EasyChair Preprint

№ 4735

An Improved Integral Distinguisher Scheme Based on Deep Learning

Behnam Zahednejad and Jin Li

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

December 10, 2020

An Improved Integral distinguisher scheme based on Deep Learning

Behnam Zahednejad [1], Jin Li [2]

[1] *Institute of Artificial Intelligence and Blockchain, Guangzhou University; bzahednezhad@gmail.com*

[2] *Institute of Artificial Intelligence and Blockchain, Guangzhou University ;lijin@gzhu.edu.cn*

Abstract

At Crypto 2019, A.Gohr made a breakthrough in the fusion of differental cryptoanalysis and deep learning on Round Reduced Speck. Inspired by his methodology, we apply deep learning to construct a neural-based integral distinguisher scheme. To guage the advantage of our distinguisher scheme, we apply it on several block ciphers including SPECK32/64, SIMECK32/64, PRESENT,RECTANGLE and LBLOCK and compare it with bit-based division property as the state-of-the art integral distinguishing method. To our great surprise, our neural network based integral distinguisher extends the number of distinguished rounds of most block ciphers by at least 1 additional rounds compare to bit-based division property. In some cases, such as PRESENT block cipher, we could achive a 8-round distinguisher, while, bit-based division property could only provide a 5 round distinguisher. These observations illustrates that the fusion of deep learning and integral cryptanalysis has a promising prospect. In addition, we apply different deep learning architectures, namely ResNet,ResNeXt and DenseNet . DenseNet, as state of the art architecture, slightly outperforms the other architectures in terms of the distinguishing accuracy. Further, we showcase the utility of Few-Shot learning in reduction of training dataset to less than 100 instances. Finally, we show that our neural distiguisher is not only helpful for block cipher designers, but also assists attackers to mount key recovery attack. To this end, we show how to exploit our distinguisher to mount key recovery attack and apply it to SPECK32/64.

Keywords: Differential cryptoanalysis, Integral cryptoanalysis, division property, Deep learning, SPECK, LBLOCK, SIMECK, PRESENT, RECTANGLE, SIMECK

1. Introduction

Deep Learning is a class of machine learning algorithms which is inspired by learning architecture and information processing of the human brain. Novel Deep Learning architectures has made a great breakthrough in various fields ranging from machine translation [1, 2], autonomous driving [3] to playing professional abstract games beyond human level [4]. Theoretically, machine learning and cryptography have so much in common [5]. For instance, cryptographic hardness assumption can be framed as learning tasks. However, practically, the application of machine learning in cryptography has been mostly dedicated to side-channel analysis [6, 7]. It was not until the work of Gohr [8] at CRYPTO'19 that the fusion of deep learning with cryptanalysis produces attacks competitive to state of the art classical cryptanalytic methods. This work applies the learning capability of deep learning to developing the classical differential distinguisher. He developed a neural based differential distinguisher on round reduced SPECK. To this end, they trained neural networks to distinguish the output of Speck with a given input difference from random data.

1.1. Related work

Most of previous works have focused on the application of machine learning on side-channel attacks. In this section, we briefly review the few works done on the use of machine learning, cryptanalysis and statistical techniques for cryptanalytic purposes. Laskari et al. [9] applied evolutionary computing methods to recovering additional subkey bits in four and six round reduced DES after performing classical differential attack. Also, Klimov et al. applied genetic algorithms and neural networks to break a proposed public-key scheme and compared the results with two other methods. Gomez et al. [10] also achieved code book recovery for short-period Vigenere ciphers using unsupervised learning of neural networks. Although, their primary purpose had been to apply unsupervised learning techniques for machine translation. Also, Abadi et al. [11] tried to prevent reading communication traffic by training two neural networks. However, no cryptographic primitive nor cryptanalytic method was considered. Greydanus [12] demonstrated that recurrent neural networks are able to simulate the restricted version of Enigma in a black box setting. The breakthrough in the fusion of machine learning and cryptanalysis was brought by Gohr [8] at CRYPTO'19. He demonstrated that despite the common wisdom, machine learning has great potential in generating powerful cryptographic distinguishers. Further, he applied a deep residual neural network to mount differential attack on

nine rounds of Speck with a mean key rank roughly five times lower than an analogous classical distinguisher. Further, they illustrated that neural networks can be used to find optimum input differences using state of the art optimization methods.

1.2. Contributions and Structure of the Paper

This paper has three main contributions:

1. We exploit learning capability of deep learning to construct a neural network based Integral distinguisher. In this regard, we train an integral classifier which tries to distinguish the output of block ciphers with a predefined input multi-set from the output of a random multi-set. Further, we apply our neural-based integral classifier to several state-of-the art block ciphers, namely PRESENT [25] , RECTANGLE [39], LBLOCK [40], SPECK [38] and SIMECK [38] and compare the results with the most recent integral method. i.e. bit-based division property [14]: As shown in Table 1, our neural-based distinguisher can extend the number of distinguished rounds for most block ciphers by at least 1 additional rounds. In some block ciphers such as PRESENT, we could extend the distinguished round for 3 additional rounds. However, for SIMECK32, our neural-based distinguisher downgrades the distinguished rounds by 3 rounds.
2. We compare the performance of different state of the art CNN architectures namely, ResNet, ResNeXt and DenseNet in training our neural distinguisher. The results indicate that DenseNet is a more promising tool in construction of integral distinguisher with slightly higher distinguishing accuracy. We also demonstrate the utility of Few-Shot Learning in construction of neural based Integral distinguisher which can train our neural distinguisher with less than 100 dataset instances.
3. Our neural based integral distinguisher is not only helpful for block cipher designers, but also assists attackers to mount key recovery attack. To this end, we elaborate a key recovery attack based on our distiguisher in section 3 and mount it on block ciphers SPECK in section 4.

