A Deep Neural Differential Distinguisher for ARX based Block Cipher

Debranjan Pal[1*], Upasana Mandal[1†], Mainak Chaudhury[1†], Abhijit Das[1†] and Dipanwita Roy Chowdhury[1†]

[1]Crypto Research Lab, Department of Computer Science and Engineering,, IIT Kharagpur, Kharagpur, 721301, West Bengal, India.

*Corresponding author(s). E-mail(s): debranjanpal@iitkgp.ac.in; Contributing authors: mandal.up98@gmail.com; mainakcacsgu.gmail.com; abhij@cse.iitkgp.ac.in; drc@cse.iitkgp.ac.in;

†These authors contributed equally to this work.

Abstract

Over the last few years, deep learning is becoming the most trending topic for the classical cryptanalysis of block ciphers. Differential cryptanalysis is one of the primary and potent attacks on block ciphers. Here we apply deep learning techniques to model differential cryptanalysis more easily. In this paper, we report a generic tool using deep neural classifier that assists to find differential distinguishers for block ciphers with reduced round. We apply this approach for the differential cryptanalysis of ARX-based encryption schemes HIGHT, LEA, and SPARX. The result shows that our deep learning based distinguishers work with high accuracy for 14-round HIGHT, 13-Round LEA and 11-round SPARX. We also achieve an improvement of the lower bound of data complexity for these three ARX based ciphers.

Keywords: HIGHT, LEA, SPARX, Neural Distinguisher, Deep Learning, Differential Cryptanalysis

1 Introduction

Deep neural networks are known as non-linear classification tools, are famous for solving a more comprehensive set of data-driven tasks like image processing, speech recognization, etc. Earlier in the field of cryptanalysis, machine learning is mainly restricted to side channel analysis and not explored much in classical cryptanalysis. The direction of research finally gets noticed when a work on cipher SPECK by Aron Gohr is published in CRYPTO'19[1], where the main idea is to perform a key recovery attack on round-reduced SPECK using ML. The ML model mainly applies the differential distinguisher properties of differential cryptanalysis. Differential cryptanalysis actually finds an input and output difference pair that occurs with some higher probability than the random case. The researchers apply the probability distribution modeling for differential distinguisher by incorporating machine learning algorithms. Rather than the old modeling technique with branch number or MILP, the attacker can explore any other strategy to distinguish the cipher from the random case by collecting the output differences corresponding to the chosen input differences. The attackers can also use machine learning based models to reduce the search complexity and hence the attack time also reduce than the estimation of the existing methods. In 1993, Ronald L. Rivest first shows the relationships between the two fields of cryptography and machine learning and explains how each area contributes ideas and techniques to the other. At CRYPTO'19, Aron Gohr proposes a new direction in the cryptanalysis field based on utilizing machine learning algorithms. He built a deep neural network based distinguisher that surprisingly surpassed state-of-the-art cryptanalysis efforts on one of the versions of the NSA block cipher SPECK [2]. In DATE 2021, A Baksi et al. [3] given neural differential distinguishers for 10-round Knot-256 permutation, 12-round Knot-512 permutation, four-round Chaskey-Permutation, eight-round Gimli-Hash/Cipher/permutation, and three-round Ascon permutation. In LATINCRYPT 2021, Yadav et al[4] applied the technique to find neural based classifier for 12-round SPECK-32 [2], eight-round GIFT-64 [5] and 12-round SIMON-32 [2].

In this paper, we introduce a generic deep learning based automated differential distinguisher. Using the tool we analyze the differential behaviour of three ARX based cipher, HIGHT [6], LEA [7] and SPARX [8]. We found an 14-round neural differential distinguisher for HIGHT, an 13-round distinguisher for LEA and an nine-round distinguisher for SPARX. We achieve a new lower bound of data complexity for reduced rounds of HIGHT, LEA and SPARX.

The rest of the paper is organized as follows. Section 2 describes the basic notations we use in our paper and the brief specifications of HIGHT, LEA, and SPARX cipher. In Section 3, we present our generic tool for constructing the deep neural classifier. Section 4 represents the implementation results. In Section 5, we conclude the paper.

2 Preliminaries

2.1 Basic Notations

All the basic notations we use throughout this paper are being stated below,

- Plaintext is of 64 bits and it is denoted as the concatenation of 8 bytes. Let P is the given plaintext, it is denoted by $\mathrm{P} = \mathrm{P}_7\mathrm{P}_6\mathrm{P}_5\mathrm{P}_4\mathrm{P}_3\mathrm{P}_2\mathrm{P}_1\mathrm{P}_0$.
- Ciphertext is of 64 bits and it is regarded as the concatenation of 8 bytes. Let C is the given plaintext, it is denoted by $\mathrm{C} = \mathrm{C}_7\mathrm{C}_6\mathrm{C}_5\mathrm{C}_4\mathrm{C}_3\mathrm{C}_2\mathrm{C}_1\mathrm{C}_0$.
- We repressent the 64-bit temporary state values as , $T_i = T_{i,7} \dots T_{i,1}T_{i,0}, i \in 0, \dots, 32$.
- The master key is of 128 bit and it is regarded as the concatetaion of 16 bytes. Considering master key as M, it is denoted by $\mathrm{M} = \mathrm{M}_{15}\mathrm{M}_{14} \dots \mathrm{M}_1\mathrm{M}_0$.
- $W_i, i \in 0, \dots, 7$ is the eight bytes of whitening key.
- $RK_i, i \in 0, \dots, 127$ is the 127 bytes of round key.
- δ_i and δ_o represents input and output difference for a differential characteristics.
- $\mathcal{CD}$ signifies the set of ciphertext differences from a neural classifier.
- d is a random state difference.
- $\boxplus$ and $\boxminus$ means addition and subtraction in mod 2^8.
- $\oplus$ means EXOR
- $A^{\lll s}$ means s-bit left rotation of a 8-bit value A

2.2 Brief Description of HIGHT, LEA and SPARX

ARX(Addition, Rotation, and XOR) construct forms a special type of symmetric key architecture that uses modular addition, bitwise rotation, and exclusive xor. In this type, the primary source of non-linearity comes from modular addition. HIGHT, SPARX and LEA are ARX-based block ciphers.

