+8+3	$2^{55.58}$	$2^{24.26}$	–	2^{64}	[11]
		$\mathcal{ND}$	1+3+8+1	$2^{48.67+2.4\star}$	2^{29}	82%	2^{63}	[3]
		$\mathcal{ND}$	1+3+8+1	$2^{46.05+2.92}/2^{41.44+2.92\dagger}$	2^{27}	21%	2^{63}	Sect.5.2
	14	$\mathcal{DD}$	1+9+4	$2^{62.47}$	$2^{30.47}$	–	2^{64}	[18]
		$\mathcal{DD}$	1+9+2+2	$2^{60.99}$	$2^{31.75}$	63%	2^{64}	[8]
		$\mathcal{DD}$	2+9+3	$2^{60.58}$	$2^{30.26}$	76%	2^{64}	[11]
		$\mathcal{ND}$	1+3+8+1+1	$2^{62.05+2.92}/2^{57.44+2.92\dagger}$	2^{27}	21%	2^{63}	Sect.5.3
SIMON 32/64	16	$\mathcal{DD}$	2+12+2	$2^{26.48}$	$2^{29.48}$	62%	2^{64}	[2]
		$\mathcal{ND}$	1+3+11+1	$2^{41.81+2.4\star}$	2^{21}	49%	2^{64}	[3]
		$\mathcal{ND}$	1+3+11+1	$2^{40.12+2.92}/2^{33.60+2.92\dagger}$	2^{22}	80%	2^{64}	Sect.7.1
	17	$\mathcal{ND}$	1+4+11+1	$2^{47.25+2.92}/2^{40.64+2.92\dagger}$	2^{28}	9%	2^{64}	Sect.7.2
	18	$\mathcal{DD}$	1+13+4	$2^{46.00}$	$2^{31.2}$	63%	2^{64}	[1]
	19	$\mathcal{DD}$	2+13+4	$2^{34.00}$	$2^{31.5}$	-	2^{64}	[7]
	21	$\mathcal{DD}$	4+13+4	$2^{55.25}$	$2^{31.0}$	-	2^{64}	[20]

1. $\mathcal{DD}$: differential distinguisher; $\mathcal{ND}$: differential-neural distinguisher; $-$: Not available;
2. *: 2.4 is the ratio of the time required for key recovery attacks using CPU and GPU in [3]; †: 2.92 is the ratio in our device.
3. We list two values of time complexity, $T_{\mathcal{ND}}$ and $T_{\mathcal{DD}}$. The $T_{\mathcal{ND}}$ is compared with $\mathcal{ND}$; the $T_{\mathcal{DD}}$ is compared with $\mathcal{DD}$. For the reason for using two values of time complexity, please refer to Section 4.3.

input. The state of the next round (x_{i+1}, y_{i+1}) is computed as follows:

$$x_{i+1} := ((x_i \ggg 7) \boxplus y_i) \oplus k_i, y_{i+1} := (y_i \lll 2) \oplus x_{i+1}.$$

The round function (out of 32) of SIMON32/64 takes a 16-bit subkey k_i and a state consisting of two 16-bit words (x_i, y_i) as input. The next round state (x_{i+1}, y_{i+1}) is computed as follows:

$$x_{i+1} := (x_i \lll 1) \cdot (x_i \lll 8) \oplus (x_i \lll 2) \oplus y_i \oplus k_i, y_{i+1} := x_i.$$

2.2 The Model of Differential-Neural Distinguisher for SPECK32/64

The model of differential-neural distinguisher in [12,5,9] is almost identical except for the input. Thus, we introduce these models collectively. The differential-neural distinguisher is a supervised model that distinguishes whether ciphertexts are encrypted by plaintexts that satisfy a specific input difference or by random numbers. Given m plaintext pairs $\{(P_{i,0}, P_{i,1}), i \in [0, m-1]\}$ and target cipher SPECK32/64, the resulting ciphertext pairs $\{(C_{i,0}, C_{i,1}), i \in [0, m-1]\}$ are regarded as an instance. Note that $m = 1$ in [12], $m \in \{1, 5, 10, 50, 100\}$ in [5], and $m \in \{2, 4, 8, 16\}$ in [9]. Each instance will be attached with a label Y:

$$Y = \begin{cases} 1, \text{ if } P_{i,0} \oplus P_{i,1} = \Delta, i \in [0, m-1] \\ 0, \text{ if } P_{i,0} \oplus P_{i,1} \neq \Delta, i \in [0, m-1] \end{cases}$$

where Δ = (0x0040, 0x0000). If Y is 1, this instance is sampled from the target distribution and defined as a positive example. Otherwise, this instance is sampled from a uniform distribution and defined as a negative example. A large number of instances need to be trained in neural networks. when the neural network can obtain a stable accuracy higher than 0.5 on a test set, it can effectively distinguish whether ciphertexts are encrypted by plaintexts that satisfy a specific input difference or by random numbers. The model of differential-neural distinguisher can be described as:

$$\begin{gathered} \Pr(Y = 1 \mid X_0, \ldots, X_{m-1}) = F\left(f(X_0), \cdots, f(X_{m-1}), \varphi(f(X_0), \cdots, f(X_{m-1}))\right) \\ X_i = (C_{i,0}, C_{i,1}), i \in [0, m-1] \\ \Pr(Y = 1 \mid X_0, \cdots, X_{m-1}) \in [0, 1] \end{gathered}$$

where $f(X_i)$ represents the basic features of a ciphertext pair X_i, and $\varphi(\cdot)$ is the derived features, and $F(\cdot)$ is the new posterior probability estimation function.

The network architecture for training differential-neural distinguisher contains several modules described in Fig. 1. The input layer of the neural network consisting of multiple-ciphertext pairs is arranged in a $[m, \omega, \frac{2L}{\omega}]$ array, where L represents the block size of the target cipher, and ω is the size of a basic unit. For example, L is 32 and ω is 16 for SPECK32/64. Module 1 is the initial with-1 convolution layer that intends to make learning bit-sliced functions such as bitwise addition. Module 2 is the Residual Network. *Conv* stands for one-dimensional convolution $Conv1D$ with $N_f = 32$ filters, and $k_s = 3$ is the size of the convolution kernel. The number of module 2 is determined by the experiment. The prediction head comprises modules 3, 4, and the output layer. FC is a fully connected layer that has $d_1 = 64$ or $d_2 = 64$ neurons. BN is the batch normalization layer. *Relu* and *Sigmoid* are two different activation functions.

2.3 Inception Network

Generally speaking, the safest way to improve network performance is to increase the width and depth of the network, which also has side effects. First, a deeper and wider network often means many parameters. When the amount of data is small, the trained network is easy to overfit, and when the depth of the network is deep, it is difficult to train and easy to cause greater errors. The two side effects of disappearance restrict the development of deep and wide convolutional neural networks, and the Inception network solves these two problems very well.

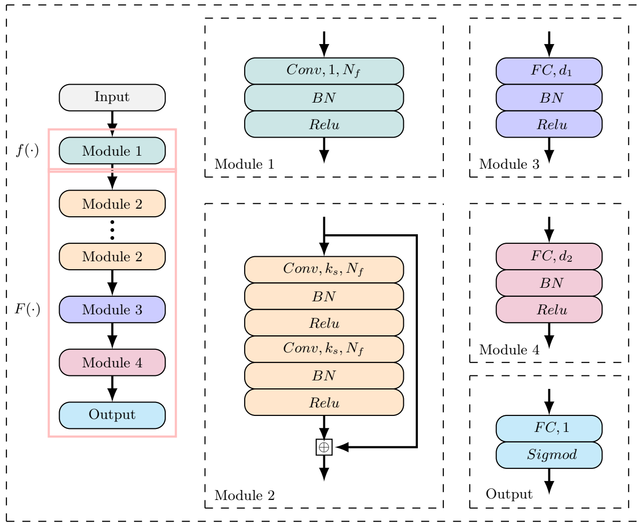


Fig. 1. The network architecture for training differential-neural distinguisher

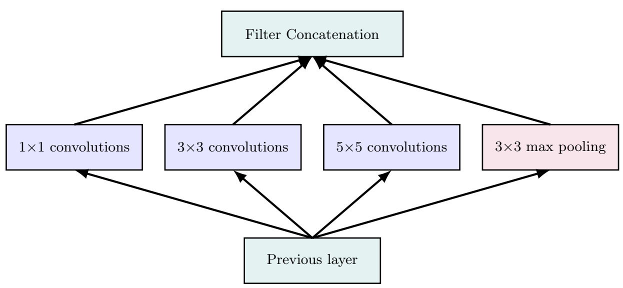


Fig. 2. Inception Module

One of the modules in the Inception network architecture is as follows in Fig. 2: in the same layer, there are 1×1, 3×3, 5×5 convolution and pooling layers, respectively, and the convolution operation and the pooling layer are pooled using filters. Padding is used in all operations to ensure that the output is the same size and the output results are all integrated after these operations. The feature of this module is that in the same layer, different features of the input of the previous layer are collected by using the above-mentioned filters of different sizes and performing pooling operations. This increases the width of the network and uses these different-sized filters and pooling operations to extract different features from the previous layer.

2.4 Differential-neural Cryptanalysis

Gohr [12] proposed a framework for differential neural cryptanalysis dedicated to recovering the last two rounds of subkeys for Speck. We decrypt the ciphertext using guessed subkey and use the differential-neural distinguisher to estimate the distance between the guessed subkey and the real key.

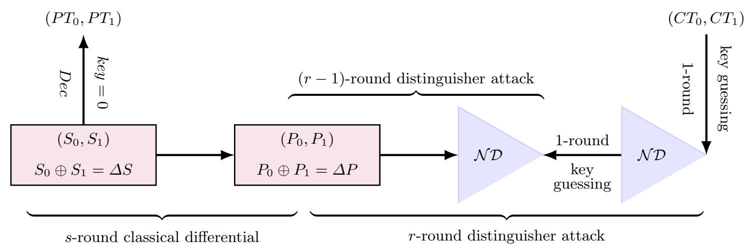


Fig. 3. $(1 + s + r + 1)$-round key recovery attack of differential-neural cryptanalysis

The overall processing of a key recovery attack based on the differential-neural distinguisher is shown in Fig. 3, where $\mathcal{ND}$ is the trained differential-neural distinguisher, (PT_0, PT_1) is plaintext pairs and (CT_0, CT_1) is ciphertext pairs. The $(1 + s + r + 1)$-round key recovery attack employs a r-round main and $(r - 1)$-round helper differential-neural distinguisher trained using input pairs with difference ΔP. A short s-round classical differential $(\Delta S \to \Delta P)$ with probability denoted by 2^{-p} is prepended on top of the differential-neural distinguisher to increase the number of the rounds of key recovery attack. To ensure the existence of data pairs satisfying the difference ΔP after s-round encryption, about $c \cdot 2^p$ (denoted by n_{cts}) data pairs with the difference ΔS are required according to the probability of difference propagation, where c is a small constant. Neutral bits of the s-round classical differential is used to expand each data pair to a structure of n_b data pairs. The n_{cts} structures of the data pairs are decrypted in one round with 0 as the subkey to get the plaintext structures because the nonlinear operation occurs before the addition of keys for Speck and Simon. All plaintext structures are encrypted to obtain the corresponding ciphertext structures. Each ciphertext structure is used to select a candidate of the subkey by the r-round main differential-neural distinguisher based on a variant of Bayesian optimization. The usage of ciphertext structures is also highly selective by using a standard exploration-exploitation technique, namely *Upper Confidence Bounds*. Each ciphertext structure is assigned a priority according to the score of the recommended subkeys and the visited times. Without exhaustively performing trail decryption, the key search policy depends on the expected response of the differential-neural distinguisher upon wrong-key decryption. The *wrong key response profile* is to recommend new candidate values from previous candidate values while minimizing the weighted Euclidean distance in a BayesianKeySearch Algorithm [12].

