A Deep Neural Differential Distinguisher for ARX based Block Cipher

Debranjan Pal, Upasana Mandal, Mainak Chaudhury, Abhijit Das, and Dipanwita Roy Chowdhury

Crypto Research Lab, IIT Kharagpur, India
debranjanpal@iitkgp.ac.in, mandal.up98@gmail.com, mainakcacsgu@gmail.com, abhij@cse.iitkgp.ac.in, drc@cse.iitkgp.ac.in

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 called NDDT[1], using deep neural classifier that assists to find differential distinguishers for symmetric block ciphers with reduced round. We apply this approach for the differential cryptanalysis of ARX-based encryption schemes HIGHT, LEA, SPARX and SAND. To the best of our knowledge, this is the first deep learning-based distinguisher for the mentioned ciphers. The result shows that our deep learning based distinguishers work with high accuracy for 14-round HIGHT, 13-Round LEA, 11-round SPARX and 14-round SAND128. The relationship between the hamming weight of input difference of a neural distinguisher and the corresponding maximum round number of the cipher has been justified through exhaustive experimentation. The lower bounds of data complexity for differential cryptanalysis have also been improved.

Keywords: HIGHT · LEA · SPARX · SAND · 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 recognition, 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 [9], 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

[1] Neural based Differential Distinfuisher searching Tool

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 show 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 [5]. In DATE 2021, A Baksi et al. [3] introduce 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[16] apply the technique to find neural based classifier for 12-round SPECK-32 [5], eight-round GIFT-64 [4] and 12-round SIMON-32 [5].

1.1 Our Contribution

In this paper we introduce a generic deep learning based automated differential distinguisher. Using the tool [2] we analyze the differential behaviour of four ARX-based cipher, HIGHT [12], LEA [11], SPARX [8] and SAND [7]. We found a 14-round neural differential distinguisher for HIGHT, an 13-round distinguisher for LEA, a nine-round distinguisher for SPARX and a 14-round distinguisher for SAND128. An experiment also performed to show the relationship between the hamming weight of input difference of a neural distinguisher and the corresponding maximum round number of the cipher. We achieve a new lower bound of data complexity for reduced rounds of HIGHT, LEA, SPARX and SAND.

1.2 Organization of the paper

The rest of the paper is organized as follows. Section 2 describes the basic notations we use in our paper and a brief explanation of differential cryptanalysis and markov cipher. In Section 3, we present our generic tool for constructing the deep neural classifier. Section 4 presents the model architectures, implementation details and the experimental results. In Section 5, we conclude the paper.

2 Preliminaries

This section mainly describes the brief specification of the four ciphers and the literature study of classical differential cryptanalysis for them.

[2] Code and data are available on request.

2.1 Basic Notations

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

- P, C and T is the plaintext, ciphertext and temporary state value.
- M, W and RK represents the master key, whitening key and the round key.
- δ_i and δ_o represents input and output difference for a differential trail.
- $\mathcal{CD}$ 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 operation.
- $A^{\ll s}$ and $A^{\gg s}$ means s-bit left and right rotation of a word A.
- $A \lll_{n/4} s$ denotes dividing A into four $n/4$ bit words (A_0, A_1, A_2, A_3) and shift each individual word to s bits left.

2.2 Differential cryptanalysis

Differential cryptanalysis [6] 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 [14]) 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 $P(\Delta Y_i = \gamma \mid \Delta X = \delta)$, 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 $(\delta, \gamma_1, \gamma_2, \ldots, \gamma_n)$,

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

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

2.3 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, 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 $P(\Delta Y_{n-1} = \gamma \mid \Delta X = \delta) > 2^{-m}$.

Theorem 1 (Lower bound complexity of differential cryptanalysis attack [14])

Let E_n is the number of encryptions for the differential cryptanalysis attack on a n-round cipher. We can write

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

2.4 Differential cryptanalysis and markov cipher

According to Lai et al. [14] 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 [14]) 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

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

A Markov chain is called homogeneous if $P(u_{i+1} = \gamma \mid u_i = \delta)$ 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 [14]) 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 (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

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

is independent of x when subkey K is uniformly random.

Theorem 2 [14] 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, SPARX and SAND64/128 cipher.

3 Modeling Differential Cryptanalysis using Deep Learning

Aron Gohr [9] first proposes the concept of the deep neural distinguisher, corresponding to a classical differential distinguisher for cipher SPECK and SIMON [5]. Gohr chooses an input plaintext difference δ_i and two plaintexts P_1

Algorithm 1 DatasetCreationProcess

Inputs: Input differences(δ_i) corresponding to a classical differential distinguisher
Outputs: Training/Validation Dataset DS

procedure DatasetCreationProcess(δ_i, R_n, iterations)
  $DS \leftarrow$Empty Set
  for i = 1 to iterations **do**
    $Key \leftarrow$RandomKey() ▷ Returns a random key
    $P_1 \leftarrow$RandomPlaintext()
    if $i \mod 2 = 0$ **then**
      $P_2 \leftarrow$RandomPlaintext()
      $C_1 \leftarrow$ RandomOracle(P_1, Key, R_n) ▷ Encryption engine for the given cipher
      $C_2 \leftarrow$ RandomOracle(P_2, Key, R_n)
      $DS \leftarrow DS \cup (C_1, C_2, C_1 \oplus C_2, 0)$
    else
      $P_2 = P_1 \oplus \delta_i$
      $C_1 \leftarrow$ RandomOracle(P_1, Key, R_n)
      $C_2 \leftarrow$ RandomOracle(P_2, Key, R_n)
      $DS \leftarrow DS \cup (C_1, C_2, C_1 \oplus C_2, 1)$ ▷ Assume DS_{Train} be the training dataset and DS_{Valid} is the validation dataset
    end if
  end for
  Return DS
end procedure

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 (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 Deep Neural Network (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.

