IACR Transactions on Symmetric Cryptology
ISSN XXXX-XXXX, Vol. 0, No. 0, pp. 1–31. DOI:XXXXXXXX

Improving Differential-Neural Distinguisher with Inception

Liu Zhang[1,2], Zilong Wang[1], Jian Guo[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] School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
guojian@ntu.edu.sg
[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 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. Chen *et al.* improved the accuracy of the differential-neural distinguisher considering derived features from multiple-ciphertext pairs. 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.

We focus on training the distinguisher for more rounds or improving accuracy using deep learning methods. To capture information on multiple dimensions, we use multiple parallel convolutional layers with kernels of different sizes, placed in front of the Residual Network, to train differential-neural distinguisher inspired by the Inception in GoogLeNet. For Speck32/64, we significantly improve the accuracy of the distinguisher in the 6, 7 and 8 rounds and train a 9-round differential-neural distinguisher. For Simon32/64, we obtain a new 12-round differential-neural distinguisher. In addition, we significantly improve the accuracy of the distinguisher in rounds 9, 10, and 11. In addition, we use neutral bits to solve the same distribution of data required to successfully launch a key recovery attack when using multiple-ciphertext pairs as the input of the neural network.

Under the combined effect of multiple improvements, the time complexity of our 12, and 13-round key recovery attacks of Speck32/64 is decreased. Also, the success rate of our 12-round key recovery attack reaches 100% in 100 trials based on differential-neural cryptanalysis. For Simon32/64, we can implement a practical 17-round key recovery attack using the deep learning method for the first time. Also, the time complexity of our 16-round key recovery attack is about 2^4 lower than that of the previous one.

Keywords: Differential-Neural Distinguisher · Inception · Speck · Simon

1 Introduction

In CRYPTO 2019, Gohr [Goh19] proposed the idea of differential-neural cryptanalysis. The differential-neural distinguisher, trained by the neural network, is introduced as the underlying distinguisher. Bayesian search is used to speed 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. If the accuracy of the differential-neural distinguisher is greater than 0.5, it is an effective distinguisher. In EUROCRYPT 2021, Benamira *et al.* [BGPT21] indicated

that Gohr's differential-neural distinguisher builds a good approximation of the differential distribution table of the cipher and learns additional information. However, the current differential-neural distinguisher seems only effective for limited rounds of ciphertext. To increase the number of rounds for key recovery attacks, a short classical high-probability classical differential $\Delta S \to \Delta P$ is prepended before the differential-neural distinguisher.

Gohr [Goh19] showed that the Residual Network [HZRS16] 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.* [BGL+21] 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 [Goh19]. The other is to study the effective differential-neural distinguisher with more rounds. Chen *et al.* [CSYY21] and Benamira [BGPT21] *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.* [BGL+21] used the Dense Network [HLvdMW17] and the Squeeze-and-Excitation Network [HSS18] to train differential-neural distinguisher, and obtained 9, 10, 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. We train a better differential-neural distinguisher. Specifically, we add the Inception composed of the multiple-parallel convolutional layers before the Residual Network. The purpose of the Inception is to capture more information on multiple dimensions in the ciphertext pairs. Some minor modifications have also been made according to the round functions of the cipher. As a result, we improve the accuracy of 6, 7, and 8-round differential-neural distinguisher and train a new 9-round differential-neural distinguisher for Speck32/64. Furthermore, we improve the accuracy of the 9, 10, and 11-round distinguishers and train a new 12-round differential-neural distinguisher for Simon32/64. The result on differential-neural distinguisher of Speck32/64 and Simon32/64 is presented in Tables 1 and 2, where N is the number of training instances and m is the number of ciphertext pairs in every instance. For more detailed comparison results, please see Table 4 and Table 6.

Table 1: Summary accuracy of distinguisher on Speck32/64 using $N = 10^7$ instances

r	[Goh19]	m=2		m=4		m=8		m=16	
		[CSYY21]	Sect. 3	[CSYY21]	Sect. 3	[CSYY21]	Sect. 3	[CSYY21]	Sect. 3
6	0.788	0.8613	0.8773	0.931	0.9497	0.9562	0.9895	0.9802	0.9992
7	0.616	0.6393	0.6649	0.6861	0.7283	0.7074	0.8106	0.6694	0.8963
8	0.514	-	-	-	0.5428	-	0.5562	-	0.5853
9	-	-	-	-	-	-	0.5024	-	0.5050

To successfully implement a key recovery attack, each ciphertext pair in the multiple-ciphertext pairs obtained after the classical differential has the same difference. Inspired by the combined response of differential-neural distinguisher in Gohr's work, we use some neutral bit with probability one to generate multiple-plaintext pairs and then encrypt

Table 2: Summary accuracy of distinguisher on SIMON32/64 using $N = 10^7$ instances

r	[BGL+21]	Sect. 6			
		m=2	m=4	m=8	m=16
9	0.6532	0.7240	0.8095	0.8958	0.9630
10	0.5629	0.5907	0.6339	0.6900	0.7608
11	0.5173	0.5240	0.5387	0.5591	0.5878
12	-	-	-	0.5152	0.5225

multiple-plaintext pairs to get multiple-ciphertext pairs. In addition, in order to evaluate the performance of key recovery attacks more reasonably, we take the success rate into the calculation formula of time complexity.

We successfully implemented several rounds of key recovery attacks using the new differential-neural distinguisher with a classical differential. We reduce the time complexity of the key recovery attack for the 12 and 13 round SPECK32/64. Surprisingly, when a 3-round classical differential is combined with our 7-round differential-neural distinguisher, the 12-round key recovery attack success rate is 100%. For SIMON32/64, we can implement a practical 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 3. The source codes are available in https://github.com/CryptAnalystDesigner/NeuralDistingsuisherWithInception.git.

Organization. Section 2 gives the preliminary on the distinguisher model, key recovery attack process, and generalized neutral bits. Section 3 introduces the training method and result for SPECK32/64. We introduce data generation, experimental environment, and complexity calculation in Section 4. Section 5 exhibits the details of our 12, 13-round key recovery attacks for SPECK32/64. Section 6 introduces the training method and result for SIMON32/64. Section 7 exhibits the details of our 16, 17-round key recovery attacks for SIMON32/64. In Section 8, we conclude the paper.

2 Preliminary

2.1 Brief Description of Speck32/64 and Simon32/64

Let ω be the word size (that is, 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_r the subkey of r rounds. Denote the bitwise XOR by $\oplus$, the addition modulo 2^ω by $\boxplus$, the bitwise AND by $\cdot$, the bitwise right rotation by $\gg$, and the bitwise left rotation by $\ll$.