2.5 Combined Response and Neutral Bits

As the number of encryption rounds increases, the accuracy of the differential-neural distinguisher decreases. To reduce the impact of the misjudgment of the single prediction of the distinguisher, Gohr used the combined response of the differential-neural distinguisher in ciphertext structure with the same distribution, which can be satisfied by neutral bits [12]. The primary notion of neutral bits can be interpreted as follows.

Definition 1 (Neutral bits of a differential, NB[6]). *Let $\Delta_{in} \to \Delta_{out}$ be a differential with input difference Δ_{in} and output difference Δ_{out} of a r-round encryption F^r. Let (P, P') be the input pair and $(C, C' \mid C = F^r(P), C' = F^r(P'))$ be the output pair, where $P \oplus P' = \Delta_{in}$. If $C \oplus C' = \Delta_{out}$, (P, P') is said to be conforming the differential $\Delta_{in} \to \Delta_{out}$. Let $e_0, e_1, \ldots, e_{n-1}$ be the standard basis of $\mathbb{F}_2^n$. Let i be an index of a bit (starting from 0). The i-th bit is a neutral bit for the differential $\Delta_{in} \to \Delta_{out}$, if $(P \oplus e_i, P' \oplus e_i)$ is also a confirming pair for any confirming pair (P, P').*

The responses $v_{i,k}$ from the differential-neural distinguisher on ciphertext pairs in the ciphertext structure (of size n_b) are combined using the Formula $s_k = \sum_{i=0}^{n_b-1} \log_2\left(\frac{v_{i,k}}{1-v_{i,k}}\right)$ and s_k is used as the score of a recommended subkey. The score s_k plays a decisive role in the execution time and success rate of the attack. The number of instances with the same distribution should be sufficiently large to enhance the distinguishing ability of the low-accuracy differential-neural distinguisher. However, neutral bits of the nontrivial classical differential are scarce. Therefore, probabilistic neutral bits (PNB) are exploited in [12]. Some probabilistic neutral bits, simultaneous-neutral bit-sets (SNBS), conditional (simultaneous-) neutral bit(-set)s (CSNBS), and switching bits for adjoining differentials (SBfADs) were found in [3] (refer to Appendix A).

3 Differential-Neural Distinguishers for Round-Reduced Speck32/64

3.1 Network Architecture

The overall structure of our neural network for training the differential-neural distinguisher is depicted in Figure 4. Our neural network comprises four main components: an input layer that incorporates multiple ciphertext pairs, an initial convolutional layer consisting of four parallel convolutional layers, a residual tower that consists of multiple convolutional neural networks with two layers each, and a prediction head that comprises multiple fully connected layers.

Input Representation. When we have the output of the r-th round denoted as $(C, C') = (x_r || y_r, x'_r || y'_r)$, it can directly computes (y_{r-1}, y'_{r-1}) without knowledge of the $(r-1)$-th subkey, using the round function of Speck. Consequently, the neural network can process data in the format of $(x_r, x'_r, y_r, y'_r, y_{r-1}, y'_{r-1})$.

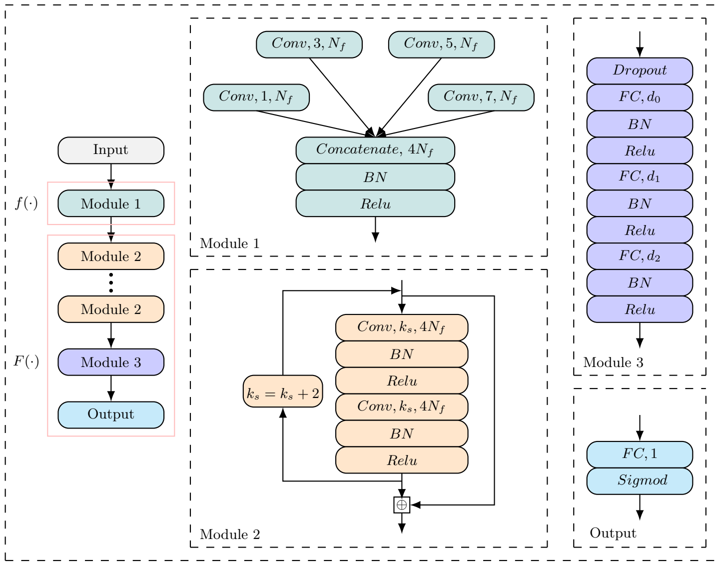


Fig. 4. The network architecture of our distinguisher for SPECK32/64

To accommodate this data format, the input layer of the neural network comprises m ciphertext pairs, where each pair consists of $3L$ units arranged in a $[m, \omega, \frac{3L}{\omega}]$ array. For SPECK32/64, the values of L and ω are set as 32 and 16 respectively.

Initial Convolution (Module 1). The input layer is linked to the initial convolutional layer, which consists of four convolutional layers with $N_f = 32$ channels using various kernel sizes. Similar to GoogLeNet's Inception [19], the outputs of the four convolutional layers are concatenated along the channel dimension. Batch normalization is applied to the concatenated outputs. Afterward, rectifier nonlinearity is applied to the batch normalized outputs, and the resulting matrix $[m, \omega, 4N_f]$ is then forwarded to the convolutional blocks layer.

Convolutional Blocks (Module 2). Each convolutional block consists of two layers of $4N_f$ filters. Each block applies first the convolution with kernel size $k_s = 3$, then a batch normalization, and finally a rectifier layer. At the end of the convolutional block, a skip connection is added to the output of the final rectifier layer of the block to the input of the convolutional block. It transfers the result to the next block. After each convolutional block, the kernel size increases by 2. The number of convolutional blocks is 5 in our model (determined by experiment).

Prediction Head (Module 3 and Output). The prediction head consists of three hidden layers and one output unit. Before the first hidden layer, we add a dropout layer to prevent model overfitting. The three fully connected layers comprise $d_0 = 512$, $d_1 = 64$, and $d_2 = 64$ units, followed by the batch normalization and rectifier layers. The final layer consists of a single output unit using the activation function *Sigmoid*.

Rationale. Firstly, we make adjustments to the model's input data format. By utilizing the ciphertext from the last round, we can calculate the right half of the penultimate round's ciphertext without knowledge of the (r-1)-th subkey, using the round function of SPECK32/64. Second, considering the cyclic shift operation and modulo addition in the round function, we introduce convolution operations with widths of 3, 5, and 7 to try to capture the information that may exist in adjacent bits. This allows us to capture multidimensional features, taking inspiration from the Inception block in GoogLeNet[19]. Thirdly, to enhance the receptive field of the convolutions, we increase the size of the convolutional kernels by 2 as the Residual Network's depth grows. The size of the convolution kernel is increased by 2 each time to ensure that the size of the convolution layer is an odd number. Additionally, we incorporate a dropout layer before the fully connected layer to enhance the network's generalization ability.

3.2 The Training of Differential-Neural Distinguisher

The accuracy serves as the paramount metric to evaluate the performance of the differential-neural distinguisher. Subsequently, the subsequent training procedure was conducted to validate the efficacy of our network architecture.

Data Generation. Training and test sets were generated using the Linux random number generator to obtain uniformly distributed keys K_i and multiple-plaintext pairs $\{(P_{i,j,0}, P_{i,j,1}), j \in [0, m-1]\}$ with the input difference $\Delta =$ (0x0040, 0x0000) and a vector of binary-valued labels Y_i. During the production of training or test sets for r-round SPECK32/64, the multiple-plaintext pairs were then encrypted for r rounds if $Y_i = 1$, while otherwise, the second plaintext of the pairs was replaced with a freshly generated random plaintext and then encrypted for r rounds.

Remark 1. We use two different numbers of datasets to train differential-neural distinguisher. In [12], the training and test sets include N and M instances, consisting of a ciphertext pair in an instance, total N and M ciphertext pairs, respectively. In [9], the training set and test set include ${}^{N}\!/_{m}$ and ${}^{M}\!/_{m}$ instances, and each instance includes m ciphertext pairs; that is, the total numbers of ciphertext pairs used are N and M. To ensure a fair comparison, we used ${}^{N}\!/_{m}$ and ${}^{M}\!/_{m}$ instances as training and test sets. However, it may lead to overfitting (see *Remark 2* for details). To overcome this problem, we also use N and M instances as training test and test set, which consists of m ciphertext pair, that is, total $N \times m$ and $M \times m$ ciphertext pairs, respectively.

Basic Training Scheme. We conducted the training for 20 epochs in the dataset for $N = 10^7$ and $M = 10^6$. The batch size processed by the dataset is adjusted according to the parameter m to maximize GPU performance. Optimization was performed against mean square error loss plus a small penalty based on L2 weights regularization parameter $c = 10^{-5}$ using the Adam algorithm [17]. A cyclic learning rate schedule was applied, setting the learning rate l_i for epoch i to $l_i = \alpha + \frac{(n-i) \mod (n+1)}{n} \cdot (\beta - \alpha)$ with $\alpha = 10^{-4}, \beta = 2 \times 10^{-3}$ and $n = 9$. The networks obtained at the end of each epoch were stored, and the best network by validation loss was evaluated against a test set.

Training (8-9)-round Distinguishers Using Staged Train Method. We use several stages of pretraining to train an 8-round differential-neural distinguisher for Speck32/64. First, we use our 7-round distinguisher to recognize 5-round Speck32/64 with the input difference (0x8000, 0x804a) (the most likely difference to appear three rounds after the input difference (0x0040, 0x0000)). The training was done in 10^7 instances for 20 epochs with a learning rate of 10^{-4}. Then we trained the distinguisher to recognize 8-round Speck32/64 with the input difference (0x0040, 0x0000) by processing 10^7 freshly generated instances for 10 epochs with a learning rate of 10^{-4}. Finally, the learning rate was reduced to 10^{-5} after processing another 10^7 new instances for 10 epochs. For the 9-round distinguisher, the overall training method is the same. The only difference is the use of an 8-round distinguisher to identify 5-round Speck32/64 with the input difference (0x850a,0x9520) (the most likely difference to appear four rounds after the input difference (0x0040,0x0000).

3.3 Result

Test Accuracy. We summarize the accuracy of 6, 7, 8-, and 9-round differential-neural distinguisher compared to [12,9,5] in Table 2. Also, we list the accuracy (Acc), the true positive rate (TPR), and the true negative rate (TNR) tested on newly generated data in Table 3. The 6, and 7-round distinguishers were trained using the basic training method, while the 8, and 9-round distinguishers were trained using the staged training method. If the accuracy is greater than 0.5, it is considered effective. The "N/m" column results in accuracy using N/m and M/m instances as training and test sets, respectively. The "N" column results from accuracy using N and M instances as the training and test sets, respectively. From Table 2, the accuracy of our differential-neural distinguisher was significantly improved in both the "N/m" column and the "N" column compared to [13,9,5]. There are some differences in accuracy under the two experiments caused by overfitting of the model; see *Remark 2* for details.