Our Approach

We propose a method that finds a generic deep neural distinguisher. Algorithm 1 generates the training and validation dataset. It takes input a plaintext difference δ_i, the number of rounds for the cipher R_n, and the number of rows in the dataset ITR. We run the encryption function of a cipher up to R_n rounds and ITR 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 the

Algorithm 2 TrainingProcess

Inputs: Training Data DS_{Train}
Outputs: Training accuracy ACR_{Train}

procedure TrainingProcess(DS_{Train}, DS_{Valid})
    Create the ML model ML_{δ_i}.
    Train ML_{δ_i} with with train data DS_{Train} and let ACR_{Train} be the train accuracy.
    if $ACR_{Train} > 0.5$ **then**
        Create new dataset
        Call $ValidationProcess(ML_{\delta_i}, DS_{Valid})$
    else
        Return "No distinguisher found"
    end if
    Return ACR_{Train}
end procedure

Algorithm 3 ValidationProcess

Inputs: New Validation Data DS_{Valid} and trained model MLD
Outputs: Validation Accuracy ACR_{Valid}

procedure ValidationProcess($ML_{\delta_i}, DS_{Valid}$)
    Load model ML_{δ_i}.
    Validate ML_{δ_i} with with validation data DS_{Valid} and let ACR_{valid} be the validation accuracy.
    if $ACR_{Valid} > 0.5$ **then**
        Distinguisher found for the corresponding cipher
    else
        Return "No distinguisher found"
    end if
    Return ACR_{Valid}
end procedure

random ciphertext pair (C_1, C_2). But if the iteration number is not divisible by two 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 the 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 Algorithm 2 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 3 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.

Algorithm 4 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} up to R_n rounds.

procedure GenericDifferentialNeuralClassifier(PD_{New}/CD_{Old}, N)
  if PD_{New} is empty **then**
    $PD_{New} \leftarrow CD_{Old}$
  end if
  $R_n \leftarrow$ N
  accSet $\leftarrow$ *empty*
  for each δ_i in PD_{New} **do**
    Set the real plaintext difference as δ_i.
    iteration $\leftarrow$ 100000
    Create a new ML model, let MLD.
    $DS_{Train} \leftarrow$ DatasetCreationProcess(δ_i, R_n, iterations)
    $ACR_{Train} \leftarrow$ TrainingProcess(MLD, DS_{Train})
    $DS_{Valid} \leftarrow$ DatasetCreationProcess(δ_i, R_n, iterations)
    $ACR_{Valid} \leftarrow$ ValidationProcess(MLD, DS_{Valid})
    $ACR_{Set} \leftarrow ACR_{Set} \cup ACR_{Valid}$
  end for
  $ACR_{Max} \leftarrow$ MAX(ACR_{Set})
  $PD_{Best} \leftarrow GetPD(ACR_{Max})$
  $CD_{Best} \leftarrow$ GetCD(PD_{new}, ACR_{Max}) ▷ Returns the plaintext difference corresponding to an accuracy
  $MLD_{PD_{Best}} \leftarrow MLD$
  truePositiveCDSet $\leftarrow$ GetCDFromTruePosSet(ACR_{Max})
  $PD_{New} \leftarrow$ GetCDSetMinHammingWeight(truePositiveCDSet) ▷ Returns the minimum hamming weights
  $R_n \leftarrow R_n + \mathcal{R}$
  repeat steps 7 to 23.
  Return ($MLD_{PD_{Best}}, ACC_{Max}, PD_{Best}$,
R_n, ITR)
end procedure

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. We choose only those new ciphertext differences which have a minimum hamming difference. The reason is that the possibility of generating active bits after applying a cipher is less, with

Table 1: Hamming weight Vs validation accuracy for HIGHT cipher

Plaintext Difference	HW	Rnd. Vs Accuracy			
		Rnd. (LB)	Acc. (Max)	Rnd. (UB)	Acc. (Min)
8201000000000000	3	5	98.31	9	50.31
0000000082010000	3	6	95.79	9	50.18
4282010000000000	5	3	99.81	5	80.28
0000000042820100	5	3	99.91	4	50.31
118925E2C8010000	16	2	96.92	3	56.25
C8010000118925E2	16	2	96.70	3	56.84
00008227213AEA01	18	3	94.51	6	50.01
80008AC28A01A0BB	19	1	99.55	3	52.61
0000C2080128BB80	14	2	89.32	4	50.36
000008E528E98000	14	2	98.25	4	51.87
0000E5A8E9800000	14	4	95.65	5	51.38
0000A82C80000000	7	4	96.11	5	56.96
00002C8000000000	4	5	99.90	6	50.12
0000800000000000	1	7	99.26	8	50.09
0080000000000000	1	8	99.89	9	50.02
80000000000000C3	5	4	99.88	8	50.01
000000000072C380	9	4	98.42	6	50.89
0000000C72E98000	12	3	97.50	5	50.36
00A70CF2E9800000	18	2	99.90	4	50.00
A700F22A80000002	15	2	92.52	3	54.56
007B2A80009002A7	16	1	99.99	4	50.11

ciphertext differences having less number of active bits. Continue this way until a round is found with accuracy less than 50%.

We described the procedure in Algorithm 4. Consider CD_{Old} is the set of ciphertext differences from an earlier distinguisher, and PD_{New} is a set of plaintext differences. In the algorithm take as input either CD_{Old} or PD_{New}. Check if PD_{New} is empty, else initialize it by CD_{Old}. For a neural distinguisher, assume the number of rounds is R_n, 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, R_n and *iteration*. Save the training dataset in DS_{Train}. Apply DS_{Train} to train MLD, which output accuracy ACR_{Train}. By 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 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.

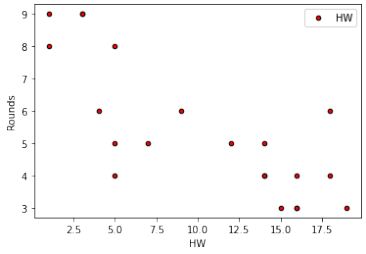

Fig. 1: Relation between hamming weight (HW) and rounds for maximum validation accuracy (round_accuracy) for HIGHT

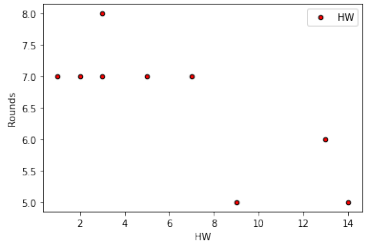

Fig. 2: Relation between hamming weight (HW) and rounds for maximum validation accuracy (round_accuracy) for LEA

With taking input as ACR_{Max} apply method *GetCDFromTruePosSet* to get the plaintext differences and store these to *truePositiveCDSet*. Store only those new ciphertext differences which have a minimum hamming difference and for asing in the next iteration. Update the plaintext difference set PD_{new} with the new ciphertext differences, which is the return value of the method *GetCDSet-MinHammingWeight*. Now increase the round number by $\mathcal{R}, 0 \leq \mathcal{R} \leq r$. Also modify the iteration number ITR(if required). Next, 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}, R_n and ITR.