SPECK32/64 and SIMON32/64 are members in the lightweight block cipher family SPECK and SIMON [BSS+13]. 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 input. The state of the next round (x_{i+1}, y_{i+1}) is computed as follows:

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

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:

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

Table 3: Summary of key recovery attacks on SPECK32/64 and SIMON32/64

Target	r	Dist.	Conf.	Time	Data	Succ. Rate	Key Space	Ref.
SPECK32/64	12	$\mathcal{DD}$	1+7+4	2^{51}	2^{19}	–	2^{64}	[Din14]
		$\mathcal{ND}$	1+2+8+1	$2^{46.57}$	$2^{22.97}$	40%	2^{64}	[Goh19]
		$\mathcal{ND}$	1+3+7+1	$2^{44.82}$	$2^{18.58}$	81%	2^{63}	[BGL+21]
		$\mathcal{ND}$	1+3+7+1	$2^{42.26}$	2^{25}	100%	2^{63}	Exp. 1
		$\mathcal{ND}$	1+3+7+1	$2^{41.26*}$	2^{23}	–	2^{63}	Sect.5.3
	13	$\mathcal{DD}$	1+8+4	2^{57}	2^{25}	–	2^{64}	[Din14]
		$\mathcal{ND}$	1+3+8+1	$2^{53.85}$	2^{29}	61%	2^{63}	[BGL+21]
		$\mathcal{ND}$	1+3+8+1	$2^{52.68}$	2^{31}	49%	2^{63}	Exp. 2
		$\mathcal{ND}$	1+3+8+1	$2^{51.68*}$	2^{29}	–	2^{63}	Sect.5.3
	14	$\mathcal{ND}$	1+9+4	$2^{62.47}$	$2^{30.47}$	–	2^{64}	[SHY16]
SIMON32/64	16	$\mathcal{DD}$	2+12+2	$2^{26.48}$	$2^{29.48}$	62%	2^{64}	[AL13]
		$\mathcal{ND}$	1+3+11+1	$2^{46.98}$	2^{21}	49%	2^{64}	[BGL+21]
		$\mathcal{ND}$	1+3+11+1	$2^{42.79}$	2^{22}	80%	2^{64}	Exp. 3
	17	$\mathcal{ND}$	1+4+11+1	$2^{54.01}$	2^{28}	9%	2^{64}	Exp. 4
	18	$\mathcal{DD}$	1+13+4	$2^{46.00}$	$2^{31.2}$	63%	2^{64}	[ALLW14]
	19	$\mathcal{DD}$	2+13+4	$2^{34.00}$	$2^{31.5}$	-	2^{64}	[BRV14]
	21	$\mathcal{DD}$	4+13+4	$2^{55.25}$	$2^{31.0}$	-	2^{64}	[WWJZ18]

1. All time complexities of key recovery attack using $\mathcal{ND}$ are recalculated using the new time complexity calculation formula proposed in Sect. 4.2.

2. $\mathcal{DD}$: differential distinguisher; $\mathcal{ND}$: differential-neural distinguisher; –: Not available;

3. *: Assuming two classical differentials share the same output differences, the time complexity is halved, and the data complexity is reduced by 4 times (refer to Sect.5.3 for detail).

2.2 The Model of Differential-Neural Distinguisher for Speck32/64

The model of differential-neural distinguisher in [Goh19, BGPT21, CSYY21] 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 an instance. Note that $m = 1$ in [Goh19], $m \in \{1, 5, 10, 50, 100\}$ in [BGPT21], and $m \in \{2, 4, 8, 16\}$ in [CSYY21]. 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. Suppose that the neural network can obtain a stable accuracy higher than 0.5 on a test set. In that case, 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 simple bit-sliced functions such as bitwise addition easier. Module 2 is the Residual Network. $Conv$ stands for one-dimensional convolution $Conv1D$ with N_f filters, and k_s is the size of the convolution kernel. The number of module 2 is determined by experiment. The prediction head consists of modules 3, 4, and the output layer. FC is a fully connected layer that has d_1 or d_2 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 network has a deep depth, it can easily cause gradients. 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.

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

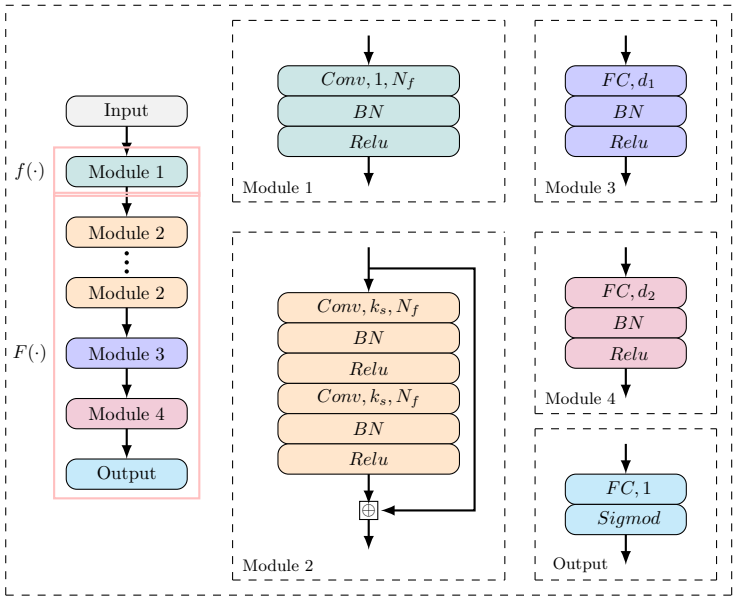


Figure 1: The network architecture for training differential-neural distinguisher

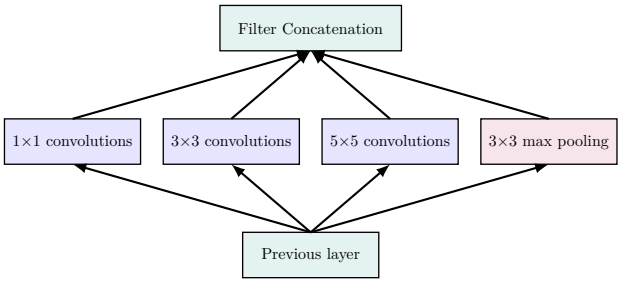


Figure 2: Inception Module

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 [Goh19] 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.

The overall processing of a key recovery attack based on 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 $(\Delta S \rightarrow \Delta P)$ 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

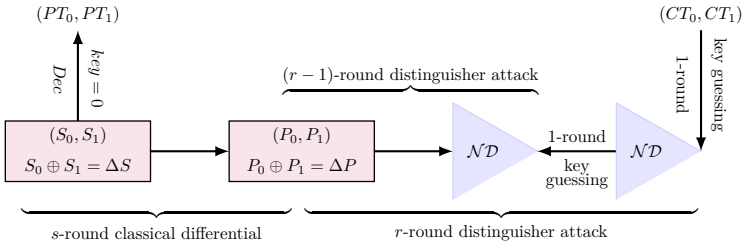


Figure 3: $(1 + s + r + 1)$-round key recovery attack of differential-neural cryptanalysis

to the probability of difference propagation, where c is a small constant. Neutral bits of the s-round classical differential are 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 [Goh19].

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 large amounts of instances with the same distribution, which can be satisfied by neutral bits [Goh19]. The primary notion of neutral bits can be interpreted as follows.

Definition 1 (**Neutral bits of a differential, NB**[BC04])**.** 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. In order to enhance the distinguish ability of the low-accuracy differential-neural distinguisher, the number of instances with the same distribution should be sufficiently large. However, neutral bits of the nontrivial classical differential are scarce. Therefore, probabilistic neutral bits (PNB) are exploited in [Goh19]. Some probabilistic neutral bits, simultaneous-neutral bit-sets (SNBS), and conditional (simultaneous-) neutral bit(-set)s (CSNBS) were found in [BGL+21] (refer to Appendix A).

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

3.1 Network Architecture

The general architecture of our neural network to train the differential-neural distinguisher is shown in Fig. 4. Our neural network consists of four parts: an input layer consisting of multiple-ciphertext pairs, an initial convolutional layer consisting of four parallel convolutional layers, a residual tower with multiple two-layer convolutional neural networks, and a prediction head consisting of multiple fully connected layers.