In Table 2, except for Benamira's result [5], other results are directly trained by the neural network. In [5], the value of m is $\{1, 5, 10, 50, 100\}$. But in order to facilitate the key recovery attack, we set the value of m at the power of 2, that is, $\{2, 4, 8, 16\}$. When $m = 1$, the case of Benamira and Gohr is the same. In [5], they use two ways to train and evaluate the differential-neural distinguisher with multiple ciphertext pairs. The straightforward one (Averaging Method) is to evaluate the neural distinguisher score for each element of the

Table 2. Summary accuracy of distinguisher on Speck32/64 using different number of instances

r	m=2				m=4				m=5		
	[9]	[13]*	$^N/_m$	N	[9]	[13]*	$^N/_m$	N	[5]	$^N/_m$	N
6	0.8613	0.877	0.8771	0.8773	0.931	0.950	0.9468	0.9497	0.9541	0.9623	0.965
7	0.6393	0.663	0.6663	0.6649	0.6861	0.725	0.7194	0.7283	0.735	0.7436	0.7491
8	-	0.521	-	-	-	0.530	0.5329	0.5428	-	-	-
r	m=8				m=10			m=16			
	[9]	[13]*	$^N/_m$	N	[5]	$^N/_m$	N	[9]	[13]*	$^N/_m$	N
6	0.9562	0.990	0.9868	0.9895	0.99	0.9915	0.9939	0.9802	1.000	0.9969	0.9992
7	0.7074	0.801	0.7991	0.8106	0.808	0.8243	0.8341	0.6694	0.883	0.8759	0.8963
8	-	0.543	0.5347	0.5590	-		-	-	0.560	0.5566	0.5854
9		-	-	0.5024	-		-	-	-	-	0.5050

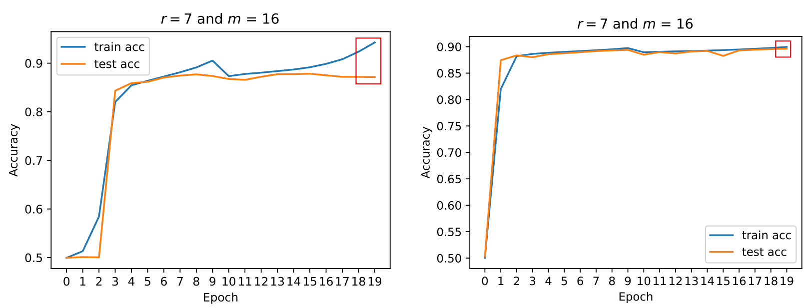
1. *: combining the scores of multiple distinguishers trained by using single-ciphertext pair under independence assumption.

Table 3. Acc, TPR, TNR of distinguisher on Speck32/64 using N instances

m	r	Acc	TPR	TNR	m	r	Acc	TPR	TNR
	6	0.8768	0.8448	0.9087		6	0.9495	0.9429	0.9559
2	7	0.6635	0.6240	0.7029	4	7	0.7291	0.7048	0.7532
	8	-	-	-		8	0.5428	0.5318	0.5537
	6	0.9897	0.9862	0.9931		6	0.9992	0.9988	0.9996
8	7	0.8103	0.8024	0.8184	16	7	0.8958	0.8906	0.9009
	8	0.5562	0.5434	0.5690		8	0.5853	0.5704	0.6002
	9	0.5016	0.2122	0.7915		9	0.5045	0.5175	0.4915

multiple ciphertext pairs and then take the results' median. The second (2D-CNN Method) is to consider the whole multiple ciphertext pairs as a single input for a neural network. These two methods are theoretically equivalent, but the second method to train the differential-neural distinguisher will have lower accuracy. Benamira use two methods to get the accuracy of the 6-round distinguisher. When $m = 5$, the accuracy of the two methods is 0.9541 and 0.9327, respectively; When $m = 10$, the accuracy of the two methods is 0.99 and 0.977, respectively, in [5]. We can see that the accuracy of the distinguisher obtained by the second method is lower than that of the first method. However, the accuracy of the 7-round distinguisher is obtained by the first method in [5]. This means the accuracy obtained should be even lower if the distinguisher is trained directly using a neural network.

Remark 2. Why do we train a differential-neural distinguisher with two different numbers of data? For the sake of fairness of comparison, we must use $^N/_m$ and $^M/_m$ instances as training and test sets (see *Remark 1*). However, from Fig. 5a, we can see that the difference between training and test accuracy is relatively significant. In other words, using $^N/_m$ and $^M/_m$ instances as training and test set to train a neural network will suffer overfitting, especially when the number of rounds r and m is large. For more overfitting phenomena, please refer to Appendix C. However, the training and test accuracy are almost equal in Fig. 5b. Therefore, using N and M instances as the training set and a test set to train the differential-neural distinguisher can avoid overfitting, speed up the model convergence, and improve the model accuracy to a certain extent. Due to the overfitting phenomenon, the accuracy of the distinguisher will be low, which also affects our training of more rounds of differential-neural distinguisher.



(a) Using $^N/_m$ instances (b) Using N instances

Fig. 5. Overfitting with the different number of instances.

Wrong Key Response Profile. The key search policy depends on an important observation that the expected response of the distinguisher upon wrong-key decryption will rely on the bitwise difference between the guessed subkey and the real subkey. The *wrong key response profile*, which is precomputed by using 3000 ciphertext pairs for each guessed subkey, is used to recommend new candidate values for the key from previous candidate values by minimizing the weighted Euclidean distance as the criteria in a BayesianKeySearch Algorithm. It recommends a set of subkeys and provides their scores without exhaustively performing the trial decryption. The μ and δ represent the empirical mean and standard deviation of the distinguisher score over these 3000 ciphertext pairs.

The abscissa in Fig. 6 and 7 is the difference between the guessed subkey and the real subkey, and the ordinate is the score obtained by putting the ciphertext pairs obtained after decrypting multiple ciphertext pairs (encrypted with the same key) with the guessed subkey into the differential-neural distinguisher. If the Hamming weight of the difference between guessed and real subkey is small, the distinguisher will score high; otherwise, the score will be low. From Fig. 6

and 7, we can see that when the Hamming weight of the key difference is small, our differential-neural distinguisher scores higher than that of the Gohr's distinguisher in the same abscissa. We can get lower scores when the Hamming weight is larger. In other words, our distinguisher is better able to judge the Hamming distance between the real subkey and the guessed subkey, thus recommending candidate subkeys that are closer to the real subkey.

Moreover, it can be observed that the score of the distinguisher is higher when the difference between the guessed key and the real key belongs to {16384, 32768, 49152}, where the vertical coordinates of "*" correspond to these positions. This means that when the 14th and 15th bits of the subkey are guessed incorrectly, it has little effect on the score of the distinguisher. Therefore, it is possible to reduce the key guessing space by not guessing these two bits, and this feature is also used in [12] to accelerate the key recovery attack.

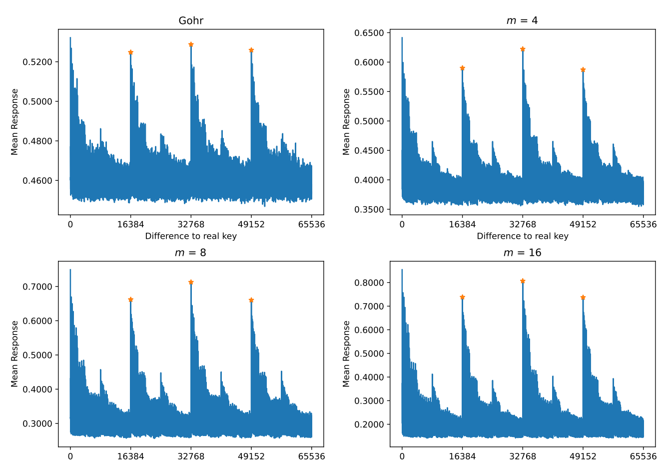


Fig. 6. Wrong key response profile for 7-round Speck32/64 using N instances

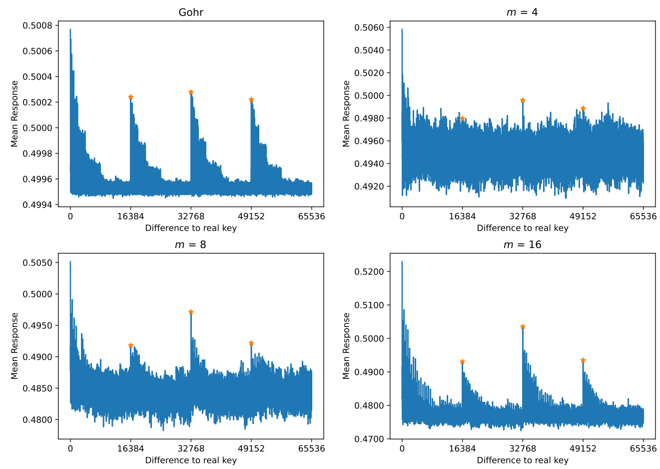


Fig. 7. Wrong key response profile for 8-round SPECK32/64 using N instances

4 From Differential-Neural Distinguisher to Key Recovery Attack

4.1 Generation of Same Distributed Data

During the training of the differential-neural distinguisher, we take multiple ciphertext pairs as an instance and then put the instance into the neural network. In the positive example, we require multiple ciphertext pairs of an instance to be obtained by encrypting multiple plaintext pairs satisfying the same difference Δ_P.

After obtaining a differential-neural distinguisher, we add a classical differential $\Delta_S \rightarrow \Delta_P$ before the differential-neural distinguisher to increase the number of rounds for the key recovery attack. However, the classical differential is probabilistic. During the key recovery attack, even if multiple plaintext pairs have the same difference Δ_S of an instance, the difference between multiple ciphertext pairs of an instance may not be the same after propagation through the classical differential. We need to take certain measures to make the multiple ciphertext pairs of an instance have the same difference after the classical difference propagation.

Neutral bits can solve the problem of obtaining multiple ciphertext pair that satisfies the same difference after propagation through the classical differential, ensuring we can launch key recovery attacks. We randomly generate a plaintext pair satisfying the initial difference Δ_S and flip the ciphertext bits corresponding to $\log_2 m$ neutral bits to obtain m plaintext pairs. The m ciphertext pairs obtained by the propagation of the m plaintext pairs through the classical differential have the same difference.

4.2 Experimental Environment and Data Complexity

Following to the settings in [12], we count a key guess as successful if the last round key was guessed correctly and the second round key is at the hamming distance at most two of the real key. The experiment is conducted by Python 3.7.15 and Tensorflow 2.5.0 in Ubuntu 20.04. The device information is Intel(R) Xeon(R) Gold 6226R*2 with 2.90GHz, 256GB RAM, and NVIDIA RTX2080Ti 12GB*5. In our implementation, the performance is not constrained by the speed of neural network evaluation when using a high-performance graphics card. Instead, the total number of iterations on the ciphertext structures determines the limiting factor. The experimental parameters for key recovery attacks are denoted below.

1. n_{kg}: the number of times to guess the subkey k_r involved in the condition.
2. n_{cts}: the number of ciphertext structure.
3. n_b: the number of ciphertext pairs in each ciphertext structure, that is, $2^{|NB|}$.
4. n_{it}: the total number of iterations on the ciphertext structures.
5. c_1 and c_2: the cutoffs with respect to the scores of the recommended last subkey and second to last subkey, respectively.
6. n_{byit1}, n_{cand1} and n_{byit2}, n_{cand2}: the number of iterations and the number of key candidates within each iteration in the BayesianKeySearch Algorithm to guess each of the last and the second to last subkeys, respectively.

Theoretical Data Complexity. During a key recovery attack, the classical differential or neutral bits used may need to satisfy some conditions involving the key. The data complexity of the experiment is calculated using the formula $n_{kg} \times n_b \times n_{ct} \times m \times 2$. The data complexity is calculated as theoretical values. In the actual experiment, when the accuracy of the differential-neural distinguisher is high, the key can be recovered quickly and successfully. Not all the data is used, so the actual data complexity is lower than the theoretical.

4.3 Experimental Time Complexity.

How to reasonably calculate the time complexity and compare it fairly with previous work has always been a difficult task.

- In classical differential cryptanalysis, researchers usually calculate the number of full encryption (the number of rounds of attack) required to perform a key recovery attack as the time complexity of the attack [10,1,7,11,8].