Table 2: Hamming weight Vs validation accuracy for LEA cipher

Plaintext Difference	HW	Rounds Vs Accuracy			
		Round (LB)	Accuracy (Max)	Round (UB)	Accuracy (Min)
80000014 80400014 80400004 80400080	13	3	94.12	6	50.15
80000000 80000000 80000010 80000014	7	3	99.20	7	53.84
00000000 80000000 80000000 80000000	3	5	99.22	8	61.75
00000100 00000000 00000000 00000000	1	5	91.89	7	62.24
00020000 00000000 00000000 00000100	2	4	94.80	7	50.70
04000000 00000000 00000020 00020000	3	4	92.43	7	50.07
00000008 00000001 00004004 04000000	5	3	95.41	7	50.05
00001200 28000200 80800800 00000008	9	2	98.85	5	50.27
00200050 05440050 10100101 00001200	14	1	99.69	5	50.20

3.2 Relation between accuracy and hamming weight

We perform an experiment using our generic distinguisher by varying the plaintext difference (thus hamming weight) and monitor the upper and lower range of

round number to calculate the minimum and maximum validation accuracy. A relation we can obtain among hamming weight(HW), number of rounds and the validation accuracy. Using plaintext difference with lower hamming weight results neural distinguisher of more rounds. Hence decreasing the hamming weight increases the maximum round with acceptable validation accuracy for a distinguisher and the reverse is also true. Figure 1 and 2 describes the relationship between the hamming weight and number of rounds for the cipher for maximum validation accuracy for HIGHT and LEA. Table 1 and Table 2 describes the relation between hamming weights and validation accuracy with different plaintext/ciphertext difference for HIGHT and LEA.

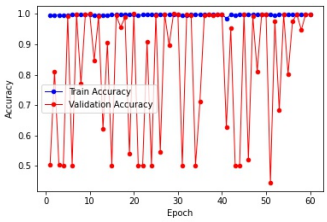

Fig. 3: HIGHT Validation Acc Vs Train Accuracy (CNN)

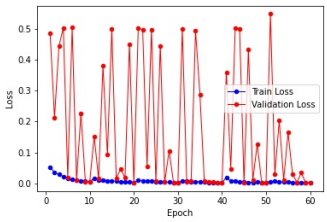

Fig. 4: HIGHT Validation loss Vs Train Loss (CNN)

Table 3: Reduced lower bound of data complexity

Cipher	Rounds	Best known Data Complexity	Lower Bound of Data Complexity (Our Approach)
HIGHT [17]	13	2^{61}	2^{30}
LEA [11]	11	2^{98}	2^{48}
SPARX [1]	9	2^{37}	2^{25}
SAND-128 [7]	7	2^{24}	2^{12}

3.3 Estimation of lower bound of data complexity

Choose one classical differential distinguisher with differential characteristics $\delta_i \rightarrow \delta_o$ of m rounds with probability 2^{-p}, where $m \leq r$. Following Theorem 1, the lower bound of data complexity of the classical differential distinguisher is bounded by $2/(2^{-p}-1/2^m-1)$. Next, form a deep neural distinguisher $MLD_{PD_{Best}}$ by taking δ_o as the input plaintext difference(using Algorithm 4).