HIGHT HIGHT is an ARX-based block cipher proposed by Hong et al. [6] at CHESS 2006. It is a lightweight cipher, and it performs fast. As it is an ARX cipher, hence it adopts the feistel structure. The round function consists of simple operations like circular left rotation, bit-wise xor, and addition in modulo 2^8. The size of plaintext and ciphertext, both is 64 bits, and the master key is 128 bit. In the key Scheduling part, eight bytes whitening key and 128 bytes subkey is being generated.

The encryption procedure of HIGHT mainly consists of four modules,

- **Key_Sch** The key schedule function Key_Sch has two main components; the first is the computation of whitening keys, which finds eight key whitening bytes, and another is responsible for generating 128 round-key bytes.
- **Initial_Trans** It takes the plaintext P and four whitening keys W_0, W_1, W_2, W_3 and converts the plaintext suitable for the input of the Round_Fun using the following operations.

$T_{0,0} \leftarrow P_0 \boxplus W_0, T_{0,1} \leftarrow P_1, T_{0,2} \leftarrow P_2 \oplus W_1, T_{0,3} \leftarrow P_3,$
$T_{0,4} \leftarrow P_4 \boxplus W_2, X_{0,5} \leftarrow P_5, T_{0,6} \leftarrow P_6 \oplus W_3, T_{0,7} \leftarrow P_7$

- **Round_Fun** Round Function Round_Fun generates $T_i = T_{i,7} \dots T_{i,0}$ into $T_{i+1} = T_{i+1,7} \dots T_{i+1,0}$ described as follows, $T_{i+1,1} \rightarrow T_{i,0}, T_{i+1,3} \rightarrow T_{i,2}, T_{i+1,5} \rightarrow T_{i,4}, T_{i+1,7} \rightarrow T_{i,6}$, $T_{i+1,0} = T_{i,7} \oplus (F_0(T_{i,6}) \boxplus RK_{4i+3})$, $T_{i+1,2} = T_{i,1} \boxplus (F_1(T_{i,0}) \oplus RK_{4i+2})$, $T_{i+1,4} = T_{i,3} \oplus (F_0(T_{i,2} \boxplus RK_{4i+1})$, $T_{i+1,6} = T_{i,5} \boxplus (F_1(T_{i,4}) \oplus RK_{4i})$. Round_Fun consists of two functions F_0 and F_1 as follows,
  $F_0(x) = x^{\lll 1} \oplus x^{\lll 2} \oplus x^{\lll 7}$, $F_1(x) = x^{\lll 3} \oplus x^{\lll 4} \oplus x^{\lll 6}$
- **Final_Trans** Final transformation generates the ciphertexts using the temporary plaintext P and four whitening keys W_4, W_5, W_6, W_7.
  $C_0 \leftarrow T_{32,1} \boxplus W_4, C_1 \leftarrow T_{32,2}, C_2 \leftarrow T_{32,3} \oplus W_5, C_3 \leftarrow T_{32,4}$
  $C_4 \leftarrow T_{32,5} \boxplus W_6, C_5 \leftarrow T_{32,6}, C_6 \leftarrow T_{32,7} \oplus W_7, C_7 \leftarrow X_{32,0}$

Rounds	Input Difference	Output Difference	Probability
5[6]	8201000000000000	009095CA01000000	2^{-12}
5[6]	0000000082010000	01000000009095CA	2^{-12}
6[6]	4282010000000000	009095CA01000000	2^{-17}
6[6]	0000000042820100	01000000009095CA	2^{-17}
8[6]	D000ED86	00848201	2^{-28}
(Mini-HIGHT)	04DC20E2	00848201	2^{-28}
11[6]	118925E2C8010000	4502010000912995	2^{-58}
11[6]	C8010000118925E2)	0091299545020100	2^{-58}
12[9]	00008227213AEA01	00B6F801009002E8	2^{-53}
13[9]	80008AC28A01A0BB	007B2A80009002A7	2^{-61}

Table 1 Classical distinguishers of round reduced HIGHT

In the decryption process of HIGHT, the key schedule function produces the 128 round keys in reverse order. The round function uses $\boxminus$ operation in place of $\boxplus$. Also, the byte-swap operation of the encryption process operates in the opposite direction.

LEA LEA is an ARX-based block cipher proposed by Hong et al. [7]. The block size of LEA is 128 bits. Mainly, LEA consists of three versions, 128-bit key size with 24 rounds, 192 bits of key with 28 rounds, and last one is with 256

Algorithm 1 HIGHT Encryption

Inputs: Plaintext P and Master key M
Outputs: Ciphertext C

```
1: Call Key_Sch
2: Call Initial_Trans
3: for i= 0 to 31 do
4:     Call Round_Fun
5: end for
6: Call Final_Trans
```

bits key size for 32 rounds. A round operation of LEA we can describe as follows, $T_{i+1,0} = RL^{<<9}((T_{i,0} \oplus RK_{i,0}) \boxplus (T_{i,1} \oplus RK_{i,1}))$, $T_{i+1,1} = RR^{<<5}((T_{i,1} \oplus RK_{i,2}) \boxplus (T_{i,2} \oplus RK_{i,3}))$, $T_{i+1,1} = RR^{<<3}((T_{i,2} \oplus RK_{i,4}) \boxplus (T_{i,3} \oplus RK_{i,5}))$, $T_{i+1,3} = T_{i,0}$. Here, $RK_{i,0}, RK_{i,1}, RK_{i,2}, RK_{i,3}, RK_{i,4}, RK_{i,5}$ are the round keys generated by applying key schedule.