Table 1: Distinguishing attacks on several lightweight block ciphers using our neural-based integral distinguisher

Cipher	Block size	Round (previous)	Round (Our Neural distng)	Round (Few-Shot)
PRESENT	64	5 [14]	8	8
RECTANGLE	64	6 [14]	7	7
LBLOCK	64	8 [37]	10	10
SPECK32/64	32	3 [37]	5	5
SIMECK32	32	14 [37]	11	11

Structure of the Paper. In section 2, we give an overview of Integral cryptanalysis and classical methods of finding Integral distinguishers in block ciphers. In section 3, we develop our neural based Integral distinguishers on round reduced PRESENT. Section 4 contains our experimental results on the application of neural networks to finding Integral distinguishers on round reduced PRESENT. Finally, a conclusion is given in section 5.

2. Integral distinguisher

Integral distinguisher of block ciphers was first proposed by Daemen et al. to analyze the security of SQUARE [15], and later formalized by Knudsen and Wagner [16]. It is a chosen plaintext attack in which the adversary chooses a multi set of plaintext values that contains all possible values for some bits and a constant value for the rest of bits. Using the encryption oracle, the cipher is prone to an integral distinguisher if the XOR of the corresponding ciphertexts becomes 0. Many integral distinguisher have been discovered against different ciphers [17, 18, 19]. So far, three different methods were suggested to finding integral distinguisher against block ciphers,i.e. integral property [16, 17] , degree estimation [20] and division property [14, 18]. In the following, we briefly describe each method after reviewing the common notations in this context.

2.1. Notations

Denote the finite field with binary elements by $\mathbb{F}_2$. Also, the n-bit string over $\mathbb{F}_2$ is denoted by $\mathbb{F}_2^n$. Here, $\mathbb{Z}$ and $\mathbb{Z}^n$ represent the the integer ring and the set of all vectors whose coordinates are integers respectively. Assuming $a \in \mathbb{F}_2^n$, then

$a[i]$ denotes the i-th bit of a, and $\sum_{i=0}^{n-1} a[i]$ represents the hamming weight of a.

Bit Product Function $\pi_u(x)$ and $\boldsymbol{\pi_u(x)}$: $\pi_u(x)$ is a function from $\mathbb{F}_2^n$ to $\mathbb{F}_2$. For any $x \in \mathbb{F}_2^n$, the function $\pi_u(x)$ is defined as follows:

$$\pi_u(x) = \prod_{i=0}^{n-1} x[i]^{u[i]}$$

Also, for all $u \in \mathbb{F}_2^{n_0} * \mathbb{F}_2^{n_1} * \ldots * \mathbb{F}_2^{n_{m-1}}$, and $\boldsymbol{x} = (x_0, x_1, ..., x_{m-1}) \in \mathbb{F}_2^{n_0} * \mathbb{F}_2^{n_1} * \ldots * \mathbb{F}_2^{n_{m-1}}$ the function $\boldsymbol{\pi_u(x)}$ is defined as follows:

$$\boldsymbol{\pi_u(x)} = \prod_{i=0}^{m-1} \pi_{ui}(x_i)$$

2.2. Integral property

Integral property based distinguishers are constructed by evaluating the propagation characteristic of the integral property. Four integral properties are defined for every multiset as follows:

- ALL (A): Every possible value appears exactly once in the multiset.
- BALANCE (B): The XOR of every value in the multi set is 0, but every possible value may not necessarily appear in the multi set.
- CONSTANT (C): All texts in the multiset are fixed to a constant value.
- UNKNOWN (U): The texts in the multiset are randomly distributed such that they are indistinguishable from any n-bit random values.

However, this method fails to derive effective distinguishers against block ciphers if they consist non-bijective functions, e.g., DES [21] and Simon [22]. In addition, this method doesn't take into account the algebraic degree of block ciphers. As a result, it fails to derive effective distinguishers against block ciphers with low-degree round functions.

2.3. Degree estimation

Degree estimation or higher-order differential attack is another method of finding integral distinguisher which exploits the algebraic degree of block ciphers. The cipher has the integral distinguisher with 2^{D+1} chosen plaintexts, where D represents the maximum algebraic degree of the block cipher. By using this method, Boura et al. [23] discovered integral distinguisher against Keccak and Luffa, respectively.

2.4. Division property

As a generalized integral property method, division property finds better distinguishers by exploiting the properties hidden between traditional integral properties, i.e. ALL (A) and BALANCE (B). Let $\mathbb{X}$ represent the multi-set in which the value of its members are in $(\mathbb{F}_2^n)^m$, and k be a m-dimensional vector whose components lie between 0 and n. The multiset $\mathbb{X}$ has division property $\mathfrak{D}^{n,m}_{k^{(0)},k^{(1)},...,k^{(q-1)}}$ if and only if the parity of $\boldsymbol{\pi_u(x)}$ over all $x \in \mathbb{X}$ is always even for $u \in \{(u_0, u_1, ..., u_{m-1}) \in (\mathbb{F}_2^n)^m \mid W(u) \not\succ k^{(0)}, ..., W(u) \not\succ k^{(q-1)}\}$