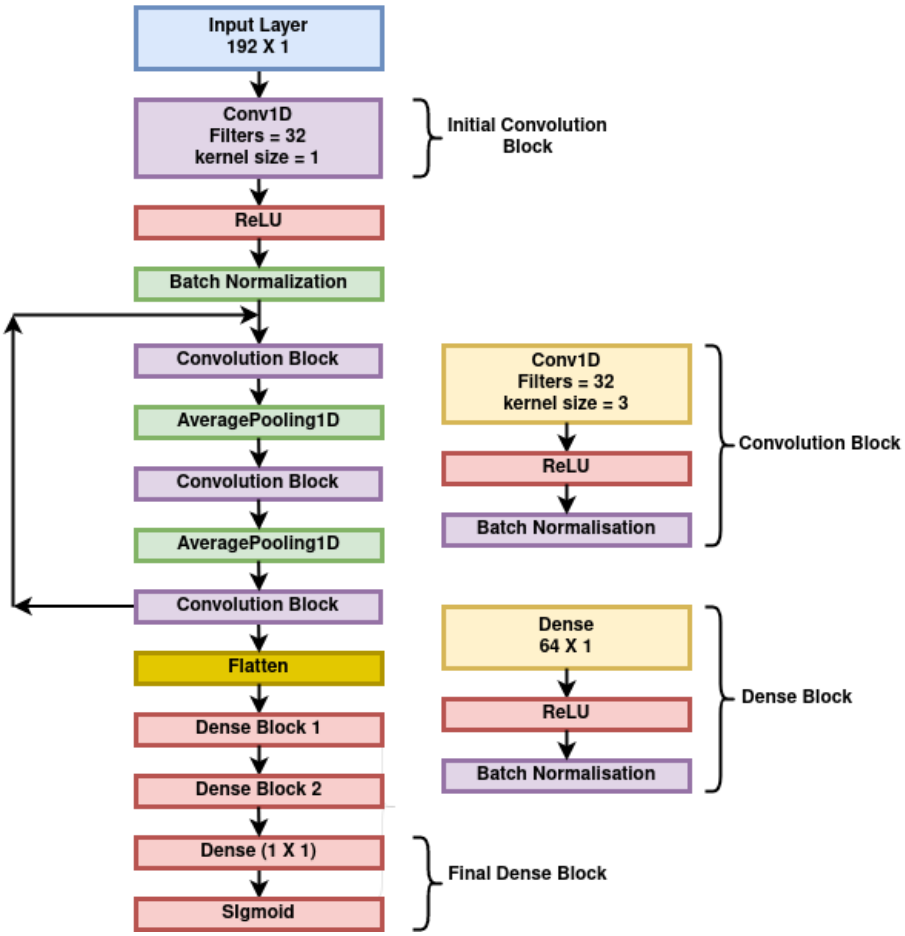

Fig. 5: CNN Architecture

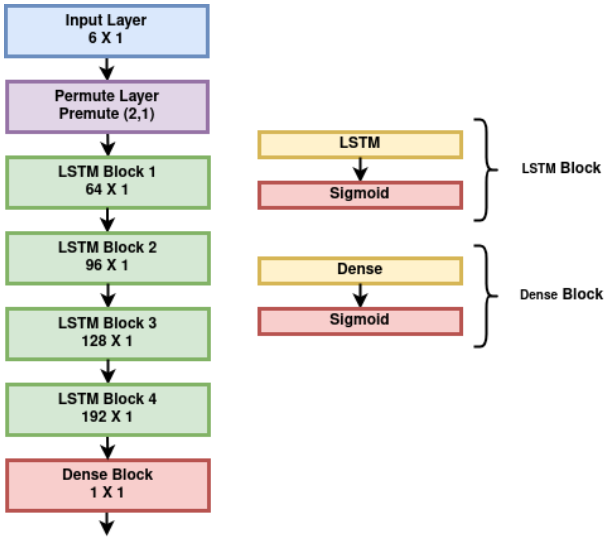

Fig. 6: LSTM Architecture

Suppose $MLD_{PD_{Best}}$ gives a high accuracy up to n rounds. Then we conclude the lower bound of the data complexity for differential cryptanalysis attack for the corresponding distinguisher with $(m+n)$ rounds is approximately 2^p. Here, we apply the Algorithm 4 for approximation of new lower bound of data complexity. The training and validation data is an one time overhead cost needed during the learning(applying Algorithm 2) and testing phase(applying Algorithm 3) of the distinguisher $MLD_{PD_{Best}}$. The new lower bound of data complexity estimation for HIGHT, LEA SPARX and SAND128 is provided in Table 3.

Table 4: Model details for real plaintext difference $0x00800000$ of 9 rounds HIGHT

Network	Activation Function	No. of Parameters	Train Time (s)	Valid. Acc.
CNN	ReLU/ sigmoid	475,777	309	68.58
LSTM	sigmoid	441,921	54	50.33

4 Experimental Results

In this section, we describe the outcome of the generic neural distinguisher after application on HIGHT, LEA, SPARX and SAND. We use google collab with installed Keras-GPU for all our ML-related experiments, including data generation. We run three different models, Convolutional Neural Network (CNN) [15],

Light Gradient Boosting Machine (LGBM) [13], and Long Short-Term Memory (LSTM) [10], for training and validation of datasets. In general, we use a total of 10^5 data samples, of which 50% is applied for training and the rest for validation.

For CNN and LSTM, we varied the number of layers, number of neurons per layer, and number of blocks per layer. Also, we applied different types of activation functions. The CNN architecture used for generating our distinguishers is depicted in Figure 5. In initial convolution block we have applied one dimensional convolution operation with 32 filters and kernel size is one. The activation function used here is ReLU. For the intermediate convolution blocks we choose one dimensional convolution operation with 32 filters, kernel size is three and same activation function. During convolution operation, the regularizationn parameter value is set to 0.0001. Compilation of the model takes the mean squared error loss function and adam optimizer. The details of model architecture is provided in Table 4.

Figure 6 describes the details of LSTM model. Four LSTM layers and one dense block is applied. For all the dense and LSTM layer output is managed via sigmoid activation function.

The input for the distinguisher is $(C_1, C_2, C_1 \oplus C_2, L)$, where L is the label defining true or random plaintext difference is used for dataet creation. During data preparation we divide the

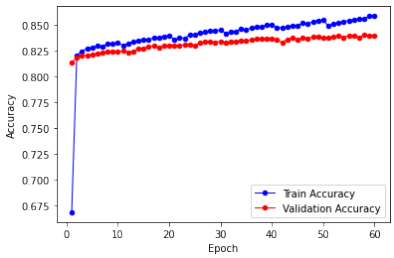

Fig. 7: HIGHT Validation Accuracy Vs Training Accuracy (LSTM)

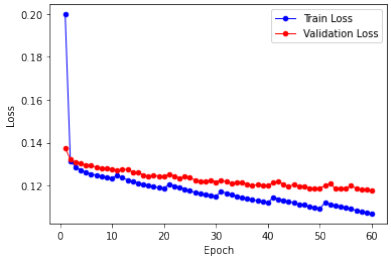

Fig. 8: HIGHT Validation loss Vs Training Loss (LSTM)

4.1 HIGHT

The designers of HIGHT [12] 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. [2] 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}. The list of differential distinguishers for hight are provided in Table 5. For the 13-round differential distinguisher, we mention the input and output differences

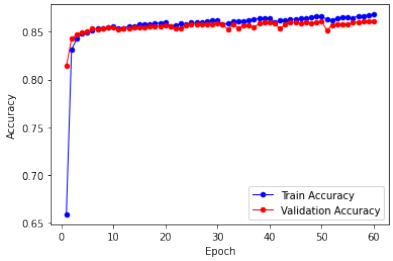

Fig. 9: LEA Validation Accuracy Vs Training Accuracy (LSTM)

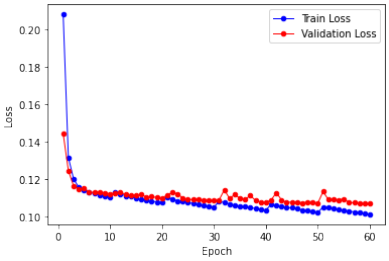

Fig. 10: LEA Validation loss Vs Training Loss (LSTM)

Table 5: Classical distinguishers of round reduced HIGHT

Rnd.	Input Difference	Output Difference	Prob.
5[12]	8201000000000000	009095CA01000000	2^{-12}
5[12]	0000000082010000	01000000009095CA	2^{-12}
6[12]	4282010000000000	009095CA01000000	2^{-17}
6[12]	0000000042820100	01000000009095CA	2^{-17}
11[12]	118925E2C8010000	4502010000912995	2^{-58}
11[12]	C8010000118925E2)	0091299545020100	2^{-58}
12[2]	00008227213AEA01	00B6F801009002E8	2^{-53}
13[2]	80008AC28A01A0BB	007B2A80009002A7	2^{-61}

from round one to round 13 in Table 6, where p means the probability of the differential trail.