- In differential-neural cryptanalysis, Gohr[12] uses GPU to train differential-neural distinguisher but uses CPU to implement key recovery attacks. Therefore, he estimates that a highly optimized, fully SIMD-parallelized implementation of Speck32/64 could perform brute force key search at a speed of about 2^{28} keys per second per core. Later, Bao [3] *et al.* used GPU to accelerate key recovery attacks for 13-round Speck32/64. For a fair comparison with previous work, they also used 2^{28} as the benchmark for encryption speed and estimated the ratio ($\text{Ratio}_{cpu/gpu}$) of the time required for key recovery attacks using CPU and GPU, where $Ratio = 2^{2.4}$ in [3].

To ensure a fair comparison, we conduct multiple key recovery attacks and determine the average running time (rt) as the representative time for each experiment. Additionally, we compute the success rate (sr) of the key recovery attack by dividing the number of successful experiments by the total number of experiments conducted. In addition, classical differential cryptanalysis methods all use the CPU to execute key recovery attack programs. Part of the program of the key recovery attack of differential-neural cryptanalysis needs to be executed by GPU. For a fairer comparison, we conducted several experiments to calculate the ratio of the time required to perform key recovery attacks using CPU and GPU in our device. The specific results are shown in Table 4.

Table 4. Calculate the ratio of time required for key recovery attack using CPU and GPU

R	Conf.	c_1	c_2	n_{cts}	n_{it}	N_e	Device	rt	$\text{Ratio}_{cpu/gpu}$	sr	Ref.
11	1+2+7+1	5	10	100	500	100	CPU	299.26	$2^{0.39}$	53%	[12]
							GPU	227.05		51%	[12]
12	1+3+7+1	7	10	2^{11}	2^{12}	40	CPU	4272.91	$2^{2.92}$	100%	This work
							GPU	564.67		100%	This work

From Table 4, we can see that the ratio of the time required for the key recovery attack on 11 rounds of speck32/64 using the distinguisher trained by Gohr [12] on the CPU and GPU is $2^{0.39}$. When performing a key recovery attack on 12-round speck32/64 using our trained distinguisher, the ratio is $2^{2.92}$. We choose the maximum value of the two ratios $2^{2.92}$ as the final value of $\text{Ratio}_{cpu/gpu}$. According to different attack methods, we give two different time complexity calculation methods.

- **Comparison with Differential-neural Cryptanalysis**: We follow the setting of Gohr [12] and Bao [3] et al. to calculate the time complexity of the key recovery attack. we used 2^{28} as the benchmark for encryption speed. Then the calculation formula of time complexity $T_{\mathcal{ND}}$ is $n_{kg} \times rt \times 2^{28} \times 2^{2.92}$.
- **Comparison with Classical Differential Cryptanalysis**: we utilize a 2^{32} plaintext to evaluate our device's encryption speed E_s. For SPECK32/64, each core can execute approximately $2^{27.09}$ 1-round encryption per second, i.e., $E_s = 2^{27.09}$; For SIMON32/64, each core can execute approximately $2^{25.48}$ 1-round encryption per second, i.e., $E_s = 2^{25.48}$. Then the calculation formula of time complexity $T_{\mathcal{DD}}$ is $n_{kg} \times rt \times \frac{E_s}{R} \times 2^{2.92}$, where R is the number of rounds for key recovery attack.

5 Key Recovery Attack on Round-Reduced Speck32/64

In this section, we demonstrate the effectiveness of our differential-neural distinguisher in enhancing the performance of key recovery attacks. We adopt a similar framework as described in previous works [12,3], with the key difference being the utilization of the differential-neural distinguisher trained on N instances. For a more complete key recovery attack procedure, please refer to Appendix B. We have successfully reduced the time and data complexity of 13-round key recovery attacks for Speck32/64 by employing an improved differential-neural distinguisher. Moreover, we leverage the outcomes of the 13-round key recovery attack to estimate the complexity involved in a 14-round key recovery attack.

5.1 Key Recovery Attack on 13-round Speck32/64

Combining a 3-round classical differential with an 8-round differential-neural distinguisher, we examine how far a practical attack can go on 13-round Speck32/64 in this subsection.

Experiment 1: The components of key recovery attack $\mathcal{A}_I^{\text{Speck}13R}$ of 13-round Speck32/64 are shown as follows.

1. 3-round classical differential (0x8020, 0x4101)$\rightarrow$ (0x0040, 0x0000);
2. generalized neutral bits of generating multiple-ciphertext pairs: $\{[22], [20], [13]\}$; generalized neutral bits of combined response of differential-neural distinguisher:$\{[5,28], [15,24], [12,19], [6,29], [6,11,12,18], [4,27,29], [14,21], [0,8,31], [30]\}$ (refer to Table 10).
3. 8-round differential-neural distinguisher $\mathcal{ND}^{\text{Speck}_{8r}}$ under difference (0x0040, 0x0000) and its wrong key response profiles $\mathcal{ND}^{\text{Speck}_{8r}} \cdot \mu$ and $\mathcal{ND}^{\text{Speck}_{8r}} \cdot \delta$.
4. 7-round differential-neural distinguisher $\mathcal{ND}^{\text{Speck}_{7r}}$ under difference (0x0040, 0x0000) and its wrong key response profiles $\mathcal{ND}^{\text{Speck}_{7r}} \cdot \mu$ and $\mathcal{ND}^{\text{Speck}_{7r}} \cdot \delta$.

In the beginning, we guess three key bits of k_0, that is $k_0[7]$, $k_0[5] \oplus k_0[14]$, and $k_0[15] \oplus k_0[8]$ because of the 3-round differential (refer to Table 9) and four key bits $k_0[12] \oplus k_0[5]$, $k_0[1]$, $k_0[2] \oplus k_0[11]$, $k_0[11] \oplus k_0[4]$ to employ four conditional neutral bits (refer to Table 10). Thus, n_{kg} is 2^7. However, to make the experimental verification economic, we tested the core of the attack with the seven conditions being fulfilled only. The concrete parameters used in our 13-round key recovery attack $\mathcal{A}^{\text{Speck}13R}$ are listed below.

$n_{kg} = 2^7$	$m = 8$	$n_b = 2^9$	$n_{cts} = 2^{11}$
$n_{it} = 2^{12}$	$c_1 = 5, c_2 = -86$	$n_{byit1} = n_{byit2} = 5$	$n_{cand1} = n_{cand2} = 32$

Because the prepended classical differential is valid when the keys fulfil $k_2[12] \neq k_2[11]$, we tested only for these valid keys, and the presented attack works for 2^{63} keys. The data complexity is $2^7 \times 2^9 \times 2^{11} \times 8 \times 2 = 2^{31}$ plaintexts. The core of the attack was examined in 100 trials; there are 41 successful trials, that is, $sr = 41\%$. The average run time of every trail in our server is 15223.20s. Thus, the time complexity $T_{\mathcal{ND}} = n_{kg} \times rt \times 2^{28} \times 2^{2.92} = 2^7 \times 15223.21 \times 2^{28} \times 2^{2.92} = 2^{48.89+2.92}$; $T_{\mathcal{DD}} = n_{kg} \times rt \times \frac{E_s}{R} \times 2^{2.92} = 2^7 \times 15223.21 \times \frac{2^{27.09}}{13} \times 2^{2.92} = 2^{44.28+2.92}$.

5.2 Using Two Classical Differentials and SBfADs for 13-round Speck32/64

Bao *et al.* found that the output of the classical differential matters to the differential-neural distinguisher but not the input difference. Hence, more than one differential can be prepended to a differential-neural distinguisher if they share the same output difference. Multiple such classical differentials can share some neutral bits. Using such classical differentials might enable data reuse, thus slightly reducing data complexity. Also, they employ SBfADs to save one guessed key bit and reduce time and data complexity by half compared to the CSNBS. To further exploit the capabilities of our differential-neural distinguisher, we again implement a 13-round key recovery attack using two classical differentials and SBfADs.

Experiment 2: The components of key recovery attack $\mathcal{A}_{II}^{\text{Speck}13R}$ of 13-round Speck32/64 are shown as follows.

1. 3-round classical differentials (0x8020, 0x4101) $\rightarrow$ (0x0040, 0x0000) and (0x8060, 0x4101) $\rightarrow$ (0x0040, 0x0000) ;
2. generalized neutral bits of generating multiple-ciphertext pairs: $\{[20], [13], [12, 19]\}$; generalized neutral bits of combined response of differential-neural distinguisher:$\{[22], [14, 21], [6, 29], [30], [0, 8, 31], [5, 28], [15, 24], [4, 27, 29]\}$ (refer to Table 10 and one SBfADs [21]).
3. 8-round differential-neural distinguisher $\mathcal{ND}^{\text{Speck}_{8r}}$ under difference (0x0040, 0x0000) and its wrong key response profiles $\mathcal{ND}^{\text{Speck}_{8r}} \cdot \mu$ and $\mathcal{ND}^{\text{Speck}_{8r}} \cdot \delta$.
4. 7-round differential-neural distinguisher $\mathcal{ND}^{\text{Speck}_{7r}}$ under difference (0x0040, 0x0000) and its wrong key response profiles $\mathcal{ND}^{\text{Speck}_{7r}} \cdot \mu$ and $\mathcal{ND}^{\text{Speck}_{7r}} \cdot \delta$.

In the beginning, we only guess two key bits of k_0, that is, $k_0[7]$ and $k_0[15] \oplus k_0[8]$ because of two 3-round classical differentials and two key bits $k_0[12] \oplus k_0[5]$, $k_0[1]$ to employ two CSNBS (refer to Table 10). Because we use two classical differentials (0x8020, 0x4101) $\rightarrow$ (0x0040, 0x0000) and (0x8060, 0x4101) $\rightarrow$ (0x0040, 0x0000), the condition $k_0[5] \oplus k_0[14]$ is unnecessary (refer to Table 9). Besides, we use SBfADs [21] instead of CSNBS $[6, 11, 12, 18]$, the key bit $k_0[2] \oplus k_0[11]$ does not need to be guessed. Note that although we used SNBS $[4, 27, 29]$, we did not use key condition $k_0[11] \oplus k_0[4]$ to increase its probability, because 0.672 is already high enough. Thus, $n_{kg} = 2^4$. However, to make the experimental verification economic, we tested the core of the attack with the four conditions being fulfilled only. The concrete parameters used in our 13-round key recovery attack $\mathcal{A}_{II}^{\text{Speck}13R}$ are listed below.

$n_{kg} = 2^4$	$m = 8$	$n_b = 2^{8+1}$	$n_{cts} = 2^{11}$
$n_{it} = 2^{12}$	$c_1 = 8, c_2 = -500$	$n_{byit1} = n_{byit2} = 5$	$n_{cand1} = n_{cand2} = 32$

Because the prepended classical differential is valid when the keys fulfill $k_2[12] \neq k_2[11]$, we tested only for these valid keys, and the presented attack works for 2^{63} keys. The data complexity is $2^4 \times 2^9 \times 2^{11} \times 8 \times 2/2 = 2^{27}$ plaintexts. Note that the data complexity is divided by 2 because if we use two classical differences, one can generate half of the required data pairs for free. The core of the attack was examined in 100 trials; there are 21 successful trials, that is, $sr = 21\%$. The average run time of every trail in our server is 17011.22s. Thus, the time complexity $T_{\mathcal{ND}} = n_{kg} \times rt \times 2^{28} \times 2^{2.92} = 2^4 \times 17011.22 \times 2^{28} \times 2^{2.92} = 2^{46.05+2.92}$; $T_{\mathcal{DD}} = n_{kg} \times rt \times \frac{E_s}{R} \times 2^{2.92} = 2^4 \times 17011.22 \times \frac{2^{27.09}}{13} \times 2^{2.92} = 2^{41.44+2.92}$.

5.3 Brute Force Guessing of 1-round Subkey for 14-round Speck32/64

Under the current framework, there are two ways to increase the number of rounds of key recovery attacks. One is to increase the length of the classical differential, and the other is to increase the length of the difference-neural distinguisher, but it seems that these two paths are not feasible under the current results.