SPARX Dinu et al. introduce SPARX [8] at ASIACRYPT'16, which is the first ARX-based family of block ciphers that provide provable bounds on Linear Trails and the maximum length of differential cryptanalysis. The SPARX-n/k family of ciphers includes the SPARX 64/128, SPARX 128/128, and SPARX 128/256, where n indicates the block size bits and k represents the key block size bits. Our paper focuses on SPARX 64/128. The SPARX 64/128 is based on a Feistel network with two state words consisting of eight Feistel steps. Each step consists of three rounds of an ARX-based round function (SPECKEY). The plaintext and ciphertext comprise $w = 2$ words of 32 bit each. The key is four words long of the same size as plaintext/ciphertext.

2.3 Differential cryptanalysis

Differential cryptanalysis[10] is a chosen-plaintext attack that finds a probabilistic relation between the penultimate round plaintext difference and the ciphertext differences by guessing a key.

Definition 1(Differential Cryptanalysis Attack [11]) Let (X, X') be the plaintext pair and after i^{th} round the corresponding ciphertext pair (Y_i, Y_i'). Then the differential probability of an i-round differential $\delta \rightarrow \gamma$ is defined by the conditional probability $P(\Delta Y_i = \gamma | \Delta X = \delta)$, where $\Delta X = X \oplus X'$ and $\Delta Y_i = Y_i \oplus Y_i'$ and the sub-keys $K^1, \ldots, K^i$ are independent and uniformly random. For mounting an attack, the attacker finds the differential probabilities corresponding to each round. So, for an n-round differential $(\delta, \gamma_1, \gamma_2, \ldots, \gamma_n)$,

$$\begin{aligned} P(\Delta Y = \gamma_1, \Delta Y = \gamma_2, \ldots, \Delta Y = \gamma_n | \Delta X = \delta) \\ \approx P(\Delta Y = \gamma_1, \Delta Y = \gamma_2, \ldots, \Delta Y = \gamma_n | \Delta X = \delta, \\ K^{(1)} = k_1, K^{(2)} = k_2, \ldots, K^{(n-1)} = k_{n-1}) \end{aligned}$$

for almost all sub-key values $k_1, k_2, \ldots, k_{n-1}$.

2.3.1 Lower bound complexity analysis of differential cryptanalysis

The favorable outcome of differential cryptanalysis for a cipher with n-rounds totally depends on the propagation of non-zero differentials up to (n-1) rounds with high probability. Using these probabilities, one can compute the lower bound of data complexity for the attack. Considering definition 1 is correct, one can mount differential cryptanalysis attack on a cipher with n rounds,

Table 2 Classical distinguishers of 13-round HIGHT[12]

Rounds	Difference	log_2p
0	80008AC28A01A0BB	0
1	0000C2080128BB80	-6
2	000008E528E98000	-10
3	0000E5A8E9800000	-1
4	0000A82C80000000	-8
5	00002C8000000000	-2
6	0000800000000000	-3
7	0080000000000000	0
8	80000000000000C3	-3
9	000000000072C380	-4
10	0000000C72E98000	-9
11	00A70CF2E9800000	-4
12	A700F22A80000002	-6
13	007B2A80009002A7	-5

block length m and independent subkeys iff the cipher consists of weak round keys and an (n-1) round differential characteristics $\delta \rightarrow \gamma$ is available so that $P(\Delta Y_{n-1} = \gamma | \Delta X = \delta) > 2^{-m}$.

2.3.2 Theorem 1

(Lower bound complexity of differential cryptanalysis attack [11]) Let E_n is the number of encryptions for the differential cryptanalysis attack on a n-round cipher. We can write

$$E_n \geq 2/(P_{max} - 1/(2^m - 1)), \text{ where } P_{max} = \max_{\gamma} \max_{\delta}(\Delta Y_{n-1} = \gamma | \Delta X = \delta)$$

2.4 Differential cryptanalysis and markov cipher

According to Lai et al. [11] an iterated cipher is a markov cipher if the subkeys in the cipher path are independent and each of the (r-1) round non-zero output differences are the part of a markov chain.

Definition 2 (Markov chain [11]) In a cipher the ciphertext differences $\Delta Y_0, \Delta Y_1, \ldots, \Delta Y_n$ generates markov chain. A sequence of discrete random variables $u_0, u_1, \ldots, u_n$ forms a markov chain if

$$P(u_{i+1} = \gamma_{i+1} | u_i = \gamma_i, u_{i-1} = \gamma_{i-1}, \ldots, u_0 = \gamma_0) = P(u_{i+1} = \gamma_{i+1} | u_i = \gamma_i)$$

A Markov chain is called homogeneous if $P(u_{i+1} = \gamma | u_i = \delta)$ is independent of i for all δ and γ and the plaintext X is independent of the subkeys $K_1, K_2, \ldots, K_n$.

Definition 3 (Markov Cipher [11]) Let $Y = f(X, K)$ be a weak round function of an iterated cipher. The cipher is Markov if for every pair (X, X') and

Table 3 Classical distinguishers of 11-round LEA[12]

Rounds	Difference	*log₂p*
0	80000234 α0402214 β0401205 γ0400281	-22
1	80400080 8a000080 82000210 80000234	-14
2	80000014 80400014 80400004 80400080	-9
3	80000000 80000000 80000010 80000014	-3
4	00000000 80000000 80000000 80000000	0
5	00000100 00000000 00000000 00000000	1
6	00020000 00000000 00000000 00000100	-2
7	04000000 00000000 00000020 00020000	-4
8	00000008 00000001 00004004 04000000	-8
9	00001200 28000200 80800800 00000008	-12
10	00200050 05440050 10100101 00001200	-23
11	η800000a 88aaa00a 220202ζ0 00200050	

(Y, Y') we define the differences by a group operation $\otimes$ with $\Delta X = X \otimes X'$ and $\Delta Y = Y \otimes Y'$ in such a way that

$$P(\Delta Y = \gamma | \Delta X = \delta, X = x), \text{ where } \gamma \neq 0 \text{ and } \delta \neq 0$$

is independent of x when subkey K is uniformly random.