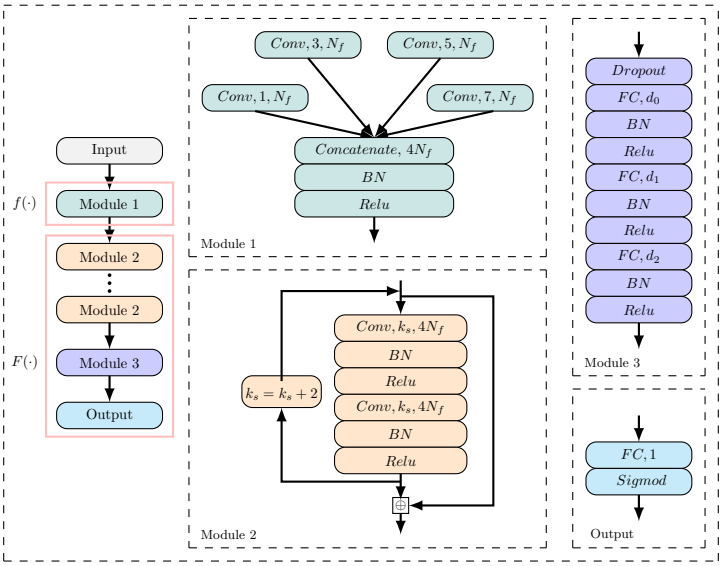


Figure 4: The network architecture of our distinguisher for Speck32/64

Input Representation. If the output of the r-th round $(C, C') = (x_r||y_r, x'_r||y'_r)$ is known, one can directly compute (y_{r-1}, y'_{r-1}) without knowing the $(r-1)$-th subkey according to the round function of Speck. Thus, the neural network accepts data of the form $(x_r, x'_r, y_r, y'_r, y_{r-1}, y'_{r-1})$. The input layer has m ciphertext pairs consisting of $3L$ units likewise arranged in a $[m, \omega, \frac{3L}{\omega}]$ array, where $L = 32, \omega = 16$ for Speck32/64.

Initial Convolution (Module 1). The input layer is connected to the initial convolutional layer, which comprises four convolutional layers with N_f channels of different kernel sizes. The four convolution layers are concatenated at the channel dimension, similar to the Inception [SLJ+15] in GoogLeNet. 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 $[m, \omega, 4N_f]$ matrix is passed 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, 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 512, 64, and 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. First, we modify the format of the input data of the model. Given the ciphertext of the last round, the right half of the ciphertext of the penultimate round can be calculated according to the round function of SPECK32/64 without knowing the (r-1)-th subkey. Second, considering the circular shift operation in the round function of SPECK and some relationship in the adjacent bits generated by the modular addition operation, we add the convolution operation of widths 3, 5, and 7 to capture the multiple dimensional features inspired by the Inception [SLJ+15] in GoogLeNet. Third, to increase the convolution's receptive field, the convolution kernel's size increases by 2 with the increase of the depth of the Residual Network. In addition, our network architecture has a solid fitting ability. We add a dropout layer before the fully connected layer to improve the network generalization ability.

3.2 The Training of Differential-Neural Distinguisher

The accuracy is the most critical indicator reflecting differential-neural distinguisher performance. The following training was carried out to verify the performance of our differential-neural distinguisher.

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 conduct two sets of experiments with different numbers of datasets.* In [Goh19], the training set and the test set include N and M instances, which consist of a ciphertext pair, that is, total N and M ciphertext pairs, respectively. In [CSYY21], 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 the same amount of data as [CSYY21] in the first set of experiments. However, using N/m and M/m instances as a training and test sets 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 in the second set of experiments, 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 [KB15]. 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)). 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 distinguish accuracy of 6, 7, 8-, and 9-round differential-neural distinguisher compared to [Goh19, CSYY21, BGPT21] in Table 4 . Also, we list the accuracy (Acc), the true positive rate (TPR) and the true negative rate (TNR) tested on newly generated data in Table 5. The 6, 7-round distinguishers were trained using the basic training method, while the 8, 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 4, the accuracy of our differential-neural distinguisher was significantly improved both the "N/m" column and the "N" column compared to [Goh19, CSYY21, BGPT21]. There are some differences in accuracy under the two experiments caused by overfitting of the model; see *Remark 2* for details.

Table 4: Summary accuracy of distinguisher on SPECK32/64 using different number of instances

r	[Goh19]	m=2			m=4			m=5		
		[CSYY21]	N/m	N	[CSYY21]	N/m	N	[BGPT21]	N/m	N
6	0.788	0.8613	0.8771	0.8773	0.931	0.9468	0.9497	0.9541	0.9623	0.965
7	0.616	0.6393	0.6663	0.6649	0.6861	0.7194	0.7283	0.735	0.7436	0.7491
8	0.514	-	-	-	-	0.5329	0.5428	-	-	-
r	[Goh19]	m=8			m=10			m=16		
		[CSYY21]	N/m	N	[BGPT21]	N/m	N	[CSYY21]	N/m	N
6	0.788	0.9562	0.9868	0.9895	0.99	0.9915	0.9939	0.9802	0.9969	0.9992
7	0.616	0.7074	0.7991	0.8106	0.808	0.8243	0.8341	0.6694	0.8759	0.8963
8	0.514	-	0.5347	0.5590	-		-	-	0.5566	0.5854
9	-	-	-	0.5024	-		-	-	-	0.5050

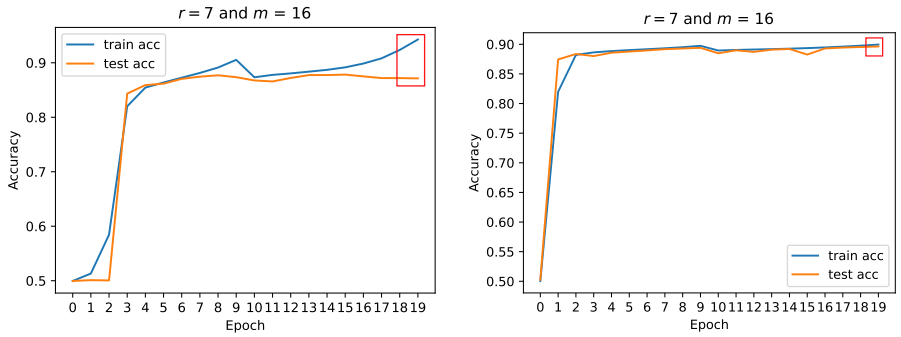
In the Table 4, except for Benamira's result [BGPT21], other results are directly trained by the neural network. In [BGPT21], the value of m is $\{1, 5, 10, 50, 100\}$. But in order to facilitate the key recovery attack, we set the value of m at the power of 2, that is, $\{2, 4, 8, 16\}$. When $m = 1$, the case of Benamira and Gohr is the same. In [BGPT21], 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

Table 5: Acc, TPR, TNR of distinguisher on SPECK32/64 using N instances

m	r	Acc	TPR	TNR	m	r	Acc	TPR	TNR
2	6	0.8768	0.8448	0.9087	4	6	0.9495	0.9429	0.9559
	7	0.6635	0.6240	0.7029		7	0.7291	0.7048	0.7532
	8	-	-	-		8	0.5428	0.5318	0.5537
8	6	0.9897	0.9862	0.9931	16	6	0.9992	0.9988	0.9996
	7	0.8103	0.8024	0.8184		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

distinguisher score for each element of the multiple ciphertext pairs and then to take the median of the results. 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 actual training of the differential-neural distinguisher using the second method will be less accurate. Benamira use two methods to get the accuracy of 6-round distinguisher. When $m = 5$, the accuracy of the two methods is 0.9541 and 0.9327, respectively, in [BGPT21]. When $m = 10$, the accuracy of the two methods is 0.99 and 0.977, respectively in [BGPT21]. We can see that the distinguisher accuracy 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 [BGPT21]. This means that if the distinguisher is trained directly using a neural network, the accuracy obtained should be even lower.

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 a training and test sets (see *Remark 1*). From Fig. 5a, we can see that the difference between training and test accuracy is relatively significant. Therefore, using N/m and M/m instances as training and test sets 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 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

Figure 5: Overfitting with different number of instances.