- **Increase the Length of the Classical Differential.** In the 13-round key recovery attack, the probability of the 3-round sub-optimal classical differentials we use is 2^{-12}, and the complexity of the core attack is about $2^{42.07}$. We use the MILP model to search for 4-round optimal classical differential path $(0x1488, 0x1008) \rightarrow (0x0040, 0x0000)$ with a probability of 2^{-17}. When we use a 4-round classical differential to implement a 14-round key recovery attack, the time complexity of the core attack will exceed 2^{47} under ideal circumstances, which is beyond our current computing equipment. Also, the 4-round classical differential doesn't have enough neutral bits to use for key recovery attacks.

- **Increase the Length of the Differential-neural Distinguisher.** Although we trained a 9-round differential-neural distinguisher, its accuracy was too low, and we can not implement a 14-round key recovery attack. According to our calculations, the accuracy of DDT should be 0.5089 and 0.5138 in the case of $m = 8$ and $m = 16$, respectively. In addition, it has been proved that the differential-neural distinguisher can learn more features than DDT, so the accuracy of the differential-neural distinguisher should be higher than that of DDT. Suppose we can further improve the accuracy of the 9-round differential-neural distinguisher. In that case, it will be possible to directly implement a 14-round key recovery attack, which we will continue to study.

Fortunately, we can estimate the time complexity of a 14-round key recovery attack. Notably, the time complexity for our 13-round key recovery attack is sufficiently low. By employing a brute force approach to guess one round of subkey, we can subsequently execute a 14-round key recovery attack. The time complexity of the 14-round attack is determined by multiplying the time complexity of the 13-round attack by 2^{16}. According to two different complexity calculation formulas, we can give the time complexity of the 14-round key recovery attack.

1. The time complexity $T_{\mathcal{ND}} = 2^{16} \times n_{kg} \times rt \times 2^{28} \times 2^{2.92} = 2^{16} \times 2^4 \times 17011.22 \times 2^{28} \times 2^{2.92} = 2^{62.05+2.92}$. Note that this time complexity exceeds that of brute force key search.

2. The time complexity $T_{\mathcal{DD}} = 2^{16} \times n_{kg} \times rt \times \frac{E_s}{R} \times 2^{2.92} = 2^{16} \times 2^4 \times 17011.22 \times \frac{2^{27.09}}{13} \times 2^{2.92} = 2^{57.44+2.92}$.

6 Differential-Neural Distinguishers on Round-Reduced SIMON32/64

In ASIACRYPT 2022, Bao *et al.* used the Dense Network and Squeeze-and-Excitation Network to train 9, 10, and 11-round differential-neural distinguishers for SIMON32/64 [3]. We give the accuracy of the 9, 10, 11, and 12-round differential-neural distinguisher for SIMON32/64 using modified network architecture in case of multiple ciphertext pairs.

6.1 Network Architecture

The network architecture of the differential-neural distinguisher used to train SIMON32/64 is generally similar to that of SPECK32/64. Based on the round function of SIMON32/64, we modify the number of convolutional layers and the size of the convolutional kernel in the Inception module and use a GlobalAveragePooling layer instead of three fully connected layers. The overall architecture is shown in Fig 8.

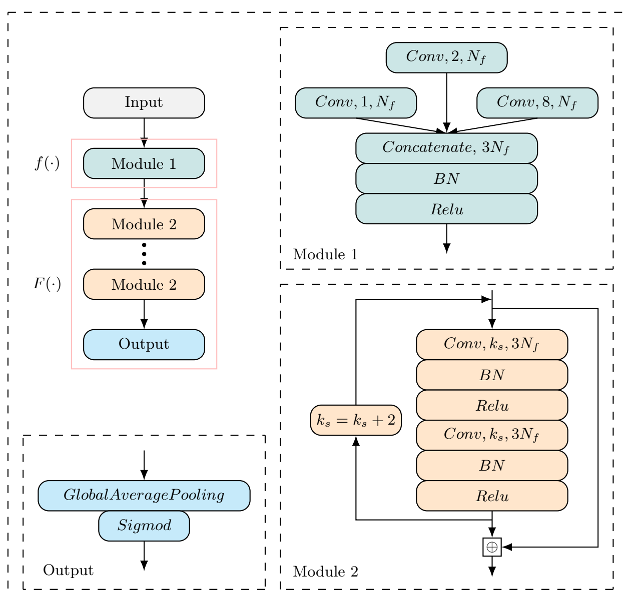


Fig. 8. The network architecture of our distinguisher for SIMON32/64

Input Representation. Based on the round function, $(y_{r-1} \oplus y'_{r-1})$ can be obtained without knowing the $(r-1)$-th subkey for SIMON32/64. Thus, the neural network accepts data of the form $(x_r, x'_r, y_r, y'_r, y_{r-1} \oplus y'_{r-1})$. The input layer has m ciphertext pairs consisting of $2.5L$ units likewise arranged in a $[m, \omega, \frac{2.5L}{\omega}]$ array, where $L = 32, \omega = 16$ for SIMON32/64.

Initial Convolution (Module 1). The input layer is connected to the initial convolutional layer, which comprises three convolution layers with $N_f = 32$ channels of kernel sizes 1, 2, and 8. The three convolution layers are concatenated at the channel dimension. Batch normalization is applied to the output of the concatenate layers. Finally, rectifier nonlinearity is applied to the output of batch normalization, and the resulting $[m, \omega, 3N_f]$ matrix is passed to the convolutional blocks layer.

Convolutional Blocks (Module 2). The convolutional blocks layer of the differential-neural distinguisher model is the same as SPECK32/64, except that the shape of the input is different.

Prediction Head (Output). The prediction head consists of a GlobalAveragePooling layer and an output unit using a *Sigmoid* activation function.

Rationale. The network architecture for training differential-neural differentiators for SPECK32/64 and SIMON32/64 is essentially the same, except for the prediction head. Using multiple fully connected layers can lead to an overload of model parameters and increase the model training time. Therefore, an attempt is made to use a GlobalAveragePooling layer instead of multiple fully connected layers.

6.2 The Training of Differential-Neural Distinguisher

Training Using the Basic Scheme. We used two different numbers of training and test sets to train the neural network. The first situation of experiments uses $^N/_m$ and $^M/_m$ instances as training and test sets. The second situation of experiments uses N and M instances as the training and test sets. Refer to the basic training scheme of SPECK32/64 in Sect. 3 for training parameters. By modifying the network architecture and using the basic training scheme, we trained the differential-neural distinguisher to recognize the output pairs of 9, 10, and 11-round SIMON32/64 with the input difference (0x0000, 0x0040).

Training Using the Staged Training Method. A 12-round differential-neural distinguisher of SIMON32/64 was trained using several pre-training stages. First, we use our 11-round distinguisher to recognize a 9-round SPECK32/64 with the input difference (0x0440, 0x0100) (the most likely difference to appear three rounds after the input difference (0x0000, 0x0040)). The training was done in 10^7 instances for 20 epochs with a learning rate of 10^{-4}. Then we trained the distinguisher to recognize 12-round SPECK32/64 with the input difference (0x0000, 0x0040) by processing 10^7 freshly generated instances for 10 epochs with a learning rate of 10^{-4}. Finally, the learning rate was dropped to 10^{-5} after processing another 10^7 new instances each.

6.3 Result

Test Accuracy. We summarize the accuracy of 9, 10, 11, and 12-round differential-neural distinguisher in Table 5. In addition, we list Acc, TPR, and TNR tested on newly generated data in Table 6. The 9, 10, and 11-round distinguisher was trained using the basic training method. Using the staged training method, a 12-round distinguisher was derived from an 11-round distinguisher. We also use two different numbers of instances to train the differential-neural distinguisher for SIMON32/64, both for the fair comparison of the experiment and to solve the problem of overfitting (refer to Appendix C). Compared to gohr's work [13], the length of the distinguisher was increased by 1 round, although the accuracy of the differential-neural distinguisher was not improved.

Table 5. Summary accuracy of the distinguisher on SIMON32/64 using different number of instances

r	m=1		m=2			m=4		
	[3]	[13]*	[13]*	N/m	N	[13]*	N/m	N
9	0.6532	0.661	0.730	0.7251	0.7240	0.811	0.7991	0.8095
10	0.5629	0.567	0.598	0.5917	0.5907	0.637	0.6239	0.6339
11	0.5173	0.520	0.529	0.5193	0.5240	0.543	0.5343	0.5387
12	-	-	-	-	-	-	-	-

r	m=8			m=16		
	[13]*	N/m	N	[13]*	N/m	N
9	0.896	0.8774	0.8958	0.963	0.9344	0.9630
10	0.692	0.6716	0.6900	0.761	0.7230	0.7608
11	0.563	0.5441	0.5591	0.589	0.5339	0.5878
12	-	-	0.5152	-	-	0.5225

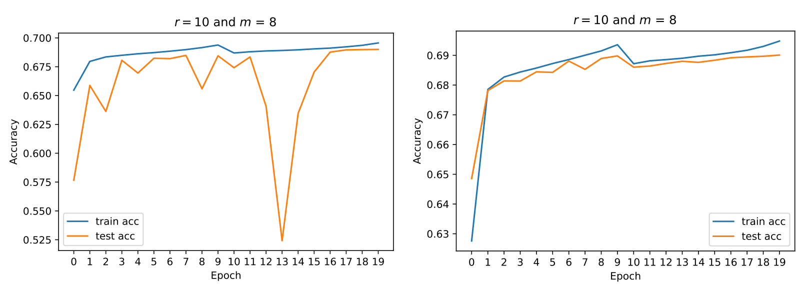
1. *: combining the scores of multiple distinguishers trained by using single-ciphertext pair under independence assumption.
2. The combination method used in [13] is the same as that in [5], which evaluates the distinguisher score of each element in multiple ciphertext pairs, and then takes the median of the results. When the distinguisher is directly trained with multiple ciphertext pairs, the accuracy of the distinguisher will be reduced. For details, see Sect.3.3

Accuracy Fluctuations. The neural network designed for SPECK32/64 to train the differential-neural distinguisher uses multiple fully connected layers as prediction heads, resulting in too many model parameters, making training time

Table 6. Acc, TPR, TNR of distinguisher on SIMON32/64 using N instances

m	r	Acc	TPR	TNR	m	r	Acc	TPR	TNR
2	9	0.7234	0.7022	0.7446	4	9	0.8101	0.7821	0.8382
	10	0.5902	0.4779	0.7022		10	0.6347	0.5570	0.7124
	11	0.5155	0.8618	0.1663		11	0.5344	0.7139	0.3550
	12	-	-	-		12	-	-	-
8	9	0.8956	0.8811	0.9101	16	9	0.9630	0.9595	0.9664
	10	0.6908	0.6846	0.6953		10	0.7619	0.7207	0.8030
	11	0.5592	0.5946	0.5239		11	0.5871	0.5338	0.6406
	12	0.5159	0.5324	0.4995		12	0.5218	0.5445	0.4991

too long. To address this issue, we use a GlobalAveragePooling layer instead of the fully connected layer as the prediction head in the neural network of SIMON32/64. From Fig. 9a and 9b, it can be seen that using the GlobalAveragePooling layer will cause relatively large fluctuations in the accuracy, but the accuracy has almost no effect.



(a) Using GlobalAveragePooling layer (b) Using fully connected layers

Fig. 9. Using different number of instances with different prediction head

Wrong Key Response Profile. The *wrong key response profile* for our 9, 10, 11, and 12-round differential-neural distinguisher is shown in Fig. 10. As we can see from the figure when the Hamming Distance between the real key and the guessed key is smaller, the score of the differential-neural distinguisher is higher, and vice versa, it is smaller. Judging how far the guessed key deviates from the real key is easy. This shows that our differential-neural distinguisher can effectively distinguish between ciphertext and random numbers. In addition, it can be observed that the score of the distinguisher is higher when the difference between the guessed key and the real key belongs to {8192, 16384, 24576, 32768, 40960, 49152, 57344}, where r = 10 and 11, the vertical coordinates of "*" correspond to these positions. This means that when the 13th, 14th, and 15th bits of the subkey are guessed incorrectly, it has little effect on the score of the distinguisher. Therefore, reducing the key guessing space by not guessing these three bits and accelerating the key recovery attack is possible.