In [14], Xiang et al. applied MILP to automatically find integral distinguisher of block ciphers. Assuming $\mathfrak{D}^{n,m}_k$ as the division property of the initial multiset in a block cipher with round function f_r, the division property of the state after r-th round ,$\mathfrak{D}^{n,m}_{\mathbb{K}_r}$, can be derived using the propagation rules described in [14]. Here, $\mathbb{K}_r$, represents the set of m-dimensional vectors. Thus, the block cipher transforms the initial k to the set of vectors in $\mathbb{K}_r$. In another words, assuming $\mathfrak{D}^{n,m}_{\mathbb{K}_i}$ as the division property after i-round propagation through f_r, the following chain of division property can be derived in a block cipher with round function f_r:

$$\{k\} \overset{def}{=} \mathbb{K}_0 \xrightarrow{f_r} \mathbb{K}_1 \xrightarrow{f_r} \mathbb{K}_2 \xrightarrow{f_r} .... \xrightarrow{f_r} \mathbb{K}_r$$

For $(k_0, k_1, \ldots, k_r) \in \mathbb{K}_0 \times \mathbb{K}_1 \times \ldots \times \mathbb{K}_r$, if k_{i-1} can propagate to k_i for all $i \in \{1, 2, \ldots, r\}$, the vector set $(k_0, k_1, \ldots, k_r)$ is called a r-round division trail. Here, $\mathbb{K}_r$ denotes the set of the last vectors of all r-round division trails, starting from k.

Proposition (Set without Integral Property). The multiset $\mathbb{X}$ with division property $\mathfrak{D}^{1,n}_{\mathbb{K}}$ doesn't have any useful integral property (Xor-sum of $\mathbb{X}$ doesn't balance on any bit) if and only if $\mathbb{K}$ contains all the n unit vectors.

To prove this proposition, assume that $\mathbb{K}_{r+1}$ contains all the n unit vectors for the very first time. In this case, a r-round distinguisher can be derived from $\mathfrak{D}^{1,n}_{\mathbb{K}_r}$. Since, $\mathbb{K}_r$ doesn't contain all the n unit vectors, a unit vector like e can always be found such that $e \notin \mathbb{K}_r$. As a result, for all $k \in \mathbb{K}_r$, it holds that $e \not\succ k$. So, the parity of $\boldsymbol{\pi_u(x)}$ is even. As a result, a balanced bit is found. In a similar fashion, any missing unit vectors represent a balanced bit. Based on this rule, to find out whether or not an r-round block cipher has integral property, it is sufficient to check the vectors in $\mathbb{K}_r$, or equivalently, to check the last vectors of all r-round division trails. This problem resembles a typical MILP problem [24] in which the variables are the components of the r-round division trail, i.e.

neural networks (CNN) architecture namely, ResNet,ResNeXt and DenseNet to mount integral distinguishing attack on block ciphers. We also illustrate the utility of few-shot learning in training our neural network based integral distinguisher framework with very low training data, i.e. less than 50 instances. Mounting distinguishing attack is only useful for the block cipher designers. In order to pave the way for the attackers to utilize our scheme, we further illustrate the feasibility of key recovery attack using our neural integral distinguisher framework.

3.1. Training our neural based Integral distinguisher

To train our neural based integral distinguisher framework, we develop a binary classification problem in which the corresponding plaintext of every ciphertext is either an instance of a multi-set with specific integral property or a set of random plaintexts. To this end, 2^{20} multi-sets of plaintext were generated and encrypted as the training data, as well as a corresponding vector of binary-valued labels Y. Here, $Y = 1$ corresponds to the real multi-set where every instance of the multi-set follows a specific integral property. Whereas, $Y = 0$ corresponds to the random multi-set wherein every instance of the multi-set is generated in a random fashion. Figure 1 illustrates two multisets with different Y. The multiset denoted by $Y = 1$ corresponds to integral properties of 1 active and 3 constant bits. Whereas, $Y = 0$ corresponds to the random multi-set in which no specific integral property is chosen.

Figure 1: Dataset instances of our neural based Integral distinguisher: The multiset denoted by $Y = 1$ corresponds to integral properties of 1 active and 3 constant bits. Whereas, $Y = 0$ corresponds to the random multi-set in which no specific integral property is chosen.

Further, to validate the trained distinguisher, 2^{16} multi-sets of plaintext and their corresponding labels were generated similarly. In our implementation, it's very cheap to generate the training data. Such that it takes only a few seconds

to generate a data set of size 2^{20}.

Training Pipeline
We run our training for 10 epochs with a dataset of size 10^7. The datasets were processed in batches of size 5000 with a validation ration of 30 %. Mean square error loss function was used during the optimization with L2 weights regularization using the Adam algorithm [36]. By using a cyclic learning rate schedule, the learning rate l_i for epoch i is set to $l_i := \alpha + \frac{(n-i) \mod(n+1)}{n}(\beta - \alpha)$, with $\alpha = 10^{-4}, \beta = 2 \times 10^{-3}$ and n = 9. The best trained network was further evaluated against a test set of size 10^6 that is not used in training or validation.

3.2. Convolution neural networks (CNN)

Convolutional Neural Network (CNN, or ConvNet) is a type of deep neural networks, which are widely applied in image and video recognition, recommender systems [26], image classification, medical image analysis, natural language processing, and financial time series [27]. CNN consists of an input and an output layer with multiple hidden layers in between The hidden layers usually consist of a series of convolutional layers that are convolved by multiplication. The input is a tensor of size (number of images) $\times$ (image height) $\times$ (image width) $\times$ (image depth). Upon passing through each convolutional layer, the input is abstracted to a feature map of size (number of images) $\times$ (feature map height) $\times$ (feature map width) $\times$ (feature map channels). Each convolutional layer has the following attributes:

- Convolutional kernels with specific width and height defined by hyper-parameters.
- The number of input channels and output channels defined by hyper-parameters
- Convolution filter (the input channels) depth should be equal to the channels (depth) of the input feature map.