Theorem 2 [11] If an n-round iterated cipher is a Markov cipher and the n round keys are independent and uniformly random, then the sequence of differences $\Delta Y_0, \Delta Y_1, \ldots, \Delta Y_n$ is a homogeneous Markov chain. Moreover, this Markov chain is stationary if ΔX is uniformly distributed over the non-neutral elements of the group.

Here our generic deep neural differential classifier tool works under markov assumption for reduced rounds of HIGHT, LEA and SPARX cipher.

2.5 Classical Differential Distinguishers of HIGHT, LEA and SPARX

The designers of HIGHT [6] proposed a reduced round classical differential distinguisher up to 11-rounds. They find two eight-round distinguisher $\alpha \rightarrow \beta$ of Mini-HIGHT, each of which of probability 2^{-28}. An 11-round characteristics $\alpha \rightarrow \beta$ also given with probability 2^{-58}. Jun Yin et al. [9] proposed a MILP-based model for finding differential characteristics of 11-round with probability 2^{-45}, 12-round with probability 2^{-53}, and the 13-round with probability 2^{-61}. For the 13-round differential distinguisher, we mention the input and output differences from round one to round 13 in Table 2, where p means the probability of the differential trail. A summary of all these distinguishers are provided in Table 1.

The best differential distinguisher available for LEA [7] is of 11-rounds with probability at most 2^{-98}, provided by the designers of the cipher. The input difference is (80000234 α0402214 β0401205 γ0400281), where $\alpha \in \{4, c\}, \beta \in \{4, c\}$, and $\gamma = \beta \oplus 1$, and output difference is (η800000a 88aaa00a 220202ζ0

Algorithm 2 DatasetCreationProcess

Inputs: Input differences(δ_o) corresponding to a classical differential distinguisher

Outputs: Training/Validation Dataset DS

```
procedure DatasetCreationProcess($\delta_i$, roundNo, iterations )
    $DS \leftarrow$Empty Set
    for i = 1 to iterations do
        $Key \leftarrow$RandomKey()
        $P_1 \leftarrow$RandomPlaintext()
        if $i \mod 2 = 0$ then
            $P_2 \leftarrow$RandomPlaintext()
            $C_1 \leftarrow$ RANDOM_ORACLE($P_1, Key, roundNo$)    ▷ Encryption engine for the given cipher
            $C_2 \leftarrow$ RANDOM_ORACLE($P_2, Key, roundNo$)
            $DS \leftarrow DS \cup (C_1, C_2, C_1 \oplus C_2, 0)$
        else
            $P_2 = P_1 \oplus \delta_s$
            $C_1 \leftarrow$ RANDOM_ORACLE($P_1, Key, roundNo$)
            $C_2 \leftarrow$ RANDOM_ORACLE($P_2, Key, roundNo$)
            $DS \leftarrow DS \cup (C_1, C_2, C_1 \oplus C_2, 1)$
        end if
    end for
    Return $DS$
end procedure
```

00200050), where $\eta \in \{4, c\}$ and $\zeta \in \{2, 6\}$. The 11-round differential distinguisher, along with each of the input differences and output differences from round one to round-11 is shown in Table 3.

Ralph et al. [13] propose an optimal six-round differential trail of SPARX32/64, where they uses the input difference as (00000000 02110A04), and the six-round output difference is (AF1ABF30 850A9520). The probability of the trial is 2^{-13}. A nine-round trail is also proposed with probability $2^{-32.87}$, where the input difference is (28000010, 28000010) and output difference is (80818283, 80008002). Ralph et al. [13] presents all the input and output differences for optimal differential trials up to ten rounds.

3 Modeling Differential Cryptanalysis using Deep Learning

Aron Gohr [1] first proposes the concept of the deep neural distinguisher, corresponding to a classical differential distinguisher for cipher SPECK and SIMON [2]. Gohr chooses an input plaintext difference δ_i and two plaintexts P_1 and P_2 such that $P_1 \oplus P_2 = \delta_i$. We call (P_1, P_2) as real pairs. Here the main purpose is to classify differently the real pairs with a random plaintext pair

Algorithm 3 TrainingProcess

```
Inputs: Training Data $DS_{Train}$
Outputs: Training accuracy $ACR_{Train}$
1: procedure TRAININGPROCESS($DS_{Train}$)
2:     Create the ML model $ML_{\delta_i}$.
3:     Train $ML_{\delta_i}$ with with train data $DS_{Train}$ and let $ACR_{Train}$ be the train accuracy.
4:     if $ACR_{Train} > 0.5$ then
5:         Create new dataset
6:         Call $ValidationProcess(ML_{\delta_i}, DS_{Valid})$
7:     else
8:         Return "No distinguisher found"
9:     end if
10:    Return $ACR_{Train}$
11: end procedure
```

Algorithm 4 ValidationProcess

```
Inputs: New Validation Data $DS_{Valid}$ and trained model $MLD$
Outputs: Validation Accuracy $ACR_{Valid}$
1: procedure VALIDATIONPROCESS($ML_{\delta_i}, DS_{Valid}$)
2:     Load model $ML_{\delta_i}$.
3:     Validate $ML_{\delta_i}$ with with validation data $DS_{Valid}$ and let $ACR_{valid}$ be the validation accuracy.
4:     if $ACR_{Valid} > 0.5$ then
5:         Distinguisher found for the corresponding cipher
6:     else
7:         Return "No distinguisher found"
8:     end if
9:     Return $ACR_{Valid}$
10: end procedure
```