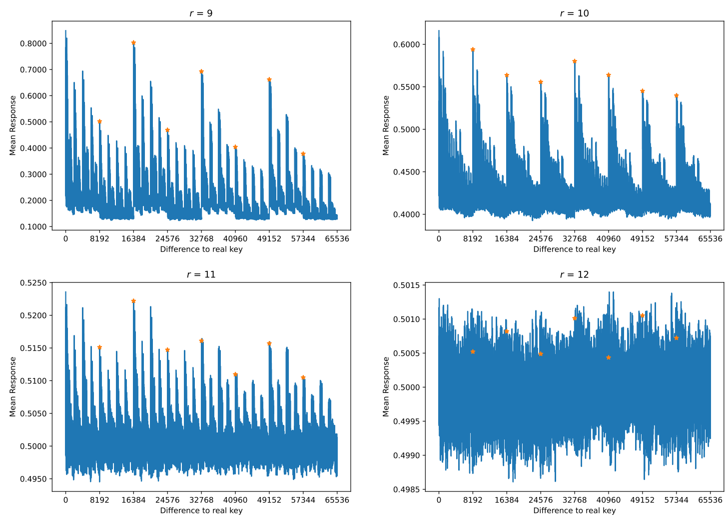


Fig. 10. Wrong key response profile for SIMON32/64 using N instances where $m = 8$

7 Key Recovery Attack on Round-Reduced SIMON32/64

Under a similar procedure to the key recovery attack on SPECK32/64, trained differential-neural distinguishers can be prepended with a classical differential to perform key recovery attacks for SIMON32/64. We improved the 16-round and devised the first 17-round differential-neural-distinguisher-based key recovery attacks on SIMON32/64. Note that based on the features found from the *wrong key response profile*, we did not guess the 14th and 15th bits of the subkey in the key recovery attack for 16-round SIMON32/64, and 13th, 14th, and 15th of the subkey in the key recovery attack for 17-round SIMON32/64. Sadly, we could not successfully perform the 18-round key recovery attack for SIMON32/64 due to the lack of a sufficient number of generalized neutral bits.

7.1 Key Recovery Attack on 16-round SIMON32/64

To verify the performance of our 11-round differential-neural distinguisher, the following experiments were carried out in this subsection.

Experiment 3: The components of the key recovery attack $\mathcal{A}^{\textsc{Simon}16R}$ of the 16-round Simon32/64 are shown below.

1. 3-round classical differential (0x0440, 0x1000)→(0x0000, 0x0040);
2. generalized neutral bits of generating multiple-ciphertext pairs: $\{[2],[3],[4]\}$; generalized neutral bits of combined response of differential-neural distinguisher:$\{[6],[8],[9],[10],[18],[22],[0,24],[12,26]\}$ (refer to Table 11).
3. 11-round differential-neural distinguisher $\mathcal{ND}^{\textsc{Simon}_{11r}}$ under difference (0x0000, 0x0040) and its wrong key response profiles $\mathcal{ND}^{\textsc{Simon}_{11r}}\cdot\mu$ and $\mathcal{ND}^{\textsc{Simon}_{11r}}\cdot\delta$.
4. 10-round differential-neural distinguisher $\mathcal{ND}^{\textsc{Simon}_{10r}}$ under difference (0x0000, 0x0040) and its wrong key response profiles $\mathcal{ND}^{\textsc{Simon}_{10r}}\cdot\mu$ and $\mathcal{ND}^{\textsc{Simon}_{10r}}\cdot\delta$.

In the beginning, we guess two key bits of k_0, that is, $k_0[1]$ and $k_0[3]$, because of the 3-round differential, the conditions for correct pairs are $x_1[1]=x'_1[1]=0$ and $x_1[3]=x'_1[3]=0$. Thus, n_{kg} is 2^2. However, to make the experimental verification economic, we tested the core of the attack with the two conditions being fulfilled only. The concrete parameters used in our 16-round key recovery attack $\mathcal{A}^{\textsc{Simon}16R}$ are listed below.

$n_{kg}=2^2$	$m=8$	$n_b=2^8$	$n_{cts}=2^8$
$n_{it}=2^9$	$c_1=37, c_2=70$	$n_{byit1}=n_{byit2}=5$	$n_{cand1}=n_{cand2}=32$

The data complexity is $2^2\times2^8\times2^8\times8\times2=2^{22}$ plaintexts. In total, 100 trials are running, and there are 80 successful trials, that is, $sr=80\%$. The average run time of the experiment is 1115.42s. Thus, the time complexity $T_{\mathcal{ND}}=n_{kg}\times rt\times2^{28}\times2^{2.92}=2^2\times1115.42\times2^{28}\times2^{2.92}=2^{40.12+2.92}$; $T_{\mathcal{DD}}=n_{kg}\times rt\times\frac{E_s}{R}\times2^{2.92}=2^2\times1115.42\times\frac{2^{25.48}}{16}\times2^{2.92}=2^{33.60+2.92}$.

7.2 Key Recovery Attack on 17-round Simon32/64

Combining a 4-round classical differential with an 11-round differential-neural distinguisher, we examine how far a practical attack can go on 17-round Simon32/64 in this subsection.

Experiment 4: The components of the key recovery attack $\mathcal{A}^{\textsc{Simon}17R}$ of the 17-round Simon32/64 are shown below.

1. 4-round classical differential (0x1000, 0x4440)→ (0x0000, 0x0040);
2. generalized neutral bits of generate multiple-ciphertext pairs: $\{[2],[6],[12,26]\}$ (refer to Table 12); generalized neutral bits of combined response of differential-neural distinguisher:$\{[10,14,28]\}$ and five neutral bits conditioned on $x[1,15],x[15,13],x[13,11],x[11,9],x[9,7]$ (refer to Table 13).
3. 11-round differential-neural distinguisher $\mathcal{ND}^{\textsc{Simon}_{11r}}$ under difference (0x0000, 0x0040) and its wrong key response profiles $\mathcal{ND}^{\textsc{Simon}_{11r}}\cdot\mu$ and $\mathcal{ND}^{\textsc{Simon}_{11r}}\cdot\delta$.
4. 10-round differential-neural distinguisher $\mathcal{ND}^{\textsc{Simon}_{10r}}$ under difference (0x0000, 0x0040) and its wrong key response profiles $\mathcal{ND}^{\textsc{Simon}_{10r}}\cdot\mu$ and $\mathcal{ND}^{\textsc{Simon}_{10r}}\cdot\delta$.

In the beginning, we guess two key bits of k_0, that is, $k_0[3]$ and $k_0[5]$, because of the 4-round differential, the conditions for correct pairs are $x_1[5] = x'_1[5] = 0$ and $x_1[3] = x'_1[3] = 0$; and six key bits $k_0[1], k_0[15], k_0[13], k_0[11], k_0[9], k_0[7]$ for employing five conditional neutral bits (refer to Table 13). Thus, n_{kg} is 2^8. However, to make the experimental verification economic, we tested the core of the attack with the eight conditions being fulfilled only. The concrete parameters used in our 17-round key recovery attack $\mathcal{A}^{\text{SIMON}17R}$ are listed below.

$n_{kg} = 2^8$	$m = 8$	$n_b = 2^6$	$n_{cts} = 2^{11}$
$n_{it} = 2^{12}$	$c_1 = 15, c_2 = 65$	$n_{byit1} = n_{byit2} = 5$	$n_{cand1} = n_{cand2} = 32$

The data complexity is $2^8 \times 2^6 \times 2^{11} \times 8 \times 2 = 2^{29}$ plaintexts. In total, 100 trials are running, and 9 successful trials, that is $sr = 9\%$. The average running time of the experiment is 2435.63s. Thus, the time complexity $T_{\mathcal{ND}} = n_{kg} \times rt \times 2^{28} \times 2^{2.92} = 2^8 \times 2435.63 \times 2^{28} \times 2^{2.92} = 2^{47.25+2.92}$; $T_{\mathcal{DD}} = n_{kg} \times rt \times \frac{E_s}{R} \times 2^{2.92} = 2^8 \times 2435.63 \times \frac{2^{25.48}}{17} \times 2^{2.92} = 2^{40.64+2.92}$.

8 Conclusion

In this paper, we designed a new network architecture to train differential-neural distinguisher, which uses multiple-parallel convolution layers to capture the features of different dimensions of cryptographic algorithms. As a result, we improved the accuracy and obtained the distinguisher with more rounds. Focusing on the problem under the same data distribution, we propose a solution using neutral bits with probability one to generate multiple-plaintext pairs. The combination of multiple improvements reduces the time complexity and increases the number of rounds of the key recovery attack.

References

1. Abed, F., List, E., Lucks, S., Wenzel, J.: Differential cryptanalysis of round-reduced simon and speck. In: FSE. Lecture Notes in Computer Science, vol. 8540, pp. 525–545. Springer (2014)
2. AlKhzaimi, H., Lauridsen, M.M.: Cryptanalysis of the SIMON family of block ciphers. IACR Cryptol. ePrint Arch. p. 543 (2013)
3. Bao, Z., Guo, J., Liu, M., Ma, L., Tu, Y.: Enhancing differential-neural cryptanalysis. In: ASIACRYPT (1). Lecture Notes in Computer Science, vol. 13791, pp. 318–347. Springer (2022)
4. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK families of lightweight block ciphers. IACR Cryptol. ePrint Arch. p. 404 (2013)
5. Benamira, A., Gérault, D., Peyrin, T., Tan, Q.Q.: A deeper look at machine learning-based cryptanalysis. In: EUROCRYPT (1). Lecture Notes in Computer Science, vol. 12696, pp. 805–835. Springer (2021)
6. Biham, E., Chen, R.: Near-collisions of SHA-0. In: CRYPTO. Lecture Notes in Computer Science, vol. 3152, pp. 290–305. Springer (2004)

7. Biryukov, A., Roy, A., Velichkov, V.: Differential analysis of block ciphers SIMON and SPECK. In: FSE. Lecture Notes in Computer Science, vol. 8540, pp. 546–570. Springer (2014)
8. Biryukov, A., Cardoso dos Santos, L., Teh, J.S., Udovenko, A., Velichkov, V.: Meet-in-the-filter and dynamic counting with applications to speck. In: International Conference on Applied Cryptography and Network Security. pp. 149–177. Springer (2023)
9. Chen, Y., Shen, Y., Yu, H., Yuan, S.: A new neural distinguisher considering features derived from multiple ciphertext pairs (2021)
10. Dinur, I.: Improved differential cryptanalysis of round-reduced speck. In: Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 8781, pp. 147–164. Springer (2014)
11. Feng, Z., Luo, Y., Wang, C., Yang, Q., Liu, Z., Song, L.: Improved differential cryptanalysis on speck using plaintext structures. Cryptology ePrint Archive (2023)
12. Gohr, A.: Improving attacks on round-reduced speck32/64 using deep learning. In: CRYPTO (2). Lecture Notes in Computer Science, vol. 11693, pp. 150–179. Springer (2019)
13. Gohr, A., Leander, G., Neumann, P.: An assessment of differential-neural distinguishers. Cryptology ePrint Archive (2022)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778. IEEE Computer Society (2016)
15. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: CVPR. pp. 7132–7141. Computer Vision Foundation / IEEE Computer Society (2018)
16. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: CVPR. pp. 2261–2269. IEEE Computer Society (2017)
17. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (Poster) (2015)
18. Song, L., Huang, Z., Yang, Q.: Automatic differential analysis of ARX block ciphers with application to SPECK and LEA. In: ACISP (2). Lecture Notes in Computer Science, vol. 9723, pp. 379–394. Springer (2016)
19. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: CVPR. pp. 1–9. IEEE Computer Society (2015)
20. Wang, N., Wang, X., Jia, K., Zhao, J.: Differential attacks on reduced SIMON versions with dynamic key-guessing techniques. Sci. China Inf. Sci. **61**(9), 098103:1–098103:3 (2018)

A Used Generalized Neutral Bit-sets for Key Recovery Attack

A.1 Generalized Neural Bit-sets for SPECK32/64 [3]

Neutral bit-sets (NB) Used in [12] for 2-round Classical Differential: The signal from the distinguisher will rather be weak. Gohr boosts it by using $|NB|$ probabilistic neutral bits to create from each plaintext pair. A plaintext structure consisting of $2^{|NB|}$ plaintext pairs that are expected to pass the initial 2-round classical differential together. Concretely, neutral bits that are probabilistically neutral are summarized as follows.