Similar to the response of a neuron in the visual cortex to a specific stimulus, Convolutional layers convolve the input and output the result to the next layer. In all CNN architectures, the convolutional layers are applied successively to the input, downsampling the spatial dimensions periodically while adding the number

of feature maps. Classical network architectures simply consist of stacked convolutional layers. Examples of classic network architectures include LeNet-5 [28], AlexNet [29] and VGG 16 [30]. However, modern architectures use novel ideas to increase the accuracy of the learning. Inception [31], ResNet [32], ResNeXt [34], DenseNet [35] are among modern network architectures:

- **ResNet**
  Deep Residual Networks (ResNet) [32] were a breakthrough in development of neural networks. The motivation was the observation of researchers that despite intuition, adding more layers not only didn't improve the performance, but downgrades the final performance. This phenomenon is known as degradation problem or vanishing gradients [33]. Researchers suggested residual blocks as a solution to this problem in which intermediate layers utilize skip connections to learn a residual function with reference to the input block (Figure 2). Skipping is a simple and effective strategy which speeds learning by using fewer layers .In this way, we reuse activation from a previous layer until the adjacent layer learns its weights. Figure 2 illustrates the application of residual blocks to forming a residual network.

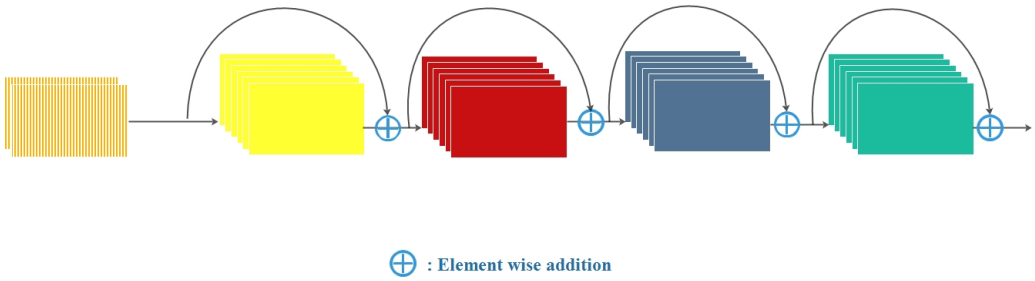

Figure 2: A schematic of ResNet architecture

- **ResNeXt**
  The ResNeXt architecture is an extension of the deep residual network which leverages a "split-transform-merge" strategy as a replacement to the standard residual block [34]. In simple language, instead of convolving over the full input feature space, a few convolutional filters is applied over a

series of lower channel dimensional representation of the input block. This idea is similar to the group convolutions previously suggested in AlexNet Paper [29], in which the input is divided into groups in channel-wise manner, instead of applying filters with the input's full channel depth. Figure 3 illustrates the difference of standard convolution vs. group convolution. In the original ResNeXt paper, a series of experiments are performed to figure out the relation of the increase of the number of groups (also known as cardinality), depth and width with the performance accuracy. The results indicate that increasing the cardinality is a more effective measure to gain a higher accuracy rather than increasing the width or depth of the network.

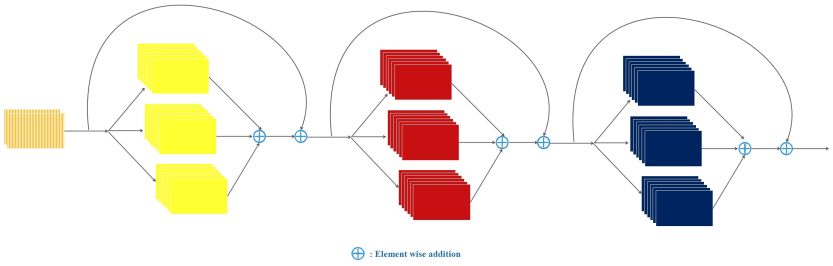

Figure 3: A schematic of ResNeXt architecture

- **DenseNet**
  In dense convolutional networks, the feature map of each layer is concatenated to the input of every successive layer within a dense block. This technique paves the way for the deeper layers to reuse and exploit the features of earlier layers. As argued by the author [35], feature-map concatenation results in higher performance, as the variation of the input of subsequent layers is increased. In this way, the network is able to directly use any previous feature map. The number of filters in each convolutional layer is referred to as growth rate", k, as each successive layer will bear k more channel. A schematic of DenseNet architecture is shown in Figure 4. Compared to the ResNet model, DenseNets provide higher accuracy and performance with a lower data complexity.