(P'_1, P'_2) such that $P'_1 \oplus P'_2 = \delta_r$, where δ_r is a random difference. Applying this approach, he trained a DNN, which performs well in classifying the real and random ciphertext pairs. He replicates a new difference distribution table (DDT) corresponding to the DDT of the classical differential distinguisher during the training phase of the neural classifier and uses the new DDT to validate data. Gohr also gives a detailed description, comparing the performance of the classical differential distinguisher with the corresponding neural differential distinguisher. He proves neural distinguishers work more efficiently. Here, we propose a method that finds a generic deep neural distinguisher. Algorithm 2 generates the training and validation dataset. It takes input a plaintext difference δ_i, the number of rounds for the cipher *roundNo*, and the number of rows in the dataset *iterations*. Mainly, We run the encryption function of a cipher up to *roundNo* rounds and *terations* times and store the ciphertexts. For each iteration, we use a new random key. Assume P_1 is a random plaintext. If the current iteration number is divisible by 2 then we take another random plaintext P_2 and encrypt (P_1, P_2) to get a ciphertext pair (C_1, C_2). Append $(C_1, C_2, C_1 \oplus C_2, 0)$ to dataset DS, where 0 is the label for two random ciphertext pair (C_1, C_2). But if the iteration number is not divisible by zero then calculate $P_2 = P_1 \oplus \delta_i$. Use encryption oracle to encrypt (P_1, P_2) and get ciphertext pair (C_1, C_2). In this case, add $(C_1, C_2, C_1 \oplus C_2, 1)$ to dataset DS, where 1 is the label for two known ciphertext pair (C_1, C_2), generated by using the plaintext difference δ_i. Finally, we return the merged dataset DS containing 50% known ciphertexts, and the rest are random.

We use Algorithmm 3 for creating a new ML model ML_{δ_i} and train the model with dataset DS_{Train}. If we achieve greater than 50% training accuracy, then call Algorithm 4 for validation of the dataset DS_{Valid}. If validation accuracy ACR_{Valid} is more than 50%, then we can claim a valid distinguisher found for the cipher.

3.1 Generic differential distinguisher

Here we automate the searching of neural distinguishers of n rounds, where $n \leq r$, r is the maximum round number of a cipher. We propose a generic neural differential classifier that automatically finds a deep neural distinguisher for any cipher for a given round. Take one classical differential distinguisher with differential characteristics $\delta_i \rightarrow \delta_o$ of m rounds with probability 2^{-p}, where $(m+n) \leq r$. Now apply δ_i to a neural distinguisher of one round and check the accuracy. In case of good validation accuracy, take the ciphertext differences as δ_i for the next iteration with one round increase. For each iteration, update and store the round number and the maximum accuracy. Continue this way until a round is found with accuracy less than 50%. We described the procedure in Algorithm 5. Consider CD_{Old} is the set of ciphertext differences from an earlier distinguisher, and PD_{New} is a set of plaintext differences. In our algorithm we take input either CD_{Old} or PD_{New}. We check if PD_{New} is empty else initialize it by CD_{Old}. For a neural distinguisher, assume the number of rounds is *roundNo*, which is initialized to a given number N with N less than or equal to the maximum number of rounds for the given cipher. Now choose any difference δ_i from the set PD_{New} and assign it as a real plaintext difference for the current neural distinguisher to be constructed. Next create a new ML distinguisher MLD. Generate the training data-set by calling the method *DatasetCreationProcess* with providing inputs δ_i, *roundNo* and *iteration*. Save the training dataset in DS_{Train}. Apply DS_{Train} to train MLD, which output accuracy ACR_{Train}. Calling the procedure *DatasetCreationProcess* generate the validation dataset DS_{Valid}. Run the method *ValidationProcess* by applying the DS_{Valid} on the trained model MLD and save the accuracy in ACR_{Valid}. For each δ_i from PD_{New}, execute the dataset creation, training, and validation process in a loop. In each iteration collect each of the ACR_{Valid} in a set ACR_{Set}. Now find the maximum accuracy ACR_{MAX} from ACR_{Set} and

Algorithm 5 GenericDifferentialNeuralClassifier

Inputs: CD_{Old} are the set of ciphertext differences from last model or PD_{New} are the set of plaintext differences if available.

Output: Maximum accuracy ACR_{Max} and best difference MAX_{δ_i} upto r rounds.

```
1: procedure GenericDifferentialNeuralClassifier($PD_{New}/CD_{Old}$, N)
2:     if $PD_{New}$ is empty then
3:         $PD_{New} \leftarrow CD_{Old}$
4:     end if
5:     roundNo ← N
6:     accSet ← $empty$
7:     for each $\delta_i$ in $PD_{New}$ do
8:         Set the real plaintext difference as $\delta_i$.
9:         iteration ← 100000
10:        Create a new ML model, let $MLD$.
11:        $DS_{Train}$ ← DatasetCreationProcess($\delta_i$, roundNo, iterations)
12:        $ACR_{Train}$ ← TrainingProcess($MLD, DS_{Train}$)
13:        $DS_{Valid}$ ← DatasetCreationProcess($\delta_i$, roundNo, iterations)
14:        $ACR_{Valid}$ ← ValidationProcess($MLD, DS_{Valid}$)
15:        $ACR_{Set} \leftarrow ACR_{Set} \cup ACR_{Valid}$
16:    end for
17:    $ACR_{Max}$ ← MAX($ACR_{Set}$)
18:    $PD_{Best} \leftarrow GetPD(ACR_{Max})$
19:    $CD_{Best}$ ← GetCD($PD_{new}, ACR_{Max}$)    ▷ Returns the plaintext difference corresponding to an accuracy
20:    $MLD_{PD_{Best}} \leftarrow MLD$
21:    truePositiveCDSet ← GetCDFromTruePosSet($ACR_{Max}$)
22:    $PD_{New}$ ← GetCDSetMinHammingWeight(truePositiveCDSet)    ▷ Returns the minimum hamming weights
23:    roundNo ← roundNo + $\mathcal{R}$
24:    repeat steps 6 to 22.
25:    Return ($MLD_{PD_{Best}}, ACC_{Max}, PD_{Best}, roundNo, iteration$)
26: end procedure
```

the best plaintext difference CD_{Best} corresponding to the ACR_{Max}. Save MLD to $MLD_{CD_{Best}}$. We also need the new ciphertext differences corresponding to the best accuracy. With taking input as ACR_{Max} apply method *GetCDFromTruePosSet* to get the plaintext differences and store these to *truePositiveCDSet*. We choose only those new ciphertext differences which have a minimum hamming difference. The reason is, the possibility of generating active bits after applying a cipher is less, with ciphertext differences having less number of active bits. Update the plaintext difference set PD_{new} with the new ciphertext differences, which is the return value of the method *GetCDSetMinHammingWeight*. Now increase the round number by $\mathcal{R}, 0 \leq \mathcal{R} \leq r$