Wrong Key Response Profile. In [Goh19], an improved key recovery attack was presented using Upper Confidence Bound and Wrong key Response Profile. More specifically, 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 key and the real key. The wrong key response profile, which can be precomputed, 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. If the Hamming distance between the correct and wrong keys is small, the distinguisher will score high; otherwise, the score will be low. From Fig. 6 and 7, the score of our differential-neural distinguisher is higher than that of Gohr's distinguisher when the Hamming distance of the key difference is small. When the Hamming distance is large, we can obtain a lower score. In other words, our distinguisher makes the score of the real key and the wrong key even more significant, i.e., our differential-neural distinguisher is better. Moreover, it can be observed that the score of distinguisher are higher when the difference between the guessed key and the real key belongs to {16384, 32768, 49152}. This means that when the 14th, 15th bits of the subkey are guessed incorrectly, it has little effect on the score of distinguisher. Therefore, it is possible to reduce the key guessing space by not guessing these two bits, and this feature is also used in [Goh19] to accelerate the key recovery attack.

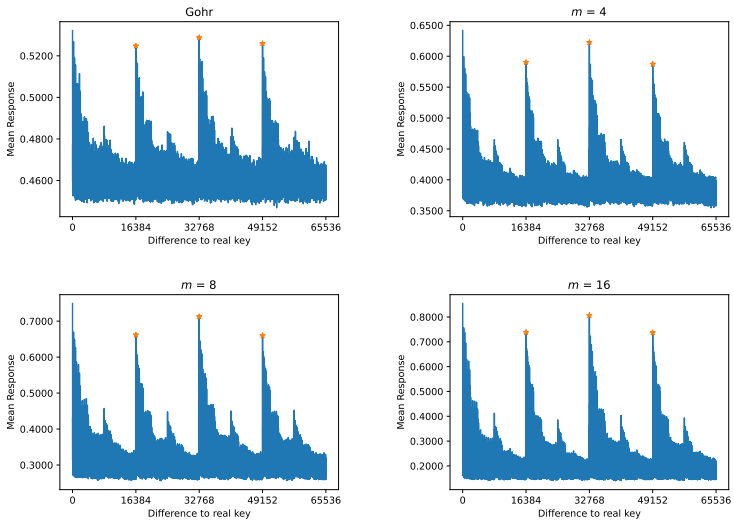


Figure 6: Wrong key response profile for 7-round Speck32/64 using N instances

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 input to the neural network. The data in multiple-ciphertext pair has the same distribution; that is, if it is a positive example, the data in multiple-ciphertext pair is encrypted by multiple-plaintext pair satisfying a fixed difference; if it is a negative example, the data in multiple-ciphertext pair is encrypted by a randomly generated multiple-

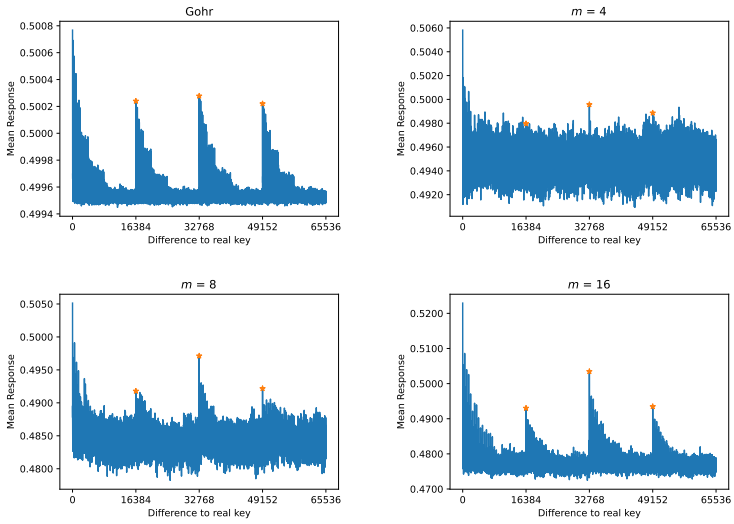


Figure 7: Wrong key response profile for 8-round SPECK32/64 using N instances

plaintext pair. After obtaining a differential-neural distinguisher, we can launch a key recovery attack. We add a classical differential before the differential-neural distinguisher to increase the number of rounds for the key recovery attack. However, the classical differential is probabilistic. Even if multiple-plaintext pairs have the same difference Δ_S, after propagation through the classical differential, the difference between multiple-ciphertext pairs may not be the same. Therefore, the initial input (multiple-plaintext pairs) cannot be generated randomly during a key recovery attack. Neutral bits can solve the problem of obtaining ciphertext that satisfies the same distribution, ensuring that we can launch key recovery attacks. We randomly generate a plaintext pair that satisfies the initial difference Δ_S and use $\log_2 m$ neutral bits to obtain m plaintext pairs. The m ciphertext pairs obtained by propagation of the m plaintext pairs through the classical differential have the same difference with substantial probability. Moreover, the optimal choice is to use neutral bits with probability one. After r rounds of encryption, the obtained multiple-ciphertext pairs have the same distribution.

4.2 Experimental Environment and Complexity Calculation

The core of the key recovery attack was examined using a server with 5 GeForce GTX 2080-Ti GPUs. When a fast graphics card is used, the performance of our implementation is not limited by the speed of neural network evaluation but by the total number of iterations on the ciphertext structures. We count a key guess as successful if the sum of the Hamming weights of the differences between the returned last two subkeys and the real two subkeys is at most two. 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 used may need to satisfy some conditions involving the key. The data complexity of the experiment is calculated using the formula $n_{kg} \times n_b \times n_{ct} \times m \times 2$, where n_{kg} indicates the number of times to guess the key involved in the condition. 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. Furthermore, a large amount of data is generated from neutral bits. If we only calculate the amount of data obtained from the outside world, n_b and m need not be included.

In [Goh19], a highly optimized, fully SIMD-parallelized implementation of Speck32/64 could perform a brute force key search at a speed of about 2^{28} keys per second per core. A real key guess takes an average of 500 seconds, and the single success rate of a key recovery attack of 11-round Speck32/64 is 52.1%. About 1000 seconds is needed to execute the key recovery attack until a real key is found. This yields an estimated computational attack complexity of $2^{28} \times 1000 \approx 2^{38}$ Speck32/64 encryptions until a solution is found.

Experimental Time Complexity. The formula in [Goh19] to calculate the time complexity does not take into account the success rate of the key recovery attack. To better evaluate the effect of key recovery attacks, we give a new calculation method and update the previous result to take into account the success rate. To reduce the experimental error, we conduct multiple key recovery attacks, take the average running time rt as the running time of an experiment, and divide the number of successful experiments by the total experimental number as the success rate sr of a single key recovery attack. We calculate how many experiments need to be performed until a reak key is found. When the overall success rate is 99%, we consider the experiment to be successful, and the number of experiments ne is: $1-(1-sr)^{ne} = 0.99$, that is, $\log_{1-sr} 0.01$. For example, this yields an estimated computational attack complexity of $2^{28} \times 500 \times \log_{1-0.52} 0.01 \approx 2^{39.6}$ for 11-round Speck32/64 in [Goh19]. Also, the average running time of Gohr's experiment is 225s in 300 trials in our equipment. To facilitate comparison, the time complexity of the experiment in [Goh19] is taken as the benchmark for conversion. The new formula for calculating the time complexity in our experiments: $n_{kg} \times 2^{28} \times rt \times {}^{500}/_{225} \times \log_{1-sr} 0.01$.

5 Key Recovery Attack on Round-Reduced Speck32/64

This section shows that our differential-neural distinguisher can improve the performance of key recovery attacks. Our attack follows the same framework in [Goh19, BGL+21], only replacing the differential-neural distinguisher trained using N instances. For a more detailed procedure, refer to Appendix B. We significantly reduced the time complexity of 12, and 13-round key recovery attacks for Speck32/64 by employing an improved differential-neural distinguisher.