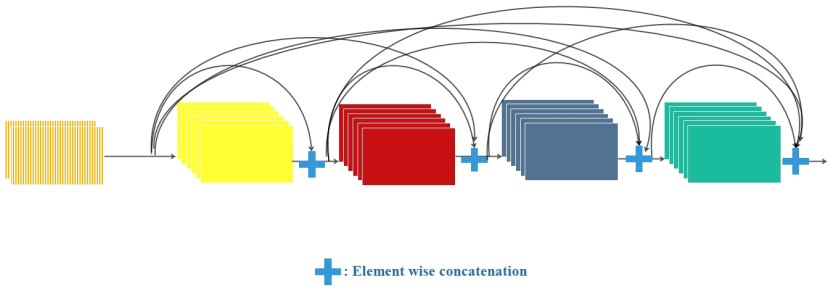

Figure 4: A schematic of DenseNet architecture

3.3. Application of Few-Shot learning in training our neural based integral distinguisher

As its name suggests, few-shot learning refers to training a model with sparse amount of training data, as opposed to training a conventional model with a large amount of data. Suppose that we want to distinguish a block cipher of block size b reduced to r rounds. To test whether our neural network can be trained via few-shot learning, we performed the following experiment:

1. A simple neural network, N, with only one residual block is trained to distinguish the target block cipher reduced to $s << r$ rounds with a specific input integral property. A single epoch of 2000 descent steps was used during the training with a batch size of 5000.
2. The output of the second-to-last layer of the trained network ,N[-2], is preserved as a representation of the input data.
3. Small samples S (between 1-50) of the r round target block cipher's output multi-set with an input integral property $\mathfrak{D}_{in}$ of n active bits ,X_1, was generated along with the corresponding labels Y_1. This sample was complimented with the same number of ciphertext drawn from random input multiset,X_0, which corresponds to label Y_0.
4. The sample S is treated as the input of the trained neural network in step 2. The output of this network is denoted as Z.
5. The r-round distinguisher is achived by applying ridge regression (with regularization parameter $\alpha = 1$) between labels $Y = Y_1||Y_0$ and predictions ,Z. To this end, the linear predictor $L : \mathbb{R}^{b\times 2^n} \rightarrow \mathbb{R}$ determines whether a given ciphertext corresponds to a real input multi-set with a specific integral property if $L(x) > 0.5$. Otherwise, it belongs to a random multi-set. To estimate the accuracy of this predictor, it was tested on a sample of size 50000. This experiment is summarized in Algorithm 2.

Algorithm 2: Application of Few-Shot learning in training an integral distinguisher with input division property $\mathfrak{D}_{in}$ for a block cipher with block size b reduced to r rounds E^r given an auxiliary integral distinguisher N for a block cipher reduced to s rounds E^s with input division property $\mathfrak{D}_{in}$

Require : N, r, t

1.$X_1 \leftarrow t$ multiset outputs of block cipher E^r with input division property $\mathfrak{D}_{in}$ of n active bits
2. $Y_1 \leftarrow (1, 1, ..., 1) \in \mathbb{R}^t$
3. $X_0 \leftarrow t$ multiset outputs of block cipher E^r with random input division property
4. $Y_1 \leftarrow 0 \in \mathbb{R}^t$
5. $N' \leftarrow N[-2]$ Here,N[-2] represents the second-to-last layer of N
6. $Z, Y \leftarrow N'(X_0||X_1), Y_0||Y_1$
7. $L \leftarrow RidgeRegression(Z, Y)$
8.**Return** $L(N'(X_0||X_1))$

3.4. Key Recovery attack

In this section, we show that our neural based integral distinguisher is not only useful for block cipher designer, but also is a handy tool for attckers to recover round key bits. Given a r-round block cipher neural distinguisher for an input multi-set with n active bits, we recover the (r+1)-th subkey of the block cipher,E^{r+1}, as follows:

1. Ask oracle for the r+1 round encryption of 2^n plaintext with n active bits. Obtain the corresponding ciphertexts as $C_1, ..., C_{2^n}$.
2. For each possible value of the final subkey k, decrypt each ciphertext C_i under k to get C_i^k.
3. Using the r-round neural distinguisher, get score Z_i^k for each partially decrypted ciphertext.
4. For each k, denote v_k as the combination of scores Z_i^k for all 2^{n+1} ciphertexts. We use the following formula to obtain the score of each candidate key v_k:

$$v_k := \sum_{i=1}^{n} log_2(Z_i^k/(1 - Z_i^k)) \tag{1}$$

5. The subkey with the highest score v_k is chosen as the correct final round key.

To gauge the accuracy of the key recovery attack, we performed the key recovery attack for 100 times on each block cipher. Each time, we report the number of candidate subkeys ranked higher than the true subkey. In the next section, we report the result of key recovery attack on SPECK block cipher. Due to hardware limtation, we couldn't apply key recovery attack on other block ciphers with key length greater than 16 bits.

4. Applications to PRESENT,RECTANGLE,LBLOCK,SPECK and SIMECK

In this section, we apply our neural based integral distinguisher to PRESENT, RECTANGLE, LBLOCK, SPECK and SIMECK block ciphers. The results are listed in Table 1. In all experiments, the number of active bits of the input pattern is 4 bits. The Round (Previous) column and Round (Our Neural distng) column represent the number of extinguished rounds of previous methods (bit based division property) and our neural based distinguisher. Also, the column Round (Few shot learning) represent the result of applying few shot learning on each cipher. The table shows that our neural-based distinguisher extends the number of distinguished rounds for all block ciphers(except for SIMECK32). All the experiments are performed in Google Colab GPU VM platform [1]. Also, all the source codes are available at [37] . To better guage the capability of CNN in cryptoanalysis, we apply different CNN architectures namely ResNet, ResNeXt and DensNet to train each block cipher and compare the accuracy after 10 epochs. The results are shown in Table 2. As expected, DensNet, as the state-of-the art CNN architecture, provides the most accurate distinguisher for all block ciphers. We claim that we obtain a distiguisher only if the distinguisher accuracy is more than 52 %. Further, we mount key recovery attack on SPECK32/64 block cipher. Due to resource and hardware limitation, we were unable to mount key recovery attack on block ciphers with round key size larger than 30 bits.