Table 7. (Probabilistic) single-bit neutral bit for 2-round Classical Differential (0x0211, 0x0a04)→ (0x0040, 0x0000) of Speck32/64 [12]

NB	Pr.	NB	Pr.	NB	Pr.	NB	Pr.	NB	Pr.	NB	Pr.	NB	Pr.
[20]	1	[21]	1	[22]	1	[14]	0.965	[15]	0.938	[23]	0.812	[7]	0.806
[30]	0.809	[0]	0.763	[8]	0.664	[24]	0.649	[31]	0.644	[1]	0.574		

Simultaneous-neutral bit-sets (SNBS) used in [3] for 2-round classical differential: for the prepended 2-round differential on top of differential-neural distinguisher, Bao *et al.* [3] can experimentally obtain 3 complete NB and 2 SNBS using an exhaustive search. Concretely, for the 2-round differential (0x0211, 0x0a04)→ (0x0040, 0x0000), bit and bit-sets that are (probabilistically) (simultaneous-)neutral are summarized in Table 8.

Table 8. (Probabilistic) SNBS for 2-round Classical Differential (0x0211, 0x0a04)→ (0x0040, 0x0000) of Speck32/64 [3]

NB	Pr.	NB	Pr.	NB	Pr.	NB	Pr.	NB	Pr.	NB	Pr.	NB	Pr.
[20]	1	[21]	1	[22]	1	[9,16]	1	[2,11,25]	1	[14]	0.965	[15]	0.938
[6,29]	0.91	[23]	0.812	[30]	0.809	[7]	0.806	[0]	0.754	[11,27]	0.736	[8]	0.664

Conditional simultaneous-neutral bit-sets (CSNBS) used in [3] for 3-round classical differential: Bao *et al.* found that there are three sufficient conditions for a pair $(x, y), (x', y')$ to conform to the 3-round differential (0x8020, 0x4101)→ (0x0040, 0x0000), summarized in Table 9. Concretely, for the 3-round four sub-optimal differential, bit and bit-sets that are (probabilistically) conditional simultaneous-neutral are summarized in Table 10.

Table 9. Three sufficient conditions conform the 3-round sub-optimal differential [3]

(0x8020, 0x4101) $\downarrow$ (0x0040, 0x0000)	(0x8060, 0x4101) $\downarrow$ (0x0040, 0x0000)	(0x8021, 0x4101) $\downarrow$ (0x0040, 0x0000)	(0x8061, 0x4101) $\downarrow$ (0x0040, 0x0000)
$x[7] = 0$ $x[5] \oplus y[14] = 1$ $x[15] \oplus y[8] = 0$	$x[7] = 0$ $x[5] \oplus y[14] = 0$ $x[15] \oplus y[8] = 0$	$x[7] = 0$ $x[5] \oplus y[14] = 1$ $x[15] \oplus y[8] = 1$	$x[7] = 0$ $x[5] \oplus y[14] = 0$ $x[15] \oplus y[8] = 1$

Table 10. (Probabilistic) (simultaneous-)neutral bit-sets for 3-round differential (0x8020, 0x4101) $\rightarrow$ (0x0040, 0x0000), (0x8060, 0x4101) $\rightarrow$ (0x0040, 0x0000), (0x8021, 0x4101) $\rightarrow$ (0x0040, 0x0000), (0x8061, 0x4101) $\rightarrow$ (0x0040, 0x0000) of SPECK32/64 [3]

	(8020, 4101)		(8060, 4101)		(8021, 4101)		(8061, 4101)		
Bit-set	Pre.	Post.	Pre.	Post.	Pre.	Post.	Pre.	Post.	Condition
[22]	0.995	1.000	0.995	1.000	0.996	1.000	0.997	1.000	-
[20]	0.986	1.000	0.997	1.000	0.996	1.000	0.995	1.000	-
[13]	0.986	1.000	0.989	1.000	0.988	1.000	0.992	1.000	-
[12,19]	0.986	1.000	0.995	1.000	0.993	1.000	0.986	1.000	-
[14,21]	0.855	0.860	0.874	0.871	0.881	0.873	0.881	0.876	-
[6,29]	0.901	0.902	0.898	0.893	0.721	0.706	0.721	0.723	-
[30]	0.803	0.818	0.818	0.860	0.442	0.442	0.412	0.407	-
[0,8,31]	0.855	0.859	0.858	0.881	0.000	0.000	0.000	0.000	-
[5,28]	0.495	1.000	0.495	1.000	0.481	1.000	0.469	1.000	$x[12] \oplus y[5] = 1$
[15,24]	0.482	1.000	0.542	1.000	0.498	1.000	0.496	1.000	$y[1] = 0$
[4,27,29]	0.672	0.916	0.648	0.905	0.535	0.736	0.536	0.718	$x[11] \oplus y[4] = 1$
[6,11,12,18]	0.445	0.903	0.456	0.906	0.333	0.701	0.382	0.726	$x[2] \oplus y[11] = 0$

Among the two 3-round differentials (0x8020,0x4101) $\rightarrow$ (0x0040,0x0000) and (0x8060,0x4101) $\rightarrow$ (0x0040,0x0000) are adjoining differentials. The bit 5 of x (the bit 21 of $x\|y$) is the SBfADs of both pairs. An SBfADs plays the same role as a deterministic unconditional NB, thus is better to be used than probabilistic and conditional NBs. Specifically, employing SBfADs saves one guessed key bit and reduces both time and data complexity by half compared to employing the CSNBS.

A.2 Generalized Neural Bit-sets for SIMON32/64 [3]

For an input pair $((x, y), (x', y'))$ to conform to the 3-round differential (0x0440, 0x1000) $\rightarrow$ (0x0000, 0x0040), one has conditions that

$$\begin{cases} x[1] = x'[1] = 0 \\ x[3] = x'[3] = 0 \end{cases}$$

Table 11. NB and SNBS for 3-round Classical Differential (0x0440, 0x1000)→ (0x0000, 0x0040) of SIMON32/64 [3]

[2]	[3]	[4]	[6]	[8]	[9]
[10]	[18]	[22]	[0,24]	[12,26]	

For an input pair $((x, y), (x', y'))$ to conform to the 4-round differential (0x1000, 0x4440)→(0x0000, 0x0040), one has conditions that

$$\begin{cases} x[5] = x'[5] = 0 \\ x[3] = x'[3] = 0 \end{cases}$$

Table 12. NB and SNBS for 4-round Classical Differential (0x1000, 0x4440)→ (0x0040, 0x0000) of SIMON32/64 [3]

[2]	[6]	[12,26]	[10,14,28]

Table 13. CSNBS for 4-round Classical Differential (0x1000, 0x4440)→ (0x0040, 0x0000) of SIMON32/64 [3]

Bit-set	C.	Bit-set	C.	Bit-set	C.	Bit-set	C.	Bit-set	C.
$x[1,15]$		$x[15,13]$		$x[13,11]$		$x[11,9]$		$x[9,7]$	
[24,10]	00	[22,8]	00	[20]	00	[18,4]	00	[16,8]	00
[24,10,9]	10	[22,8,7]	10	[20,5]	10	[18,4,3]	10	[16,8,1]	10
[24,10,0]	01	[22,8,14]	01	[20,12]	01	[18,4,10]	01	[16]	01
[24,10,9,0]	11	[22,8,7,14]	11	[20,12,5]	11	[18,4,3,10]	11	[16,1]	11

C.: Condition on $x[i, j]$, e.g., $x[i, j] = 10$ means $x[i] = 1$ and $x[j] = 0$.

B Procedure of $(1 + s + r + 1)$-round key recovery attack

The attack procedure is as follows.

1. Initialize variables $Gbest_{key} \leftarrow$ (None, None), $Gbest_{score} \leftarrow -\infty$.
2. For each of the n_{kg} guessed key bits, on which the conditions depend,
   (a) Generate n_{cts} random data with difference ΔP, and satisfying the conditions being conforming pairs (refer to Appendix A).

(b) Using n_{cts} random data and $\log_2 m$ neutral bit with probability one to generate n_{cts} data pairs. Every data pairs have m data.

(c) From the n_{cts} random data pairs, generate n_{cts} structures using the n_b generalized neutral bit.

(d) Decrypt one round using zero as the subkey for all data in the structures and obtain the n_{cts} plaintext structure.

(e) Query for the ciphertexts under $(1 + s + r + 1)$-round Speck or Simon of the $n_{cts} \times n_b \times 2$ plaintext structures, thus obtaining n_{cts} ciphertext structures, denoted by $\{\mathcal{C}_1, \ldots, \mathcal{C}_{n_{cts}}\}$.

(f) Initialize an array $\omega_{\max}$ and an array n_{visit} to record the highest distinguisher score obtained so far and the number of visits have received in the last subkey search for the ciphertext structures.

(g) Initialize variables $best_{score} \leftarrow -\infty$, $best_{key} \leftarrow$ (None, None), $best_{pos} \leftarrow$ None to record the best score, the corresponding best recommended values for the two subkeys obtained among all ciphertext structures and the index of these ciphertext structures.

(h) For j from 1 to n_{it}:

  i. Compute the priority of each of the ciphertext structures as follows: $s_i = \omega_{\max i} + \alpha \cdot \sqrt{\log_2 j / n_{\text{visit}i}}$, for $i \in \{1, \ldots, n_{cts}\}$, and $\alpha = \sqrt{n_{cts}}$; The formula of priority is designed according to a general method in reinforcement learning to achieve automatic exploitation versus exploration trade-off based on Upper Confidence Bounds. It is motivated to focus the key search on the most promising ciphertext structures [12].

  ii. Pick the ciphertext structure with the highest priority score for further processing in this j-th iteration, denote it by $\mathcal{C}$, and its index by idx, $n_{\text{visit}idx} \leftarrow n_{\text{visit}idx} + 1$.

  iii. Run the BayesianKeySearch Algorithm [12] with $\mathcal{C}$, the r-round diffe-rential-neural distinguisher $\mathcal{ND}^r$ and its wrong key response profile $\mathcal{ND}^r \cdot \mu$ and $\mathcal{ND}^r \cdot \sigma$, n_{cand1}, and n_{byit1} as input parameters; obtain the output, that is, a list L_1 of $n_{byit1} \times n_{cand1}$ candidate values for the last subkey and their scores, i.e., $L_1 = \{(g_{1i}, v_{1i}) : i \in \{1, \ldots, n_{byit1} \times n_{cand1}\}\}$.

  iv. Find the maximum $v_{1\max}$ among v_{1i} in L_1, if $v_{1\max} > \omega_{\max idx}$, $\omega_{\max idx} \leftarrow v_{1\max}$.

  v. For each of recommended last subkey $g_{1i} \in L_1$, if the score $v_{1i} > c_1$,

    A. Descrypt the ciphertext in $\mathcal{C}$ using the g_{1i} by one round and obtain the ciphertext structures $\mathcal{C}'$ of $(1+s+r)$-round Speck or Simon.