5.1 Key Recovery Attack on 12-round Speck32/64

To verify the performance of our 7-round differential-neural distinguisher, the following experiments were carried out in this subsection.

Experiment 1: The components of the key recovery attack $\mathcal{A}^{\text{Speck12}R}$ of 12-round Speck32/64 are shown below.

1. 3-round classical differential (0x8020, 0x4101)→ (0x0040, 0x0000);

2. generalized neutral bits of generating multiple-ciphertext pairs: $\{[22],[20],[13]\}$; generalized neutral bits of the combined response of differential-neural distinguisher: $\{[12,19],[14,21],[6,29],[30],[0,8,31],[5,28]\}$ (see Table 11).

3. 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 $\mathcal{ND}^{\text{SPECK}_{7R}} \cdot \delta$.

4. 6-round differential-neural distinguisher $\mathcal{ND}^{\text{SPECK}_{6R}}$ under difference (0x0040, 0x0000) and its wrong key response profiles $\mathcal{ND}^{\text{SPECK}_{6R}} \cdot \mu$ and $\mathcal{ND}^{\text{SPECK}_{6R}} \cdot \delta$.

At 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 the 3-round differential (see Table 10) and one key bit $k_0[12] \oplus k_0[5]$ to employ one conditional neutral bit (see Table 11). Thus, n_{kg} is 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 12-round key recovery attack $\mathcal{A}^{\text{SPECK}12R}$ are listed below.

$n_{kg} = 2^4$	$m = 8$	$n_b = 2^6$	$n_{cts} = 2^{11}$
$n_{it} = 2^{13}$	$c_1 = 7, c_2 = 10$	$n_{byit1} = n_{byit2} = 5$	$n_{cand1} = n_{cand2} = 32$

The theoretical data complexity is $n_{kg} \times n_b \times n_{ct} \times m \times 2 = 2^4 \times 2^6 \times 2^{11} \times 8 \times 2 = 2^{25}$ plaintexts. The experimental data complexity is $n_{kg} \times n_b \times \min(n_{ct}, \text{num\_used}) \times m \times 2 = 2^4 \times 2^6 \times 2^{11} \times 8 \times 2 = 2^{23.16}$ plaintexts, where num_used is the actual number of iterations in the experiment. The core of the attack was examined in 100 trials. There are 100 successful trials and we count the success rate as 100%. The average run time of every trail in our server is 553.03s. The time complexity is $n_{kg} \times 2^{28} \times rt \times {}^{500}/_{225} = 2^4 \times 2^{28} \times 553.03 \times {}^{500}/_{225} = 2^{42.26}$.

5.2 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 2: The components of key recovery attack $\mathcal{A}^{\text{SPECK}13R}$ of 13-round SPECK32/64 are shown as follows.

1. 3-round classical differential (0x8020, 0x4101)→ (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 11).

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 $\mathcal{ND}^{\text{SPECK}_{7R}} \cdot \delta$.

At 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 10) 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 11). 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 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 $n_{kg} \times n_b \times n_{ct} \times m \times 2 = 2^7 \times 2^9 \times 2^{11} \times 8 \times 2 = 2^{31}$ plaintexts. The core of the attack was examined in 35 trials. There are 17 successful trials, and we count the success rate as 0.4857. The average run time of every trail in our server is 13690s. Thus, the time complexity is $n_{kg} \times 2^{28} \times rt \times {}^{500}/_{225} \times \log_{1-sr} 0.01 = 2^7 \times 2^{28} \times 13690 \times {}^{500}/_{225} \times \log_{1-0.4857} 0.01 = 2^{52.68}$.

5.3 Using Multiple classical differentials

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 path can be prepended to a differential-neural distinguisher as long as 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. Bao launched the $R_{1+3+7+1}$-round key recovery attack using four sub-optimal classical differentials and the $R_{1+3+8+1}$-round key recovery attack using two sub-optimal classical differentials (refer to Table 10). However, in our experiment, we only use one classical differential when we launch the key recovery attack by the prepended 3-round classical differential on top of the differential-neural distinguisher. When using two sub-optimal differential paths to generate multiple-ciphertext pairs, we only need to use two conditions conform two 3-round sub-optimal classical differentials (0x8020, 0x4101)→ (0x0040, 0x0000) and (0x8060, 0x4101)→ (0x0040, 0x0000). So, the time complexity is halved and the data complexity is reduced 4 times for the 12, 13-round key recovery attacks in the ideal situation.

6 Differential-Neural Distinguishers on Round-Reduced Simon32/64

In CRYPTO'19, Gohr used the Residual Network to train the differential-neural distinguisher for Speck32/64. Bao *et al.* modified the architecture of Gohr's neural network using the Dense Network and Squeeze-and-Excitation Network, training 9, 10, and 11-round differential-neural distinguishers for Simon32/64 [BGL+21]. This section presents our differential-neural distinguisher on Simon32/64. Compared to Bao's results in [BGL+21], we significantly improve the accuracy of the 9, 10, and 11-round and obtain a new 12-round differential-neural distinguisher for Simon32/64.

6.1 Network Architecture

The network architecture for training differential-neural distinguisher for Simon32/64 is similar to Speck32/64 in general, with only minor modifications based on the round function of Simon32/64. The overall architecture is shown in Fig 8.

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 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 $[m, \omega, 3N_f]$

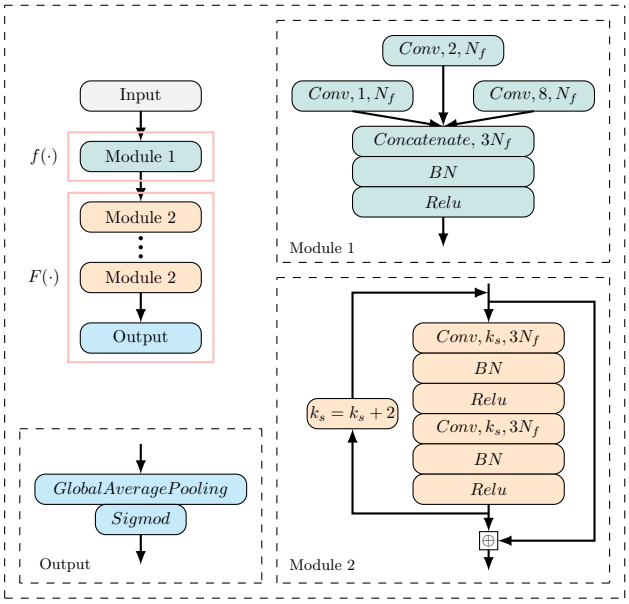


Figure 8: The network architecture of our distinguisher for SIMON32/64

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 layer.

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. For training parameters, refer to the basic training scheme of SPECK32/64 in Sect. 3. 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)). 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 6. In addition, we list Acc, TPR and TNR tested on newly generated data in Table 7. 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 Bao, the accuracy of our differential-neural distinguisher was improved significantly when m took different values. In addition, we train the differential-neural distinguisher for one more round.