[1] https://colab.research.google.com/

Table 2: Comparing the accuracy of different CNN architectures in distinguishing block cipher after 10 epochs

Cipher	distinguished round	ResNet	ResNeXt	DenseNet
PRESENT	8	78 %	82 %	86 %
RECTANGLE	7	76 %	81 %	85 %
LBLOCK	10	81 %	85 %	86 %
SPECK32/64	5	98 %	99 %	99 %
SIMECK32	11	67 %	70 %	72 %

4.1. Applications to SPECK & SIMECK

SPECK is a family of lightweight block ciphers publicly released by the National Security Agency (NSA) in June 2013 [38]. SPECK is an add–rotate–xor (ARX) cipher which supports a variety of block and key sizes. The block always consists of two words of 16, 24, 32, 48 or 64 bits in size. Also, the corresponding key is 2, 3 or 4 words. The round function includes two rotations, adding the right word to the left word, xoring the key with the left word and then xoring the left word with the right word. Assuming the cipher state (L_i, R_i) as the input, the round function produces the state (L_{i+1}, R_{i+1}) as follows:

$$L_{i+1} := ((L_i \gg \alpha) \boxplus R_i) \oplus K, R_{i+1} := ((R_i \ll \beta) \oplus L_{i+1} \quad (2)$$

where α, β are constants specific to each member of the Speck cipher family. We apply state of the art CNN architecture to train our neural based integral distinguishing scheme on SPECK32/64 block cipher. Also, we implement bit-based division property in [37] to compare it with our neural based distinguisher. Using bit-based division property, we found a 3 round distinguisher for SPECK32/64 block cipher with 24 balanced output bits. This distinguisher corresponds to an input multi-set of 4 active bits in the right word, as illustrated below:

Input division: (cccccccccccccccc,ccccccccccccaaaa)
Output: (b???bbbbbbbbbbb?,b???bbbbbbbbbbb?)

c: constant bit , a: active bit , ?: unknown bit , b: balanced bit

SIMECK [38] is also another ARX based block cipher which inherits good design components of Speck. Using bit-based division property, we found the following 14-round distinguisher with 3 balanced output bits:

Input division: (ccccccccccccccc,ccccccccccccaaaa)
Output: (b?????????????b,b?????????????)

As shown in Table 1, for SPECK32/64, our neural based integral distinguisher can extend the number of distinguished rounds from 3 to 5 with the same data complexity i.e. 2^4.

Also, for SIMECK32 we could achieve a 11-round distiguisher which is 3 rounds less than the bit-based division property.

Further, we performed few-shot learning using a pre-trained auxiliary network reduced to three and 10 rounds for SPECK32/64 and SIMECK32, respectively. As shown in Table 1, we could achieve a 5-round distinguisher using a 2 digit number of training examples for SPECK and a 11-round distinguisher for SIMECK.

In addition, the results of the key recovery attack on 6-round SPECK32/64 are shown in Table 3.

Table 3: Result of the key recovery attack on 6-round SPECK32/64. The reported values are based on 100 trials of key recovery attack. The rank of a key is defined as the number of subkeys ranked higher than the real subkey.

Cipher	**zero key rank**	**key rank less than 10**	**key rank less than 100**
SPECK32/64	3	12	42 %

4.2. Applications to PRESENT & RECTANGLE

In this section, we apply state of the art CNN architecture to train our neural based integral distinguishing scheme on PRESENT and RECTANGLE block cipher. PRESENT [25] and RECTANGLE [39] are two SP-network block ciphers, which use bit permutations in their linear layers. PRESENT is a SPN block cipher suitable for extremely constrained environments [25]. As stated by the authors: "we are not building a block cipher that is necessarily suitable for wide-spread use; we already have the AES for this. Instead, we are targeting some very specific applications for which the AES is unsuitable."

By supporting both encryption and decryption with the same physical requirements, the PRESENT cipher turns to a lightweight block cipher that is still smaller than an encryption-only AES. In addition, the encryption-only present is an ultra-lightweight cipher. PRESENT consists of 31 rounds with a block length

of 64 bits which supports two key lengths of 80 and 128 bits. In each of the 31 rounds, the round key K_i for $1 \leq i \leq 31$ is XORed with the block input, where K_{32} is applied for post-whitening. Then, a linear bitwise permutation and a non-linear substitution layer is applied in each round. The non-linear substitution layer consists of a 4-bit S-box which is applied 16 times for the 64 bits of the input. A schematic of PRESENT cipher is shown in Figure 5.

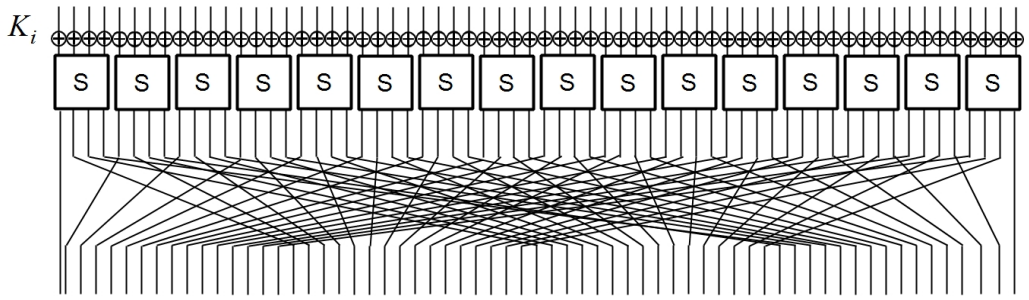

Figure 5: A schematic of PRESENT block cipher

In the following, we discuss each operation in detail:

addRoundKey. This layer consists of xor operation between the current state $b_{63} \dots b_0$ and the round key $K_i = k^i_{63} \dots k^i_0$ for $1 \leq i \leq 32$.

$$b_j \rightarrow b_j \oplus k^i_j \text{ for } 0 \leq j \leq 63$$

sBoxlayer. The S-box used in PRESENT cipher is a 4-bit to 4-bit S-box: $\mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4$. Table 4 shows the detail of this box in hexadecimal notation.