Observation

In [12] $\delta_i = 0x0800000000000000$ is used as a classical distinguisher for the output difference of sixth round. For the automatic generic neural distinguisher, we take this $\delta_i = 0x0800000000000000$ as the input plaintext difference for the seventh round. Here, we construct a six-round neural distinguisher that provides high accuracy, and this one works as a 13-round distinguisher. From Table 6, we can calculate the total data complexity of this distinguisher, which is at least 2^{30}.

The LSTM model can classify the corresponding real and random ciphertext differences up to 14 rounds, whereas the LGBM and CNN model performs better and allows up to 16 rounds. For all three models, we describe the result using training accuracy, validation accuracy, true positive rate, and true negative rate. Table 7 depicts the performance of the CNN and LGBM model. Figure 3, and 7 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 4, and 8 for CNN and LSTM model.

Table 6: Classical distinguishers of 13-round HIGHT[17]

Rnd.	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	8080000000000000	0
8	80000000000000C3	-3
9	000000000072C380	-4
10	0000000C72E98000	-9
11	00A70CF2E9800000	-4
12	A700F22A80000002	-6
13	007B2A80009002A7	-5

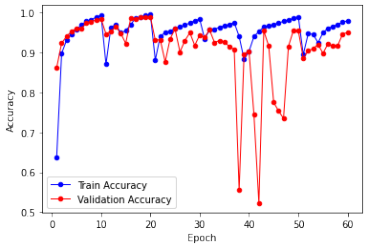

Fig. 11: LEA Validation Acc Vs Train Accuracy (CNN)

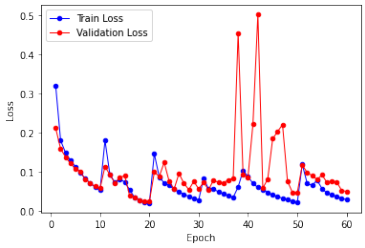

Fig. 12: LEA Validation loss Vs Train Loss (LSTM)

4.2 LEA

The best differential distinguisher available for LEA [11] 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 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 9.

Observation

For LEA, we use $\delta_i = (0x00000000, 0x80000000,\ 0x80000000, 0x80000000)$ as the plaintext difference given by Hong et al. [11] at Table 10. We take the

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

Model	Round No.	Train. Acc.	Valid. Acc.	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

Table 8: Accuracy, true positive rate (TPR) and true negative rate (TNR) for LEA cipher

Model	Round No.	Train. Acc.	Valid. Acc.	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 9: Classical distinguishers of 11-round LEA[17]

Rnd.	Difference	$log_2 p$
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	

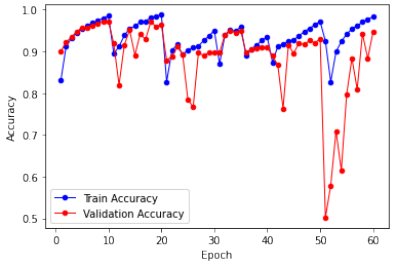

Fig. 13: Training/Validation Accuracy vs Epoch for SPARX(CNN)

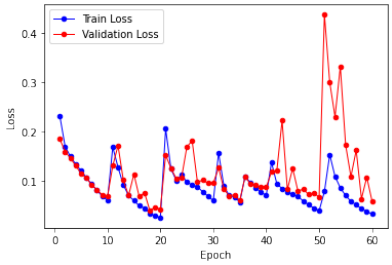

Fig. 14: Training/Validation Loss vs Epoch for SPARX(CNN)

fourth-round output difference from the eleven-round differential characteristics (given in Table 9) and apply our generic distinguisher described Algorithm 4. The CNN and LGBM model provides acceptable results for up to 13 rounds. The LSTM model can classify the cipher up to 12 rounds. We summarize the results from the CNN, LSTM, and LGBM models in Table 8. Figure 11, and 9 describes the relation between training/validation 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 9 to generic neural distinguisher as δ_i. The variation of training and validation loss by increasing epoch number is depicted in Figure 12, and 10 for CNN and LSTM model. In this case, our generic distinguisher can also classify LEA up to 13 rounds. Here we can redefine the lower bound of data complexity of 11-round LEA to 2^{49} considering $\delta_i = (0x00000000, 0x80000000, 0x80000000, 0x80000000)$, provided in Table 9.

Table 10: Accuracy, true positive rate (TPR) and true negative rate (TNR) for SPARX

Model	Rnd. No.	Train. Acc.	Valid. Acc.	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

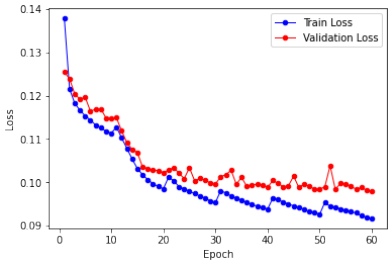

Fig. 15: Training/Validation Loss vs Epoch for SPARX (LSTM)

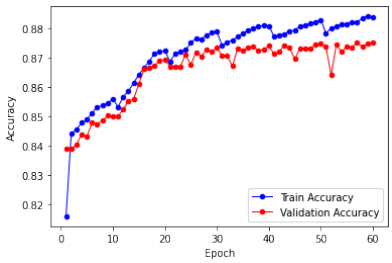

Fig. 16: Training/Validation Accuracy vs Epoch for SPARX(LSTM)

4.3 SPARX

Ralph et al. [1] 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. [1] presents all the input and output differences for optimal differential trials up to ten rounds. The 9-round differential distinguisher, along with each of the input differences and output differences from round one to round-9 is shown in Table 11, where $\mathcal{L}$ is a fiestel function.