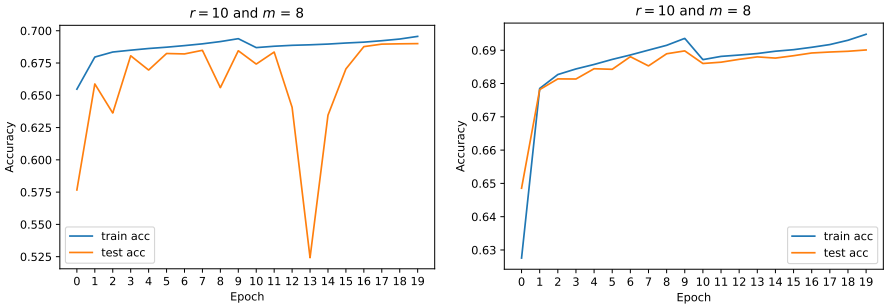
Table 6: Summary accuracy of the distinguisher on SIMON32/64 using different number of instances

r	[BGL+21]	m=2		m=4		m=8		m=16	
		N/m	N	N/m	N	N/m	N	N/m	N
9	0.6532	0.7251	0.7240	0.7991	0.8095	0.8774	0.8958	0.9344	0.9630
10	0.5629	0.5917	0.5907	0.6239	0.6339	0.6716	0.6900	0.7230	0.7608
11	0.5173	0.5193	0.5240	0.5343	0.5387	0.5441	0.5591	0.5339	0.5878
12	-	-	-	-	-	-	0.5152	-	0.5225

Table 7: 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

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 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, we can see that using different prediction heads does not significantly impact the accuracy of the differential-neural distinguisher. However, it will lead to different fluctuations in accuracy.



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

Figure 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. It is easy to judge how far the guessed key deviates from the real key. 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 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. This means that when the 13th, 14th, and 15th bits of the subkey are guessed incorrectly, it has little effect on the score of distinguisher. Therefore, it is possible to reduce the key guessing space by not guessing these three bits and accelerate the key recovery attack.

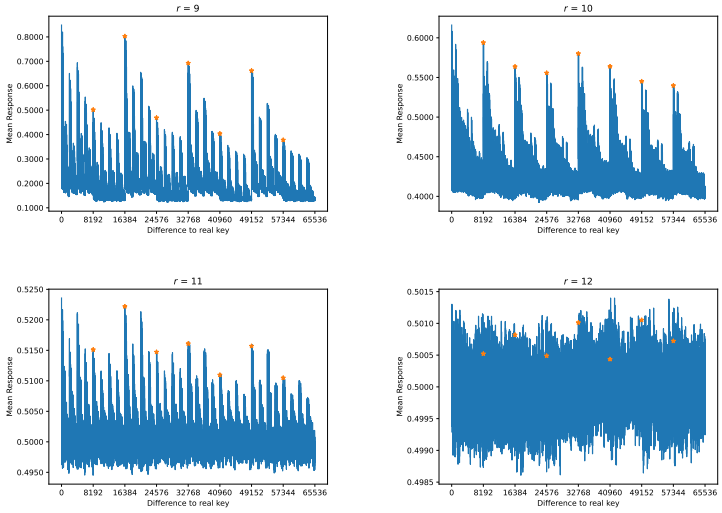


Figure 10: Wrong key response profile for SIMON32/64 using N instances where $m = 8$

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, due to the lack of a sufficient number of generalized neutral bits, we were unable to successfully perform the 18-round key recovery attack for SIMON32/64.

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 $\mathcal{A}^{\text{SIMON16}R}$ 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 distinguis-her:$\{[6],[8],[9],[10],[18],[22],[0,24],[12,26]\}$ (refer to Table 12).
3. 11-round differential-neural distinguisher $\mathcal{ND}^{\text{SIMON}_{11}R}$ under difference (0x0000, 0x0040) and wrong key response profiles $\mathcal{ND}^{\text{SIMON}_{11}R} \cdot \mu$ and $\mathcal{ND}^{\text{SIMON}_{11}R} \cdot \delta$.
4. 10-round differential-neural distinguisher $\mathcal{ND}^{\text{SIMON}_{10}R}$ under difference (0x0000, 0x0040) and wrong key response profiles $\mathcal{ND}^{\text{SIMON}_{10}R} \cdot \mu$ and $\mathcal{ND}^{\text{SIMON}_{10}R} \cdot \delta$.

At 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 $\mathcal{A}^{\text{SIMON16}R}$ 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 $n_{kg} \times m \times n_{cts} \times n_b \times 2 = 2^2 \times 8 \times 2^8 \times 2^8 \times 2 = 2^{22}$ plaintexts. In total, 100 trials are running and there are 80 successful trials. Thus, the success rate is 0.8. The average run time of the experiment is 1115.42s in 100 trials. The time complexity is $n_{kg} \times 2^{28} \times rt \times {}^{500}/_{225} \times \log_{1-sr} 0.01 = 2^2 \times 2^{28} \times 1115.42 \times {}^{500}/_{225} \times \log_{1-0.8} 0.01 = 2^{42.79}$.

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 $\mathcal{A}^{\text{SIMON17}R}$ 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 13); 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 14).
3. 11-round differential-neural distinguisher $\mathcal{ND}^{\text{SIMON}_{11}R}$ under difference (0x0000, 0x0040) and wrong key response profiles $\mathcal{ND}^{\text{SIMON}_{11}R} \cdot \mu$ and $\mathcal{ND}^{\text{SIMON}_{11}R} \cdot \delta$.

4. 10-round differential-neural distinguisher $\mathcal{ND}^{\text{SIMON}_{10R}}$ under difference (0x0000, 0x0040) and wrong key response profiles $\mathcal{ND}^{\text{SIMON}_{10R}} \cdot \mu$ and $\mathcal{ND}^{\text{SIMON}_{10R}} \cdot \delta$.

At 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 14). 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 $n_{kg} \times m \times n_{cts} \times n_b \times 2 = 2^8 \times 8 \times 2^6 \times 2^{11} \times 2 = 2^{29}$ plaintexts. In total, 100 trials are running and 9 successful trials. Thus, the success rate is 0.1. The average running time of the experiment is 2435.63s in 100 trials. The time complexity is $n_{kg} \times 2^{28} \times rt \times {}^{500}/_{225} \times \log_{1-sr} 0.01 = 2^8 \times 2^{28} \times 2435.63 \times {}^{500}/_{225} \times \log_{1-0.09} 0.01 = 2^{54.01}$.

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

[AL13] Hoda AlKhzaimi and Martin M. Lauridsen. Cryptanalysis of the SIMON family of block ciphers. *IACR Cryptol. ePrint Arch.*, page 543, 2013.

[ALLW14] Farzaneh Abed, Eik List, Stefan Lucks, and Jakob Wenzel. Differential cryptanalysis of round-reduced simon and speck. In *FSE*, volume 8540 of *Lecture Notes in Computer Science*, pages 525–545. Springer, 2014.

[BC04] Eli Biham and Rafi Chen. Near-collisions of SHA-0. In *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 290–305. Springer, 2004.

[BGL+21] Zhenzhen Bao, Jian Guo, Meicheng Liu, Li Ma, and Yi Tu. Conditional differential-neural cryptanalysis. *IACR Cryptol. ePrint Arch.*, page 719, 2021.

[BGPT21] Adrien Benamira, David Gérault, Thomas Peyrin, and Quan Quan Tan. A deeper look at machine learning-based cryptanalysis. In *EUROCRYPT (1)*, volume 12696 of *Lecture Notes in Computer Science*, pages 805–835. Springer, 2021.

[BRV14] Alex Biryukov, Arnab Roy, and Vesselin Velichkov. Differential analysis of block ciphers SIMON and SPECK. In *FSE*, volume 8540 of *Lecture Notes in Computer Science*, pages 546–570. Springer, 2014.

[BSS+13] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK families of lightweight block ciphers. *IACR Cryptol. ePrint Arch.*, page 404, 2013.

[CSYY21] Yi Chen, Yantian Shen, Hongbo Yu, and Sitong Yuan. A new neural distinguisher considering features derived from multiple ciphertext pairs, 2021.