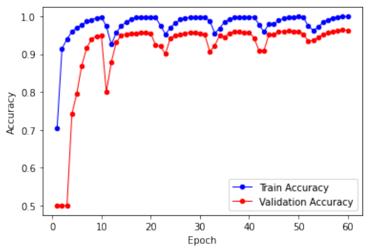

Fig. 1 HIGHT Validation Acc Vs Train Accuracy (CNN)

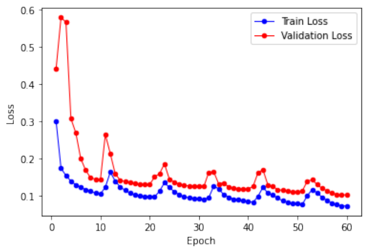

Fig. 2 HIGHT Validation loss Vs Train Loss (CNN)

and iteration numbers(if required) and rerun the above procedure using the new set of plaintext differences PD_{new}. Repeat the above process until we get an accuracy value less than 50% from the methods *TrainingProcess* or *ValidationProcess*. Finally return the best ML distinguisher $MLD_{PD_{Best}}$ with $MAX_{acc}, PD_{Max}, roundNo$ and *iteration*.

3.2 Estimation of lower bound of data complexity

Choose one classical differential distinguisher with differential characteristics $\delta_i \to \delta_o$ of m rounds with probability 2^{-p}, where $m \leq r$. Following Theorem 1, the lower bound of data complexity of the classical distinguisher is bounded by $2/(2^{-p} - 1/2^m - 1)$. Next, form a deep neural distinguisher by taking δ_o as the input plaintext difference. Suppose the neural distinguisher gives a high accuracy up to n rounds. Then we conclude the lower bound of the data complexity for the corresponding distinguisher with $(m + n)$ rounds is approximately 2^p. The new lower bound of data complexity estimation for HIGHT, LEA and SPARX is provided in Table 6.

4 Experimental Results

In this section, we describe the outcome of the generic neural distinguisher after application on HIGHT, LEA and SPARX. We use google colab with installed Keras-GPU for all our ML-related experiments, including data generation. We run three different models, convolutional neural network(CNN), Light Gradient Boosting Machine (LGBM), and Long short-term memory (LSTM) for training and validation of datasets. In general, we use total 10^5 data samples of which 50% is applied for training and rest of those for validation. For CNN and LSTM, we varied number of layers, number of neurons per layer and number of blocks per layer. Also we applied different types of activation functions.

4.1 HIGHT

In [6] $\delta_i = 0x0800000000000000$ is used as a classical distinguisher for the output difference of sixth round. For the autometic generic neural distinguisher, we take the input difference of a classical distinguisher with $\delta_i = 0x0800000000000000$ as the input plaintext difference for seventh round. Here,

Table 4 Accuracy, true positive rate (TPR) and true negative rate (TNR) for HIGHT

Model	Round	Train Accuracy	Validation Accuracy	TPR	TNR
CNN	5	99.99	99.98	1.0	0.9998
	6	100.00	99.99	1.0	0.999
	7	98.82	97.023	0.995	0.9449
	8	98.98	98.71	0.998	0.9753
	9	67.892	50.48	0.5052	0.4942
	10	75.90	50.30	0.5221	0.4653
LGBM	5	100.00	99.94	0.9999	0.9988
	6	100.00	99.99	1.0	0.9997
	7	80.05	78.89	0.985	0.5944
	8	99.19	98.56	0.9893	0.981
	9	65.31	50.26	0.5656	0.4394
	10	66.17	50.01	0.5019	0.5054
LSTM	5	94.76	94.814	0.97343	0.92261
	6	82.61	82.16	0.93443	0.7098
	7	60.28	60.09	0.5663	0.635
	8	66.91	66.36	0.6434	0.684

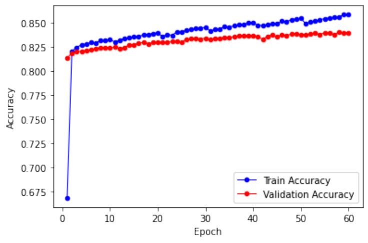

Fig. 3 HIGHT Validation Acc Vs Train Accuracy (LSTM)

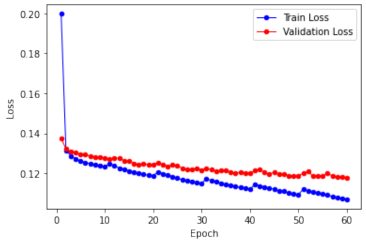

Fig. 4 HIGHT Validation loss Vs Train Loss (LSTM)

we construct a six-round neural distinguisher which gives a high accuracy, and this one works as a 13-round distinguisher. Here the total data complexity of this distinguisher is at least 2^{30}.

The LGBM and LSTM model can classify the corresponding real and random ciphertext differences up to 11 rounds, whereas the CNN model performs better and allows up to 14 rounds. For all the three models we describe the result using training accuracy, validation accuracy, true positive rate, and true negative rate. Table 4 depicts the performance of the CNN and LGBM model. Figure 1,and 3 explains the relation between number of epochs and training/validation accuracy for CNN and LSTM model. The relation between training/validation loss with increasing epochs is depicted in Figure 2, and 4 for CNN and LSTM model.