Table 11: Classical distinguishers of 9-round SPARX [1]

Rounds	Difference	$log_2 p$
0	00000000 00508402	-
1	00000000 24023408	4
2	00000000 50c080e0	7
3	00000000 01810203	5
$\mathcal{L}$	01810203 00000000	0
4	000c0800 00000000	5
5	20000000 00000000	3
6	00400040 00000000	1
$\mathcal{L}$	00400040 00400040	0
7	80408140 80408140	4
8	00400542 00400542	6
9	8542904a 8542904a	8
$\mathcal{L}$	08150815 8542904a	0

Table 12: Accuracy, true positive rate (TPR) and true negative rate (TNR) for SAND64 cipher

Model	Round No.	Train. Acc.	Valid. Acc.	TPR	TNR
CNN	4	99.9	99.9	0.999	0.929
	5	99.9	99.94	0.998	1.00
	6	97.66	97.61	0.953	0.999
	7	50.62	50.58	0.628	0.369
	8	50.19	50.25	0.269	0.721
LGBM	4	99.86	99.98	1.00	0.997
	5	99.36	99.34	0.999	0.986
	6	90.41	90.22	0.994	0.811
	7	64.54	51.45	0.467	0.561
LSTM	4	97.00	96.73	0.968	0.967
	5	94.99	94.93	0.931	0.967
	6	75.69	74.66	0.694	0.799

Table 13: Accuracy, true positive rate (TPR) and true negative rate (TNR) for SAND128 cipher

Model	Round No.	Train. Acc.	Valid. Acc.	TPR	TNR
CNN	5	99.9	99.9	0.999	0.999
	6	99.9	99.99	0.999	0.999
	7	99.01	99.23	0.979	0.999
	8	90.13	89.69	0.769	0.989
	9	50.18	50.38	0.701	0.296
LGBM	4	100	100	1.00	0.999
	5	100	99.97	0.999	0.999
	6	99.56	99.27	0.993	0.991
	7	64.01	50.44	0.523	0.486
	8	64.52	50.13	0.501	0.402
LSTM	4	98.66	98.33	0.967	0.999
	5	98.51	98.66	0.976	0.997
	6	92.31	92.17	0.852	0.99

Observation

Ralph et al. [1] found the five round output difference $\delta_i = 0x00400040/0x00000000$ to describe the six round differential trail(see Table 11). 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 10. The models also provide good accuracy for input difference $\delta_i = 0x00408000/0x00000000$ and $\delta_i = 0x28000010$, $0x28000010$ up to six rounds.

Figure 13 and 16 describe the relation between training/validation 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 14, and 15 for the CNN and LSTM model. We found the lower bound of data complexity of 9-round as 2^{25} using $\delta_i = 0x00400040/0x00000000$(see Table 11).

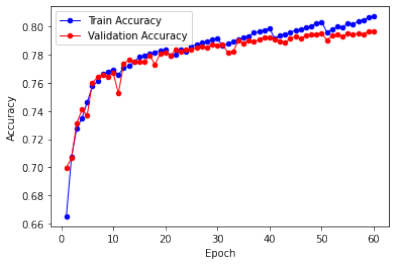

Fig. 17: Training/Validation Accuracy vs Epoch for SAND64 (LSTM)

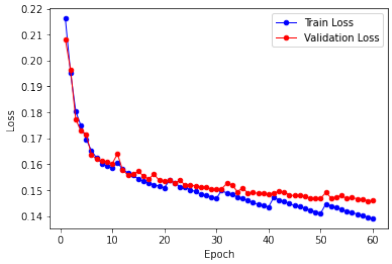

Fig. 18: Training/Validation Loss vs Epoch for SAND64 (LSTM)

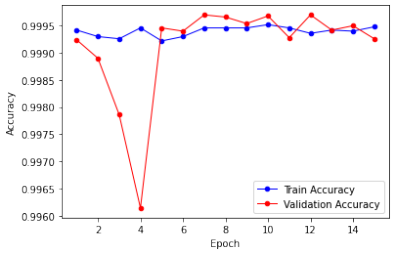

Fig. 19: Training/Validation Accuracy vs Epoch for SAND128 (CNN)

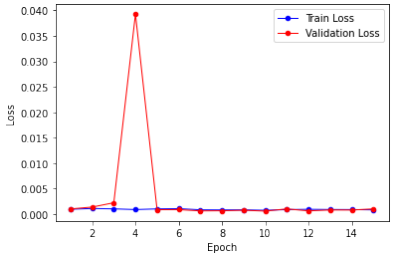

Fig. 20: Training/Validation Loss vs Epoch for SAND128 (CNN)

4.4 SAND-64/128

The designers of SAND [7] provided a seven round differential characteristics for SAND128 with probability 2^{-24} and uses plaintext difference as (0x09000890 88880230).

Observation

In case of SAND-128, we use (0x800000000 00000000) as plaintext difference, which is the input of the fifth round of the classical distinguisher defined in Table [7]. We found distinguishers with good accuracy up to nine rounds. The training and validation accuracy of the CNN, LGBM, and LSTM models with true positive and negative rates is shown in Table 13. Figure 19, and 21 describes the relation between training/validation accuracy and the number of epochs for CNN and LSTM model. Figure 20, and 22 describes the relation between training/validation loss and the number of epochs for CNN and LSTM model. In total a 14 round neural distinguisher is achieved. For seven rounds distinguisher we can compare the data complexity of with the designers claim, which is 2^{24}. In our case the lower bound of data complexity is reduced to 2^{12}. For SAND64 we apply (0x80000000) as input difference of our neural distinguisher. In this case we found distinguishers with good accuracy up to eight rounds. The training and validation accuracy with true positive and negative rates is shown in Table 12.Figure 17 describes the relation between training/validation accuracy and the number of epochs for LSTM model. Figure 18 describes the relation between training/validation loss and the number of epochs for LSTM model.