Table 4: S-Box of PRESENT cipher

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S[x]	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

However, this table is hard to be implemented in neural networks [2]. A bet-

[2]Specially, for large number of inputs, it takes days to execute the same box for each input. A better way to increase the speed of the implementation is to use the parallelism in python using Numpy array which can computes operations like xor, addition, rotation etc. in a parallel manner

ter way to increase the implementation speed is to use algebraic normal form (ANF) of PRESENT Sbox. Assuming $x = (x_3, x_2, x_1, x_0)$ as the input and $y = (y_3, y_2, y_1, y_0)$ as the corresponding output, the ANF representation of PRESENT Sbox would be:

$$y_3 = 1 \oplus x_0 \oplus x_1 \oplus x_3 \oplus x_1x_2 \oplus x_0x_1x_2 \oplus x_0x_1x_3$$
$$y_2 = 1 \oplus x_2 \oplus x_3 \oplus x_0x_1 \oplus x_0x_3 \oplus x_1x_3 \oplus x_0x_1x_3 \oplus x_0x_2x_3$$
$$y_1 = x_1 \oplus x_3 \oplus x_1x_3 \oplus x_2x_3 \oplus x_0x_1x_2 \oplus x_0x_1x_3 \oplus x_0x_2x_3$$
$$y_0 = x_0 \oplus x_2 \oplus x_3 \oplus x_1x_2$$

The input of the sBoxLayer is $b_{63}\ldots b_0$ which is considered as sixteen 4-bit words, wherein the 4-bit S-box is applied on each word.

pLayer. This linear layer is a bit permutation layer which acts on the output of the S-box layer of i-th round $(b^i_{63}\ldots b^i_0)$ and the input of the (i+1)-th round $(a^{i+1}_{63}\ldots a^{i+1}_0)$. These variables are related as follows:

$$a^{i+1}_{16j \ mod \ 63} = b^i_j \text{ for } j \in \{0, 1, ..., 62\}$$
$$a^{i+1}_j = b^i_j \text{ for } j = 63$$

Previously, Wu et al.[13] found a 5-round distinguisher for PRESENT block cipher with 1 balanced output bit. In the input multi-set of this distinguisher, the left most 60 bits are fixed to random constants and the 4 right most bits are active bits. Using the same input multi-set, Xiang et al. [14] found another 5 round distinguisher with 4 balanced output bits :

Input: (ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccaaaa)
Output: (??bbbb)

RECTANGLE also has similar cipher structure [39], wherein the input block is arranged in a 4*16 matrix. Using bit-based division property, Xiang et al. [14] found the following 6 round distinguisher with 23 active bits:

$$\text{Input: } \begin{pmatrix} cccccccccccaaaaaa \\ cccccccccccaaaaaa \\ cccccccccccaaaaaa \\ cccccccccccaaaaaa \end{pmatrix} \rightarrow \text{Output: } \begin{pmatrix} bbbbbbbbbbbbbbbb \\ bbbbbbbbbbbbbbbb \\ ??????????????bb \\ bbbbbb?????????? \end{pmatrix}$$

Surprisingly, for PRESENT block cipher, our neural based integral distin-

Output: (????bbbb????bbbb????bbbb?b??bbbb,??????????????????????????????)

To our great surprise, our neural-based integral distiguisher discovers a 10-round distiguisher which is 2 rounds more than the previous method. In both methods, only 4 bits of the input pattern are active. In addition, using few-shot learning with a pre-trained auxiliary network reduced to 8 rounds, we could achieve a 10-round distinguisher using only 100 training examples.

5. Conclusion

In this paper, we exploited learning capability of deep learning to develop an integral distinguisher scheme and applied it to several block ciphers including SPECK, LBLOCK, SIMECK , PRESENT, RECTANGLE and SIMECK. The results indicate that our neural based integral distinguisher scheme outperforms the state of the art classical integral cryptoanalysis method (bit based division property).For most block ciphers, it extends the number of distinguished rounds for at least 1 additional rounds. In some cases, such as PRESENT cipher, it provides an additional 3 rounds distinguisher. We also observed the superiority of the most recent deep learning architecture, i.e. DenseNet over previous architectures ,i.e. ResNet and ResNeXt in terms of distinguishing accuracy. Further, we show how to use our distinguisher scheme to mount key recovery attack which improves the utility of our scheme.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473,2014.

[2] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi,Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144, 2016.

[3] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deep driving: Learning affordance for direct perception in autonomous driving. In Computer Vision (ICCV), 2015 IEEE International Conference on, pages 2722-2730. IEEE, 2015

[4] Christopher Clark and Amos Storkey. Training deep convolutional neural networks to play go. In International Conference on Machine Learning, pages 1766-1774,2015.

[5] Ronald L Rivest. Cryptography and machine learning. In International Conference on the Theory and Application of Cryptology, pages 427-439. Springer, 1991.