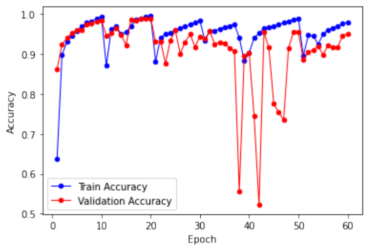

Fig. 5 LEA Validation Acc Vs Train Accuracy (CNN)

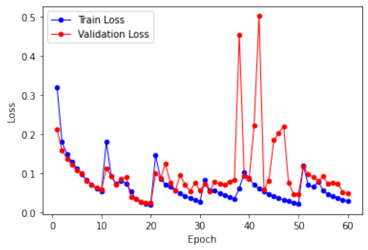

Fig. 6 LEA Validation loss Vs Train Loss (LSTM)

Table 5 Accuracy, true positive rate (TPR) and true negative rate (TNR) for LEA cipher

Model	Round	Train Accuracy	Validation Accuracy	TPR	TNR
CNN	5	99.87	98.45	0.990	0.978
	6	99.39	95.04	0.956	0.944
	7	96.08	96.08	0.886	0.812
	8	75.60	51.37	0.536	0.491
	9	60.58	50.45	0.500	0.505
LGBM	6	92.54	91.83	0.897	0.939
	7	87.84	87.62	0.845	0.906
	8	66.19	62.51	0.499	0.749
	9	65.05	50.18	0.499	0.508
LSTM	5	96.36	96.23	0.960	0.964
	6	84.77	84.53	0.846	0.844
	7	52.46	52.31	0.649	0.396
	8	60.05	59.81	0.570	0.625

Table 6 Reduced Lower bound of Data Complexity

Cipher	Rounds	Best known Data Complexity	Lower Bound of Data Complexity (Our Approach)
HIGHT [12]	13	2^{61}	2^{30}
LEA [7]	11	2^{98}	2^{48}
SPARX [13]	9	2^{37}	2^{25}

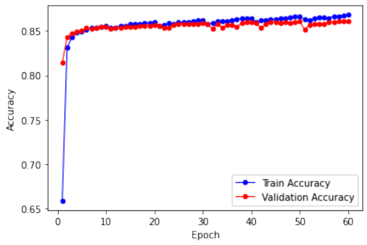

Fig. 7 LEA Validation Acc Vs Train Accuracy (LSTM)

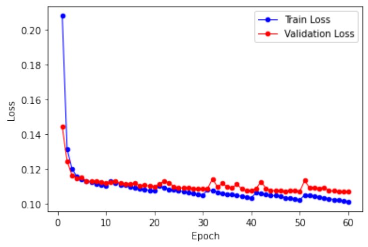

Fig. 8 LEA Validation loss Vs Train Loss (LSTM)

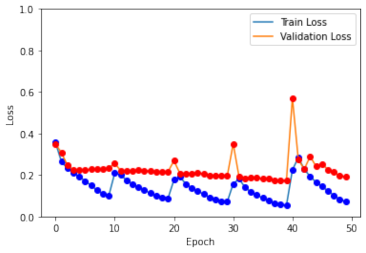

Fig. 9 Training/Validation Loss vs Epoch for SPARX(CNN)

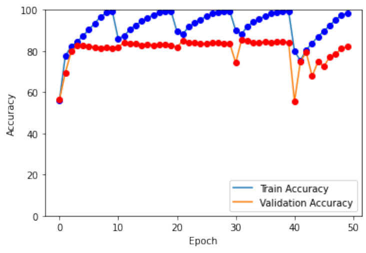

Fig. 10 Training/Validation Accuracy vs Epoch for SPARX(CNN)

4.2 LEA

For LEA, we use $\delta_i = (0x00000000, 0x80000000, 0x80000000, 0x80000000)$ as the plaintext difference given by Hong et al. [7] at Table 10. We take the fourth-round output difference from the eleven-round differential characteristics and apply our generic distinguisher described Algorithm 5. The CNN and LGBM model provides acceptable results for up to 13 rounds. The LSTM model can classify the cipher upto 12 rounds. We summarize the results from the CNN, LSTM and LGBM model in Table 5. Figure 5, and 7 describes the relation between training accuracy and the number of epochs for CNN and LSTM model. We get high accuracy up to 80% after applying the five round output $(0x00000100, 0x00000000, 0x00000000, 0x00000000)$ from Table 3 to generic neural distinguisher as δ_i. The variation of training and validation loss by increasing epoch number is depicted in Figure 6, and 8 for CNN and LSTM model. In this case also our generic distinguisher can classify LEA upto 13 rounds. Here we can redefine the lower bound of data complexity of 11-round LEA to 2^{49}.

4.3 SPARX

Ralph et al. [13] found the five round output difference $\delta_i = 0x00400040/0x00000000$ to describe the six round differential trail. We use the same difference $\delta_i = 0x00400040/0x00000000$ as input difference and found a neural distinguisher up to five rounds with good accuracy. Adding up, we found a neural distinguisher for SPARX32/64 up to 11 rounds. Here we use three neural models. The training and validation accuracy of the CNN, LGBM and LSTM models with true positive and negative rates is shown in Table 7. The models also provide good accuracy for input difference $\delta_i = 0x00408000/0x00000000$ and $\delta_i = 0x28000010, 0x28000010$ up to six rounds.

Figure 10 and 12 describes the relation between training accuracy and the number of epochs for the CNN and LSTM model. The variation of training and validation loss by increasing epoch number is depicted in Figure 9, and 11 for the CNN and LSTM model. We found the lower bound of data complexity of 9-round as 2^{25}.