5 Conclusion

In this paper, we introduce a novel technique for finding neural differential distinguishers. Using the tool, we report neural classifiers for the cipher HIGHT up to 14-rounds, LEA up to 13-rounds, up to nine rounds for SPARX and up to 14 rounds for SAND128, which are the first neural distinguisher for the cipher. To

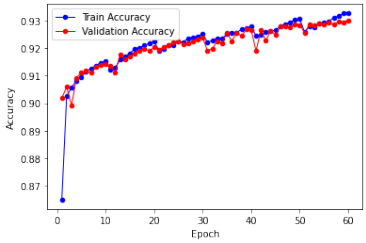

Fig. 21: Training/Validation Accuracy vs Epoch for SAND128 (LSTM)

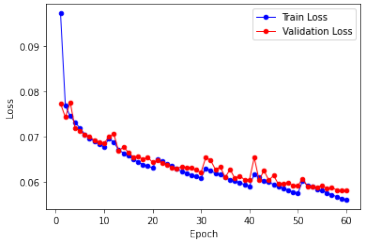

Fig. 22: Training/Validation Loss vs Epoch for SAND128 (LSTM)

the best of our knowledge, this is the first neural distinguisher for HIGHT, LEA, SAND and SPARX. We have performed an experiment to describe the relationship between the hamming weight of input difference of a neural distinguisher and the corresponding maximum round number of the cipher. A general approach for finding the lower bound of data complexity for differential cryptanalysis is also provided. As a future work we want to cover more ciphers applying our tool for finding new differential distinguishers.

References

1. 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 (2018). https://doi.org/10.1007/978-3-319-93387-0_24, https://doi.org/10.1007/978-3-319-93387-0_24
2. 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, https://doi.org/10.1049/iet-ifs.2018.5539
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 (2021). https://doi.org/10.23919/DATE51398.2021.9474092, https://doi.org/10.23919/DATE51398.2021.9474092
4. 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 (2017). https://doi.org/10.1007/978-3-319-66787-4_16, https://doi.org/10.1007/978-3-319-66787-4_16
5. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK lightweight block ciphers. In: Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015. pp.

175:1–175:6. ACM (2015). https://doi.org/10.1145/2744769.2747946, https://doi.org/10.1145/2744769.2747946
6. Biham, E., Shamir, A.: Differential Cryptanalysis of the Data Encryption Standard (1993). https://doi.org/10.1007/978-1-4613-9314-6, https://doi.org/10.1007/978-1-4613-9314-6
7. Chen, S., Fan, Y., Sun, L., Fu, Y., Zhou, H., Li, Y., Wang, M., Wang, W., Guo, C.: SAND: an AND-RX feistel lightweight block cipher supporting s-box-based security evaluations. Des. Codes Cryptogr. **90**(1), 155–198 (2022). https://doi.org/10.1007/s10623-021-00970-9, https://doi.org/10.1007/s10623-021-00970-9
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. 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 (2019). https://doi.org/10.1007/978-3-030-26951-7_6, https://doi.org/10.1007/978-3-030-26951-7_6
10. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997), https://doi.org/10.1162/neco.1997.9.8.1735
11. 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 (2013). https://doi.org/10.1007/978-3-319-05149-9_1, https://doi.org/10.1007/978-3-319-05149-9_1
12. 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 (2006). https://doi.org/10.1007/11894063_4, https://doi.org/10.1007/11894063_4
13. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.: Lightgbm: A highly efficient gradient boosting decision tree. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA. pp. 3146–3154 (2017), https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html
14. 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 Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings. Lecture Notes in Computer Science, vol. 547, pp. 17–38 (1991). https://doi.org/10.1007/3-540-46416-6_2, https://doi.org/10.1007/3-540-46416-6_2

15. O'Shea, K., Nash, R.: An introduction to convolutional neural networks. CoRR **abs/1511.08458** (2015), http://arxiv.org/abs/1511.08458
16. 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 (2021). https://doi.org/10.1007/978-3-030-88238-9_10, https://doi.org/10.1007/978-3-030-88238-9_10
17. 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 - 13th International Conference, Inscrypt 2017, Xi'an, China, November 3-5, 2017, Revised Selected Papers. Lecture Notes in Computer Science, vol. 10726, pp. 404–426 (2017). https://doi.org/10.1007/978-3-319-75160-3_24, https://doi.org/10.1007/978-3-319-75160-3_24

A Brief Description of HIGHT, LEA, SPARX and SAND

Brief description of HIGHT HIGHT is an ARX-based block cipher proposed by Hong et al. [12] at CHESS 2006. It is a lightweight cipher, and it performs fast. 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 bits. 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 schedule, key whitening phase, round function, and final key exoring.

- **Key schedule** The key schedule function 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 transformation** 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 function using the following operations.
- **Round function** Round Function generates $T_i = T_{i,7}|| \dots ||T_{i,0}$ into $T_{i+1} = T_{i+1,7}|| \dots ||T_{i+1,0}$ described in Algorithm 5.
- **Final transformation** Final transformation generates the ciphertexts using the temporary plaintext P and four whitening keys W_4, W_5, W_6, W_7.

In the decryption process of HIGHT, the key schedule function produces the 128 round keys in reverse order.

LEA LEA is an ARX-based block cipher proposed by Hong et al. [11]. 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 bits key size for 32 rounds. Assume plaintext be $P = P_0, P_1, P_2, P_3$, master key