[6] Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. In International Conference on Security, Privacy, and Applied Cryptography Engineering, pages 3-26.Springer, 2016

[7] Stjepan Picek, Ioannis Petros Samiotis, Jaehun Kim, Annelie Heuser, Shivam Bhasin, and Axel Legay. On the performance of convolutional neural networks for side-channel analysis. In International Conference on Security, Privacy, and Applied Cryptography Engineering, pages 157-176. Springer, 2018.

[8] Gohr, Aron. "Improving Attacks on Round-Reduced SPECK32/64/64 Using Deep Learning." Annual International Cryptology Conference. Springer, Cham, 2019.

[9] Elena Laskari, Gerasimos Meletiou, Yannis Stamatiou, and Michael Vrahatis. Cryptography and cryptanalysis through computational intelligence. In Computational Intelligence in Information Assurance and Security, pages 1-49. Springer,2007.

[10] Aidan N. Gomez, Sicong Huang, Ivan Zhang, Bryan M. Li, Muhammad Osama, and Lukasz Kaiser. Unsupervised cipher cracking using discrete GANs. In International Conference on Learning Representations, 2018.

[11] Martin Abadi and David G Andersen. Learning to protect communications with adversarial neural cryptography. arXiv preprint arXiv:1610-06918, 2016

[12] Sam Greydanus. Learning the enigma with recurrent neural networks. arXiv preprint arXiv:1708-07576, 2017.

[13] Wu, Shengbao, and Mingsheng Wang. "Integral attacks on reduced-round PRESENT." International Conference on Information and Communications Security. Springer, Cham, 2013.

[14] Xiang, Zejun, et al. "Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers." International Conference on the Theory and Application of Cryptology and Information Security. Springer, Berlin, Heidelberg, 2016.

[15] Daemen, Joan, Lars Knudsen, and Vincent Rijmen. "The block cipher Square." International Workshop on Fast Software Encryption. Springer, Berlin, Heidelberg, 1997.

[16] Knudsen, Lars, and David Wagner. "Integral cryptanalysis." International Workshop on Fast Software Encryption. Springer, Berlin, Heidelberg, 2002.

[17] Li, Ping, Bing Sun, and Chao Li. "Integral cryptanalysis of ARIA." International Conference on Information Security and Cryptology. Springer, Berlin, Heidelberg, 2009.

[18] Funabiki, Yuki, et al. "Improved integral attack on HIGHT." IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 102.9 (2019): 1259-1271.

[19] Todo, Yosuke. "Structural evaluation by generalized integral property." Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 2015.

[20] Li, Wu, Zhang, : Improved integral attacks on reduced-round CLEFIA block cipher. In: Jung, S., Yung, M. (eds.) WISA 2011. LNCS, vol. 7115, pp. 28–39. Springer, Heidelberg (2012)

[21] Knudsen,: Truncated and higher order differentials. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 196–211. Springer, Heidelberg (1995)

[22] National Institute of Standards and Technology: Data Encryption Standard (DES).Federal Information Processing Standards Publication 46 (1977)

[23] Beaulieu, Ray, et al. "The SIMON and SPECK Families of Lightweight Block Ciphers." IACR Cryptol. ePrint Arch. 2013 (2013): 404.

[24] Chachuat. ”Mixed-Integer Linear Programming (MILP): Model Formulation, ChE 4G03: Optimization in Chemical Engineering, McMaster University, Department of Chemical Engineering.” (2009).

[25] Boura, Christina, Anne Canteaut, and Christophe De Canniere. ”Higher-order differential properties of Keccak and Luffa.” International Workshop on Fast Software Encryption. Springer, Berlin, Heidelberg, 2011.

[26] Bogdanov, Andrey, et al. ”PRESENT: An ultra-lightweight block cipher.” International workshop on cryptographic hardware and embedded systems. Springer, Berlin, Heidelberg, 2007.

[27] Van den Oord, Aaron, Sander Dieleman, and Benjamin Schrauwen. ”Deep content-based music recommendation.” Advances in neural information processing systems. 2013.

[28] Collobert, Ronan, and Jason Weston. ”A unified architecture for natural language processing: Deep neural networks with multitask learning.” Proceedings of the 25th international conference on Machine learning. 2008.

[29] LeCun, Yann, et al. ”Gradient-based learning applied to document recognition.” Proceedings of the IEEE 86.11 (1998): 2278-2324.

[30] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. ”Imagenet classification with deep convolutional neural networks.” Advances in neural information processing systems. 2012.

[31] Simonyan, Karen, and Andrew Zisserman. ”Very deep convolutional networks for large-scale image recognition.” arXiv preprint arXiv:1409.1556 (2014).

[32] Huang, Gao, et al. ”Densely connected convolutional networks.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[33] Goh, Garrett; Hodas, Nathan ; Vishnu, Abhinav. ”Deep learning for computational chemistry”. Journal of Computational Chemistry. 38 (16): 1291–1307. arXiv:1701.04503. Bibcode:2017arXiv170104503G. doi:10.1002/jcc.24764. PMID 28272810, (15 June 2017).

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770-778, 2016.

[35] Xie, Saining, et al. "Aggregated residual transformations for deep neural networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[36] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[37] https://github.com/zahednejadb/Neural-distinguisher

[38] Beaulieu, Ray, et al. "The SIMON and SPECK Families of Lightweight Block Ciphers." IACR Cryptol. ePrint Arch. 2013 (2013): 404.

[39] Zhang, Wentao, et al. "RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms." Science China Information Sciences 58.12 (2015): 1-15

[40] Wu, Wenling, and Lei Zhang. "LBlock: a lightweight block cipher." International Conference on Applied Cryptography and Network Security. Springer, Berlin, Heidelberg, 2011.