[Din14] Itai Dinur. Improved differential cryptanalysis of round-reduced speck. In *Selected Areas in Cryptography*, volume 8781 of *Lecture Notes in Computer Science*, pages 147–164. Springer, 2014.

[Goh19] Aron Gohr. Improving attacks on round-reduced speck32/64 using deep learning. In *CRYPTO (2)*, volume 11693 of *Lecture Notes in Computer Science*, pages 150–179. Springer, 2019.

[HLvdMW17] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR*, pages 2261–2269. IEEE Computer Society, 2017.

[HSS18] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, pages 7132–7141. Computer Vision Foundation / IEEE Computer Society, 2018.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778. IEEE Computer Society, 2016.

[KB15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

[SHY16] Ling Song, Zhangjie Huang, and Qianqian Yang. Automatic differential analysis of ARX block ciphers with application to SPECK and LEA. In *ACISP (2)*, volume 9723 of *Lecture Notes in Computer Science*, pages 379–394. Springer, 2016.

[SLJ+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9. IEEE Computer Society, 2015.

[WWJZ18] Ning Wang, Xiaoyun Wang, Keting Jia, and Jingyuan Zhao. 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

Neutral bit-sets (NB) Used in [Goh19] 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 8: (Probabilistic) single-bit neutral bit for 2-round Classical Differential (0x0211, 0x0a04)→ (0x0040, 0x0000) of Speck32/64 [Goh19]

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 [BGL+21] for 2-round classical differential: for the prepended 2-round differential on top of differential-neural distinguisher, Bao *et al.* [BGL+21] 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 9.

Table 9: (Probabilistic) SNBS for 2-round Classical Differential (0x0211, 0x0a04)→ (0x0040, 0x0000) of Speck32/64 [BGL+21]

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 [BGL+21] 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 10. Concretely, for the 3-round four sub-optimal differential, bit and bit-sets that are (probabilistically) conditional simul-taneous-neutral are summarized in Table 11.

Table 10: Three sufficient conditions conform the 3-round sub-optimal differential [BGL+21]

(0x8020, 0x4101) ↓ (0x0040, 0x0000)	(0x8060, 0x4101) ↓ (0x0040, 0x0000)	(0x8021, 0x4101) ↓ (0x0040, 0x0000)	(0x8061, 0x4101) ↓ (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 11: (Probabilistic) (simultaneous-)neutral bit-sets for 3-round differential (0x8020, 0x4101) → (0x0040, 0x0000), (0x8060, 0x4101) → (0x0040, 0x0000), (0x8021, 0x4101) → (0x0040, 0x0000), (0x8061, 0x4101) → (0x0040, 0x0000) of SPECK32/64 [BGL+21]

	(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	$y[1] = 0$
[6,11,12,18]	0.445	0.903	0.456	0.906	0.333	0.701	0.382	0.726	$x[2] \oplus y[11] = 0$
[4,27,29]	0.672	0.916	0.648	0.905	0.535	0.736	0.536	0.718	$x[11] \oplus y[4] = 1$

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

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

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

Table 12: NB and SNBS for 3-round Classical Differential (0x0440, 0x1000)→ (0x0000, 0x0040) of SIMON32/64 [BGL+21]

[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 13: NB and SNBS for 4-round Classical Differential (0x1000, 0x4440)→ (0x0040, 0x0000) of SIMON32/64 [BGL+21]

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

Table 14: CSNBS for 4-round Classical Differential (0x1000, 0x4440)→ (0x0040, 0x0000) of SIMON32/64 [BGL+21]

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 [Goh19].
      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 [Goh19] with $\mathcal{C}$, the r-round differential-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 [Goh19] 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.