Algorithm 5 HIGHT Encryption

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

procedure HIGHT_ENC(P, N)
  $T_{0,0} \leftarrow P_0 \boxplus W_0, T_{0,1} \leftarrow P_1$
  $T_{0,2} \leftarrow P_2 \oplus W_1, T_{0,3} \leftarrow P_3$
  $T_{0,4} \leftarrow P_4 \boxplus W_2, T_{0,5} \leftarrow P_5$
  $T_{0,6} \leftarrow P_6 \oplus W_3, T_{0,7} \leftarrow P_7$
  for i= 0 to 31 **do**
    $T_{i+1,1} \rightarrow T_{i,0}$
    $T_{i+1,3} \rightarrow T_{i,2}$
    $T_{i+1,5} \rightarrow T_{i,4}$
    $T_{i+1,7} \rightarrow T_{i,6}$
    $T_{i+1,0} = T_{i,7} \oplus (F_0(T_{i,6}) \boxplus RK_{4i+3})$,
    $T_{i+1,2} = T_{i,1} \boxplus (F_1(T_{i,0}) \oplus RK_{4i+2})$
    $T_{i+1,4} = T_{i,3} \oplus (F_0(T_{i,2} \boxplus RK_{4i+1})$,
    $T_{i+1,6} = T_{i,5} \boxplus (F_1(T_{i,4}) \oplus RK_{4i})$.
  end for
  $C_0 \leftarrow T_{32,1} \boxplus W_4, C_1 \leftarrow T_{32,2}$
  $C_2 \leftarrow T_{32,3} \oplus W_5, C_3 \leftarrow T_{32,4}$
  $C_4 \leftarrow T_{32,5} \boxplus W_6, C_5 \leftarrow T_{32,6}$
  $C_6 \leftarrow T_{32,7} \oplus W_7, C_7 \leftarrow X_{32,0}$
  Return C
end procedure

$M = M_0, M_1, M_2, M_3$ and ciphertext is $C = C_0, C_1, C_2, C_3$ with each P_i, M_i, C_i is a 32 bit word, $i \in \{0, 1, 2, 3\}$. The encryption process first initializes P to T_0. A round operation of 128 bit LEA we describe in Algorithm 6

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. Mainly SPARX encryption executes n_s steps and each step consists of r_a times ARX box and key addition for each of 32 bit words. Each step also performs a linear mixing operation δ for w-bit words. The SPARX 64/128 is formed by a Feistel network with two state words having eight Feistel steps. Each step consists of three rounds of an ARX-based round function(three-round 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. The details of SPARX encryption procedure is described in Algorithm 7

SAND In 2022 Chen et al. [7] proposes a new AND-RX based family of block ciphers SAND. Two versions are available for SAND. For SAND64 the size of

Algorithm 6 LEA Encryption

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

procedure LEA_ENC(P, N)

  for i= 0 to 23 **do**
    $T_0 = RL^{<<1}(T_0 \boxplus RL^{<<i}(\delta_{i\%4}))$ ▷ δ is an array of constants.
    $T_1 = RL^{<<3}(T_1 \boxplus RL^{<<(i+1)}(\delta_{i\%4}))$,
    $T_2 = RL^{<<6}(T_2 \boxplus RL^{<<(i+2)}(\delta_{i\%4}))$,
    $T_3 = RL^{<<11}(T_3 \boxplus RL^{<<(i+3)}(\delta_{i\%4}))$,
    $RK_i = [T_0, T_1, T_2, T_1, T_3, T_1]$
  end for
  $T_i = P_i$, $i \in \{0, 1, 2, 3\}$

  for i= 0 to 23 **do**
    $T_{i+1,0} = RL^{<<9}((T_{i,0} \oplus RK_{i,0}) \boxplus (T_{i,1} \oplus RK_{i,1}))$,
    $T_{i+1,1} = RR^{<<5}((T_{i,1} \oplus RK_{i,2}) \boxplus (T_{i,2} \oplus RK_{i,3}))$,
    $T_{i+1,2} = RR^{<<3}((T_{i,2} \oplus RK_{i,4}) \boxplus (T_{i,3} \oplus RK_{i,5}))$,
    $T_{i+1,3} = T_{i,0}$
  end for
  Set $C = [T_{i,0}, T_{i,1}, T_{i,2}, T_{i,3}]$.
  Return C
end procedure

block is 64 bits and the round number is 48. For SAND128 the block size is 128 bits and round number is 54. The size is 128 bits for both cases. Assume $P = (P_l, P_r)$ be the input plaintext with P_l is the left side n bits and P_r be the right side n bits. We denote $C = (C_l, C_r)$ be the ciphertext with C_l is the left side n bits and C_r be the right side n bits. Suppose (T_1^r, T_2^r) be the temporary current state and RK^r be the round key for the r^{th} round then the state for the next round can be determined as, $(T_1^r, T_2^r) = (P_n(G_0(T_1^r \lll_{n/4} \alpha) \oplus G_1(T_1^r \ll_{n/4} \beta)) \oplus T_2^r \oplus RK^r, T_1^r)$ Here G_0 and G_1 are the non-linear functions defined as follows, $G_0(x_0, x_1, x_2, x_3) = (y_0, y_1, y_2, y_3) = (x_3 \odot x_2 \oplus x_0, x_1, x_2, y_0 \odot x_1 \oplus x_3)$, $G_0(x_0, x_1, x_2, x_3) = (y_0, y_2, y_1, y_3) = (x_0, y_3 \odot x_1 \oplus x_2, y_2 \odot x_0 \oplus x_1, x_3)$. P_n is the permutation function. The value of rotation variable (α, β) is set to $(0, 1)$.

Algorithm 7 SPARX Encryption
Inputs: The plaintext $(P_0, P_1, \ldots, P_{w-1})$, Master key $(M_0, M_1, \ldots, M_{v-1})$
Output: The ciphertext $(C_0, C_1, \ldots, C_{w-1})$

procedure SPARX_ENC(P,C)
    Let $C_i \leftarrow P_i \ \forall i \in [0, ..., w-1]$
    Let $RK_i \leftarrow M_i \ \forall i \in [0, ..., v-1]$
    for s=0 to $n_s - 1$ **do**
        for i = 0 to (w-1) **do**
            for r = 0 to $r_a - 1$ **do**
                $P_i \leftarrow P_i \oplus K_r$
                $P_i \leftarrow A(y_i)$
            end for
            $(RK_0, RK_1, \ldots, RK_{v-1}) \leftarrow KEY\_SCH(RK_0, RK_1, \ldots, RK_{v-1})$
        end for
        $(C_0, C_1, \ldots, C_{w-1}) \leftarrow \lambda_w(C_0, C_1, \ldots, C_{w-1})$
    end for
    $C_i \leftarrow C_i \oplus RK_i \ \forall i \in [0, ..., w-1]$
    return $(C_0, C_1, \ldots, C_{w-1})$
    Return C
end procedure