    B. Run BayesianKeySearch Algorithm with $\mathcal{C}'$, the differential-neural distinguisher $\mathcal{ND}^{r-1}$ and its wrong key response profile $\mathcal{ND}^{r-1} \cdot \mu$ and $\mathcal{ND}^{r-1} \cdot \sigma$, n_{cand2}, and n_{byit2} as input parameters; obtain the output, that is a list L_2 of $n_{byit2} \times n_{cand2}$ candidate values for the last subkey and their scores, i.e., $L_2 = \{(g_{2i}, v_{2i}) : i \in \{1, \ldots, n_{byit2} \times n_{cand2}\}\}$.

    C. Find the maximum v_{2i} and the corresponding g_{2i} in L_2, and denote them by $v_{2\max}$ and $g_{2\max}$.

D. If $v_{2\max} > best_{score}$, update $best_{score} \leftarrow v_{2\max}$, $best_{key} \leftarrow (g_{1i}, g_{2\max})$, $best_{pos} \leftarrow idx$.

vi. If $best_{score} > c_2$, go to Step 2i.

(i) Make a final improvement using VerifierSearch [12] on the value of $best_{key}$ by examining whether the scores of a set of keys obtained by changing at most 2 bits on top of the incrementally updated $best_{key}$ could be improved recursively until no improvement is obtained, update $best_{score}$ to the best score in the final improvement; If $best_{score} > Gbest_{score}$, update $Gbest_{score} \leftarrow best_{score}$, $Gbest_{key} \leftarrow best_{key}$.

3. Return $Gbest_{key}, Gbest_{score}$.

C Overfitting in training differential-neural distinguisher with N/m and N instances

From Fig. 11, 13, 15, 17, 19, we can see that the difference between training accuracy and test accuracy is relatively large. Therefore, using N/m and M/m instances as training and test sets to train a neural network will suffer from overfitting, especially when the number of rounds r and m is large. However, the difference between training accuracy and test accuracy is very small and almost equal in Fig. 12, 14, 16, 18, 20. Therefore, using N and M instances as training and test sets to train the differential-neural distinguisher can avoid overfitting, speed up the model convergence, and improve the model accuracy to a certain extent.

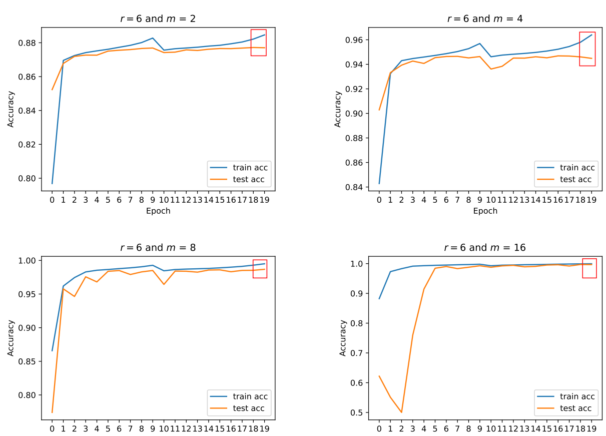


Fig. 11. Using N/m instances for 6-round Speck32/64

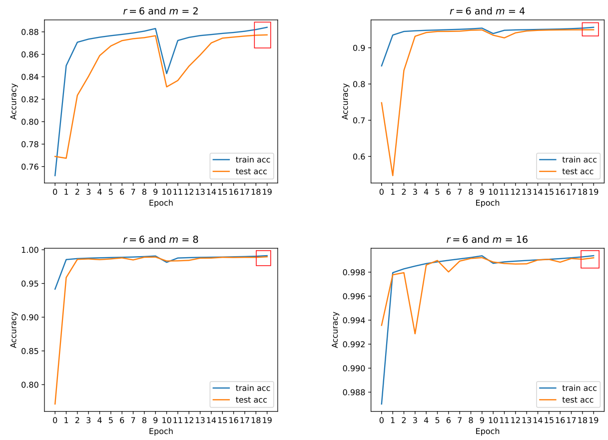


Fig. 12. Using N instances for 6-round SPECK32/64

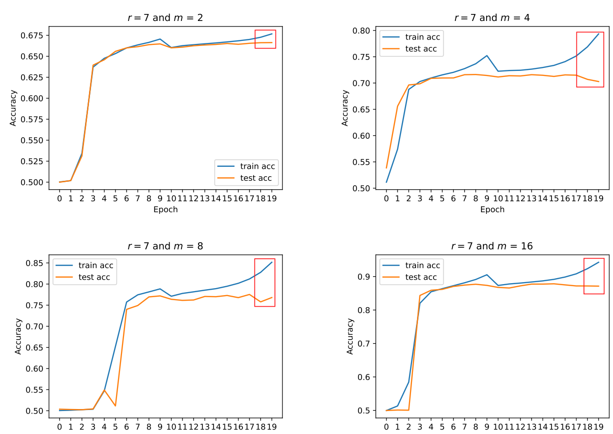


Fig. 13. Using $^N/_m$ instances for 7-round SPECK32/64

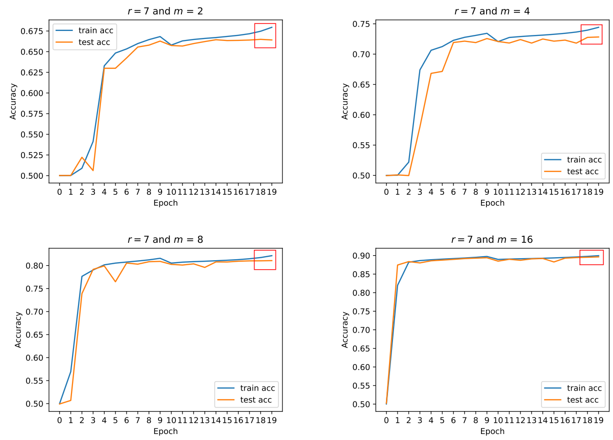


Fig. 14. Using N instances for 7-round Speck32/64

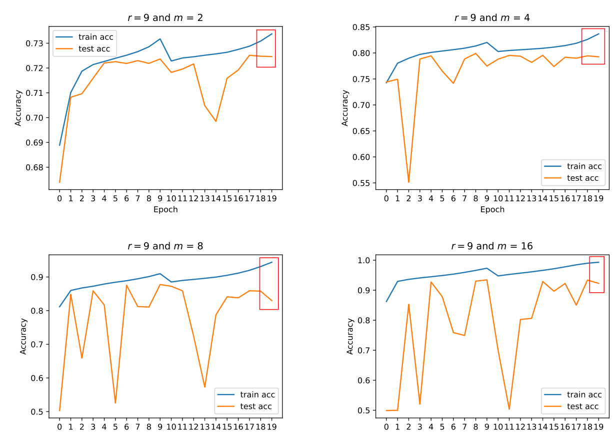


Fig. 15. Using $^{N}/_{m}$ instances for 9-round Simon32/64

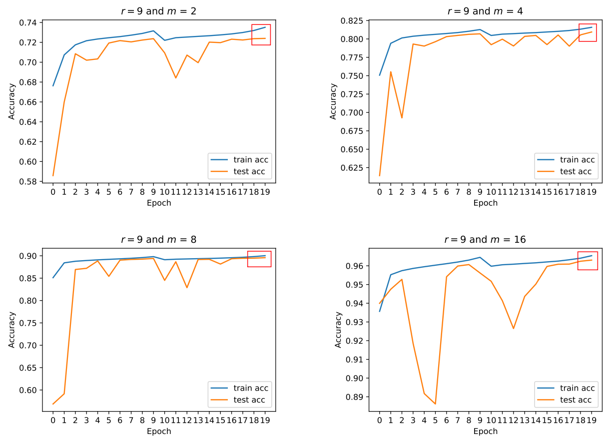


Fig. 16. Using N instances for 9-round SIMON32/64

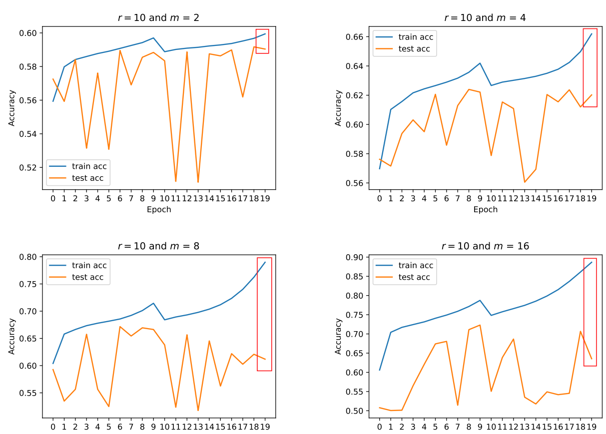


Fig. 17. Using $^{N}/_{m}$ instances for 10-round SIMON32/64

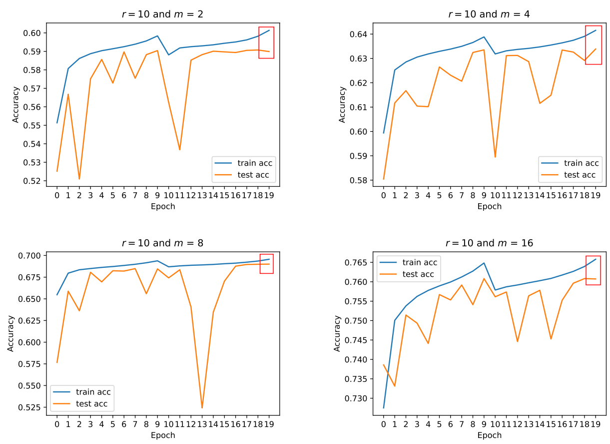


Fig. 18. Using N instances for 10-round Simon32/64

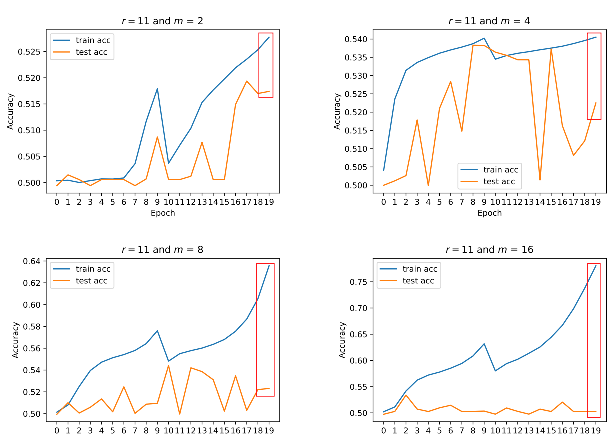


Fig. 19. Using $^{N}/_{m}$ instances for 11-round Simon32/64

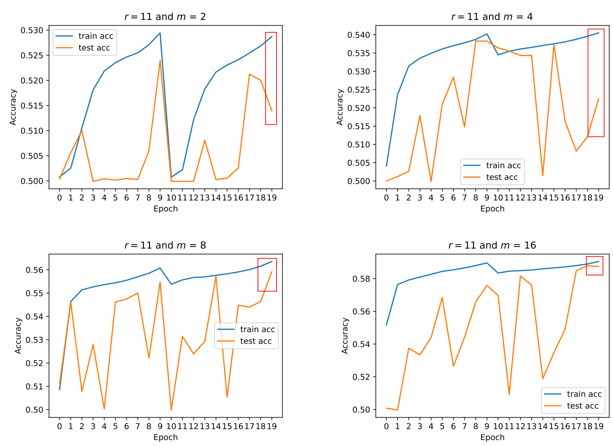


Fig. 20. Using N instances for 11-round SIMON32/64