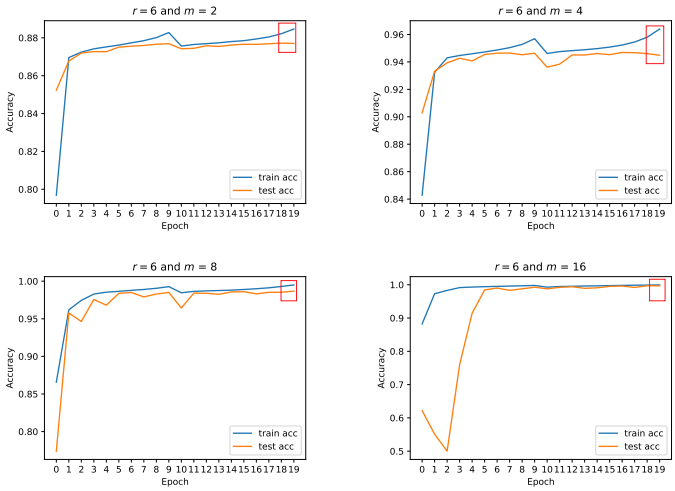


Figure 11: Using N/m instances for 6-round SPECK32/64

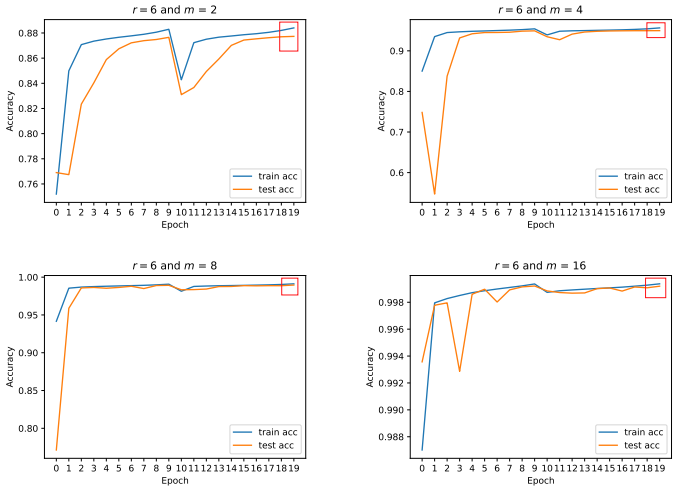


Figure 12: Using N instances for 6-round SPECK32/64

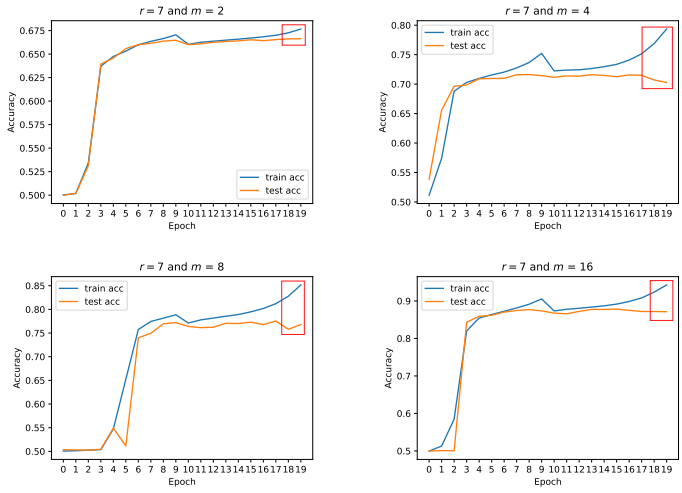


Figure 13: Using N/m instances for 7-round SPECK32/64

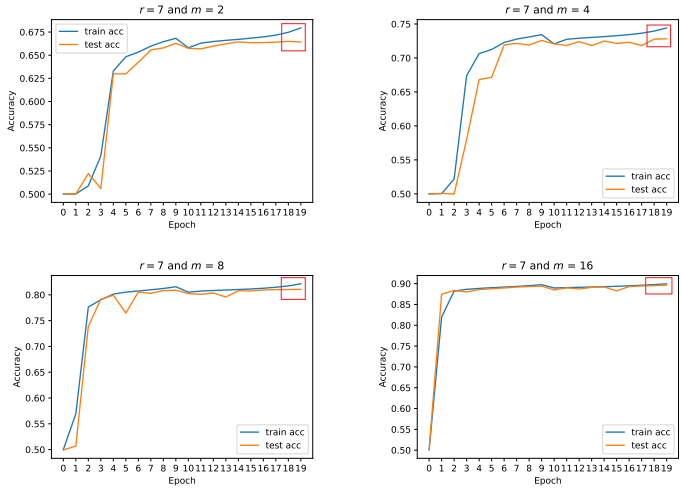


Figure 14: Using N instances for 7-round SPECK32/64

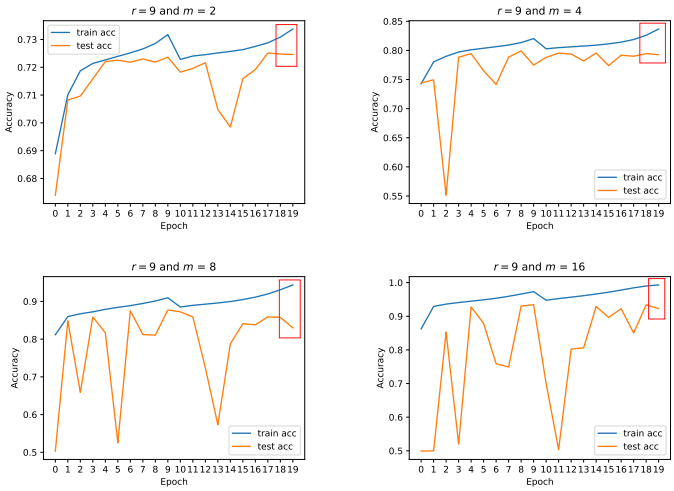


Figure 15: Using $^{N}/_{m}$ instances for 9-round SIMON32/64

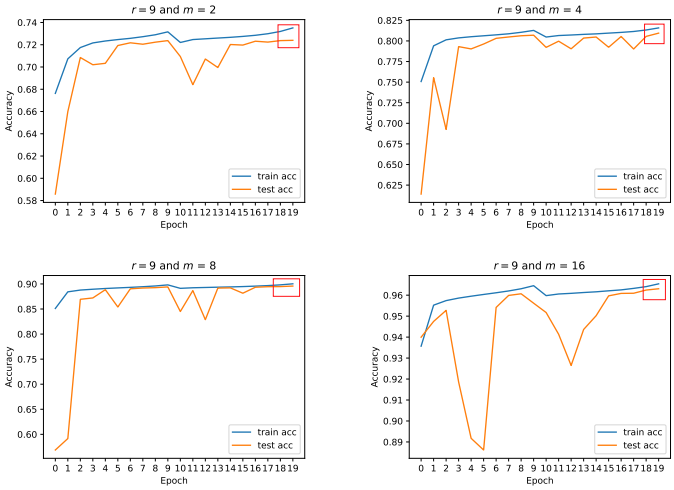


Figure 16: Using N instances for 9-round SIMON32/64

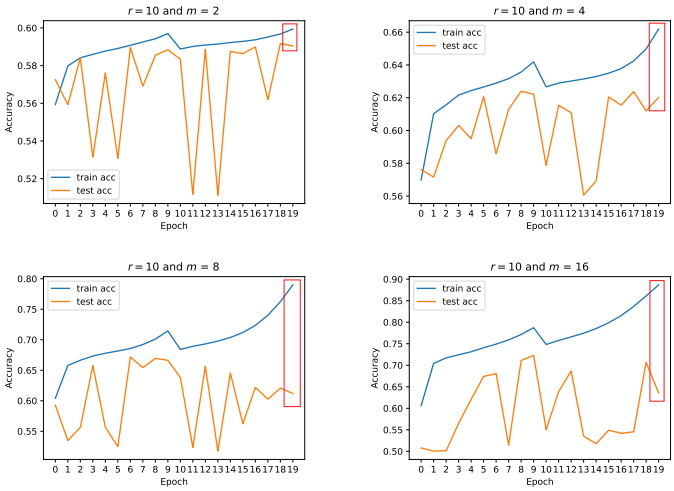


Figure 17: Using N/m instances for 10-round SIMON32/64

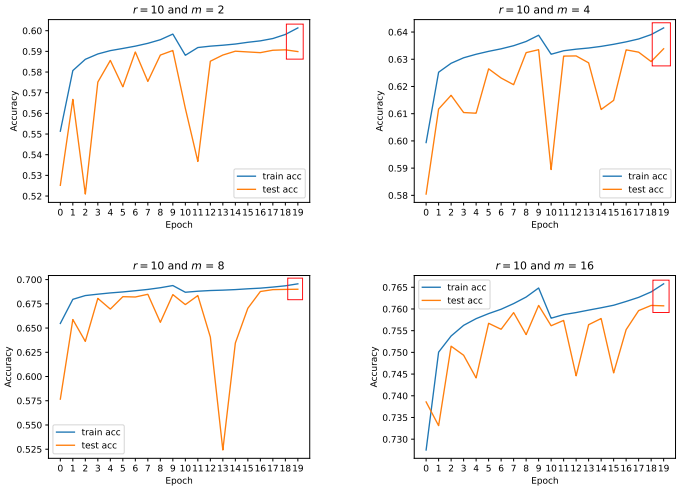


Figure 18: Using N instances for 10-round SIMON32/64

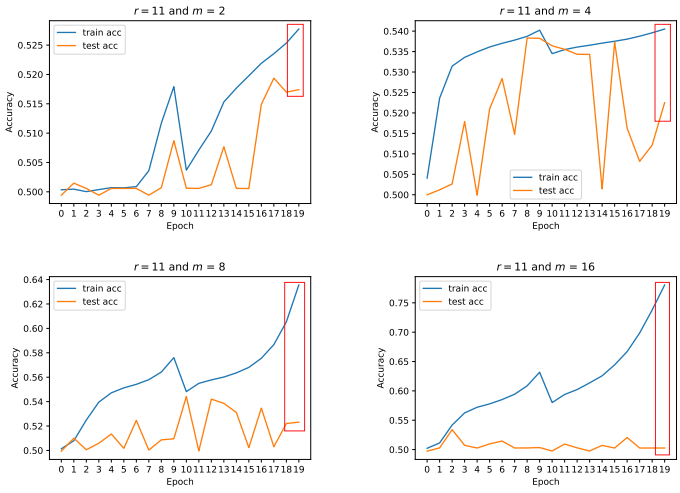


Figure 19: Using N/m instances for 11-round SIMON32/64

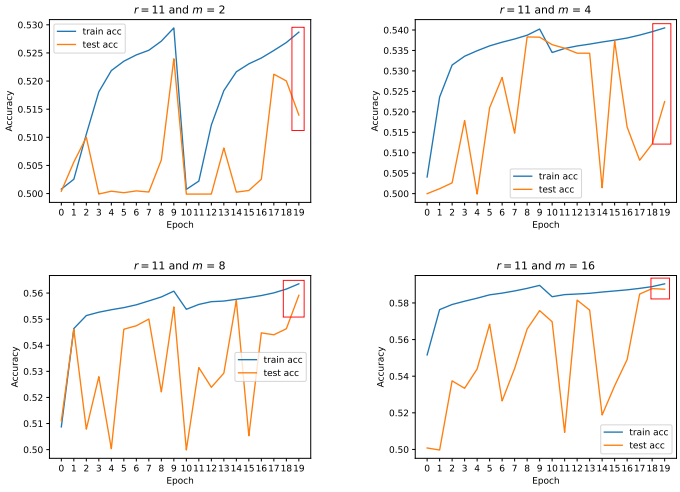


Figure 20: Using N instances for 11-round SIMON32/64