Table 7 Accuracy, true positive rate (TPR) and true negative rate (TNR) for SPARX

Model	Round No	Training Accuracy	Validation Accuracy	TPR	TNR
CNN	3	99.41	93.27	0.918	0.948
	4	97.43	76.61	0.751	0.781
	5	66.41	50.38	0.478	0.528
LGBM	3	90.32	89.90	0.866	0.932
	4	69.00	67.00	0.616	0.728
	5	65.54	50.64	0.520	0.493
LSTM	3	84.17	83.57	0.782	0.889
	4	63.35	62.93	0.681	0.578
	5	50.94	50.45	0.289	0.717

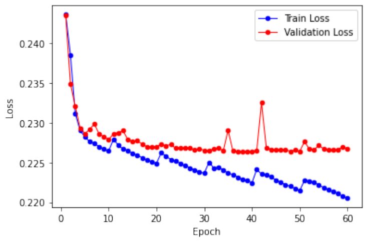

Fig. 11 Training/Validation Loss vs Epoch for SPARX (LSTM)

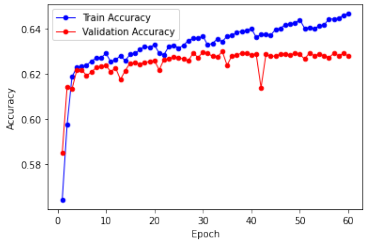

Fig. 12 Training/Validation Accuracy vs Epoch for SPARX(LSTM)

5 Conclusion

In this paper, we introduce a novel technique of finding neural differential distinguisher. Using the tool we report neural classifier for the cipher HIGHT up to 14-rounds, LEA up to 13-rounds, and up to nine rounds for SPARX, which are the first neural distinguisher for the cipher. A general approach for finding the lower bound of data complexity is also provided. We want to cover more ciphers applying our tool for finding new differential distinguishers.

References

[1] Gohr, A.: Improving attacks on round-reduced speck32/64 using deep learning. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II. Lecture Notes in Computer Science, vol. 11693, pp. 150–179. Springer, ??? (2019). https://doi.org/10.1007/978-3-030-26951-7_6. https://doi.org/10.1007/978-3-030-26951-7_6

[2] Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK families of lightweight block ciphers. IACR Cryptol. ePrint Arch., 404 (2013)

[3] Baksi, A., Breier, J., Chen, Y., Dong, X.: Machine learning assisted differential distinguishers for lightweight ciphers. In: Design, Automation & Test in Europe Conference & Exhibition, DATE 2021, Grenoble, France, February 1-5, 2021, pp. 176–181. IEEE, ??? (2021). https://doi.org/10.23919/DATE51398.2021.9474092. https://doi.org/10.23919/DATE51398.2021.9474092

[4] Yadav, T., Kumar, M.: Differential-ml distinguisher: Machine learning based generic extension for differential cryptanalysis. In: Longa, P., Ràfols, C. (eds.) Progress in Cryptology - LATINCRYPT 2021 - 7th International Conference on Cryptology and Information Security in Latin America, Bogotá, Colombia, October 6-8, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12912, pp. 191–212. Springer, ??? (2021). https://doi.org/10.1007/978-3-030-88238-9_10. https://doi.org/10.1007/978-3-030-88238-9_10

[5] Banik, S., Pandey, S.K., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: GIFT: A small present - towards reaching the limit of lightweight encryption. In: Fischer, W., Homma, N. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10529, pp. 321–345. Springer, ??? (2017). https://doi.org/10.1007/978-3-319-66787-4_16. https://doi.org/10.1007/978-3-319-66787-4_16

[6] Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B., Lee, C., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J., Chee, S.: HIGHT: A new block cipher suitable for low-resource device. In: Goubin, L., Matsui, M. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4249, pp. 46–59. Springer, ??? (2006). https://doi.org/10.1007/11894063_4. https://doi.org/10.1007/11894063_4

[7] Hong, D., Lee, J., Kim, D., Kwon, D., Ryu, K.H., Lee, D.: LEA: A 128-bit block cipher for fast encryption on common processors. In: Kim, Y., Lee, H., Perrig, A. (eds.) Information Security Applications - 14th International Workshop, WISA 2013, Jeju Island, Korea, August 19-21, 2013, Revised Selected Papers. Lecture Notes in Computer Science, vol. 8267, pp. 3–27. Springer, ??? (2013). https://doi.org/10.1007/978-3-319-05149-9_1. https://doi.org/10.1007/978-3-319-05149-9_1

[8] Dinu, D., Perrin, L., Udovenko, A., Velichkov, V., Großschädl, J., Biryukov, A.: Design strategies for ARX with provable bounds: Sparx and LAX. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory

and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10031, pp. 484–513 (2016). https://doi.org/10.1007/978-3-662-53887-6_18. https://doi.org/10.1007/978-3-662-53887-6_18

[9] Bagherzadeh, E., Ahmadian, Z.: Milp-based automatic differential search for LEA and HIGHT block ciphers. IET Inf. Secur. **14**(5), 595–603 (2020). https://doi.org/10.1049/iet-ifs.2018.5539

[10] Biham, E., Shamir, A.: Differential Cryptanalysis of the Data Encryption Standard. Springer, ??? (1993). https://doi.org/10.1007/978-1-4613-9314-6. https://doi.org/10.1007/978-1-4613-9314-6

[11] Lai, X., Massey, J.L., Murphy, S.: Markov ciphers and differential cryptanalysis. In: Davies, D.W. (ed.) Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings. Lecture Notes in Computer Science, vol. 547, pp. 17–38. Springer, ??? (1991). https://doi.org/10.1007/3-540-46416-6_2. https://doi.org/10.1007/3-540-46416-6_2

[12] Yin, J., Ma, C., Lyu, L., Song, J., Zeng, G., Ma, C., Wei, F.: Improved cryptanalysis of an iso standard lightweight block cipher with refined milp modelling. In: Chen, X., Lin, D., Yung, M. (eds.) Information Security and Cryptology, pp. 404–426. Springer, Cham (2018)

[13] Ankele, R., List, E.: Differential cryptanalysis of round-reduced sparx-64/128. In: Preneel, B., Vercauteren, F. (eds.) Applied Cryptography and Network Security - 16th International Conference, ACNS 2018, Leuven, Belgium, July 2-4, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10892, pp. 459–475. Springer, ??? (2018). https://doi.org/10.1007/978-3-319-93387-0_24. https://doi.org/10.1007/978-3-319-93387-0_24