Deep Learning based Differential Classifier of PRIDE and RC5

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

Crypto Research Lab, IIT Kharagpur, India
{debranjan.crl, mandal.up98}@gmail.com,
{abhij, drc}@cse.iitkgp.ac.in

Abstract. Deep learning-based cryptanalysis is one of the emerging trends in recent times. Differential cryptanalysis is one of the most potent approaches to classical cryptanalysis. Researchers are now modeling classical differential cryptanalysis by applying deep learning-based techniques. In this paper, we report deep learning-based differential distinguishers for block cipher PRIDE and RC5, utilizing deep learning models: CNN, LGBM and LSTM. We found distinguishers up to 23 rounds for PRIDE and nine rounds for RC5. To the best of our knowledge this is the first deep learning based differential classifier for cipher PRIDE and RC5.

Keywords: Deep Learning · Block Cipher · Differential Cryptanalysis · Neural Distinguisher · PRIDE · RC5

1 Introduction

Researchers are using deep learning techniques to solve problems from a variety of domains. Nowadays, cryptology researchers are also applying deep learning(DL) mainly for cryptanalysis purposes. Rivest [13] first introduced the relationship between the area of machine learning and cryptography. In the cryptography domain, earlier DL was restricted only to side-channel analysis. In 2019, Aron Gohr [7] first showed a way to find deep learning-based distinguishers. He tries to differentiate between the performance of the classical differential distinguisher and the neural distinguisher. He reveals deep learning-based classifiers perform more efficiently for key recovery of round-reduced SPECK32/64. In 2021, T Yadav et al. [15] use the Ghors approach and reports new neural distinguishers for reduced rounds of SIMON-32 [4], SPECK-32 [4] and GIFT-64 [3]. Baksi et al. [2] proposed a technique by utilizing multiple input differences to generate an ML-based distinguisher. They successfully applied the technique to reduced rounds of Ascon permutation, Chaskey permutation, Knot256/512 permutation, and Gimili permutation/hash/cipher to retrieve the corresponding neural distinguishers. Pal et al. [12] proposes a generic tool for generating deep learning-aided differential distinguishers and applies the strategy to reduced rounds of HIGHT, LEA, SPARX, and SAND64/128. Liu et al. [17] uses Gohr's

approach and found improved distinguishers for PRESENT, DES, and Chaskey. In 2022 Gohr et al. [8] report a new approach applying multiple ciphertext pairs. They found new distinguishers for SKINNY, Katan, and ChaCha and improved distinguishers for PRESENT, SIMON, and SPECK. Zezhou et al. [9] uses neural distinguishers to recover the last subkey of 11 and 13 rounds of SIMON32. Here our objectives are as follows,

- Searching deep learning based differential distinguishers for the cipher PRIDE and RC5.
- Applying different different deep learning models to get more accuracy for neural distinguishers.
- Increasing number of rounds for the neural distinguishers.

1.1 Our contribution

In this paper, we introduce differential distinguishers for block cipher PRIDE[1] and RC5 [14] utilizing deep learning-based techniques. We found neural-based classifiers up to 23 rounds for PRIDE and distinguishers up to nine rounds for RC5.

1.2 Organization of the paper

We organize the rest of this paper in the following manner. Section 2 explains the backgrounds and preliminaries. Here we describe the specifications of PRIDE and RC5 in brief, the description of deep learning models used and a literature review on differential cryptanalysis attacks on PRIDE and RC5. In Section 3, we present our approach to finding neural classifiers. The experimental results are shown in Section 4. Section 5 concludes the paper.

2 Background

In this subsection we describe the background and preliminaries of our work.

2.1 Short description of PRIDE

PRIDE follows SPN structure with block size 64 bits and 128 bits key size. The 128 bit master key K is divided into two parts, k_0 and k_1, each of 64 bits. The first part, k_0, is applied as a whitening key for both pre-whitening and post-whitening. The last part, k_1, is again divided into eight bytes, which are used for generating the round keys. It iterates a total of 20 rounds. Each round of PRIDE is mainly composed of three operations:

- **AddRoundKey** Exor the state with the round key.
- **Substitution Operation** After key Exoring the state is applied to 16 parallel SBox(see Table 1).
- **Linear Layer** PRIDE uses a linear layer L, which consists of three parts: first, a permutation layer P is used, then a constant matrix M is multiplied with the state and finally, the inverse of the first permutation P^{-1} is applied. The values for M, P, P^{-1} is provided in specification of PRIDE [1].

Table 1: PRIDE SBbox

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

Algorithm 1 Key Expansion of RC5

procedure KEY_EXPANSION(K,S,rounds)
  b ← 16
  w ← 32
  u = w/8
  c = b/u
  t = 2*(rounds + 1)
  P ← 0xb7e15163
  Q ← 0x9e3779b9
  S[0] = P
  for i = 1 to t **do**
    $S[i] = (S[i-1] + Q) \mod 2^{32}$
  end for
  for i = b-1 to 0 **do**
    $L[i/u] = ((L[int(i/u)] << 8) + K[i]) \mod 2^{32}$
  end for
  $i = j = X = Y = 0$
  for k = 0 to 3t - 1 **do**
    $X = S[i] = ROTL((S[i] + ((X + Y) \mod 2^{32})) \mod 2^{32}, 3)$
    $Y = L[j] = ROTL((L[j] + ((X+Y) \mod 2^{32})) \mod 2^{32} (X+Y) \mod 2^{32})$
    $i = int((i + 1) \mod t)$
    $j = int((j + 1) \mod c)$
  end for
end procedure

2.2 Short description of RC5

RC5 [14] is a symmetric key block cipher. The plaintext and ciphertext are represented as a two-word block. For word size, the allowable values are 16, 32, and 64. The design parameters of RC5 consist of word size w, number of rounds r, the number of bytes in the secret key K is b, and the secret key K represented as $K = K[0], K[1], \ldots, K[b-1]$. The expanded key is stored in table S. The encryption module of RC5 consists of two parts,

(a) Key Expansion The key expansion function helps to expand the main key K, to form a table S. The process for key expansion is provided in Algorithm 1. Here, P and Q are magic constants, P_w = Odd((e - 2)2^w) and Q_w = Odd((ϕ - 1)2^w), where e = 2.71828182 and ϕ = 1.618033988749. The function ROTL(x,y), is the circular rotation of the word x by y bits.

(b) Encryption The encryption function works with the help of elements in the key expansion table S. The encryption algorithm is as follows:

$$X = X + S[0]$$

$Y = Y + S[1]$
for $i = 1$ to rounds,
$X = (\text{ROTL}((X \oplus Y), Y) + S[2 * i]) \mod 2^{32}$
$Y = (\text{ROTL}((Y \oplus X), X) + S[2 * i + 1]) \mod 2^{32}$

Here, X and Y are the w-bit words of the input plaintext. After r number of rounds, the X and Y will be the two words of ciphertext i.e $Z = [X, Y]$.

2.3 Deep Learning Models

For finding the differential distinguishers, we use three models CNN, LSTM, and LGBM. Now we describe about these three models in brief.

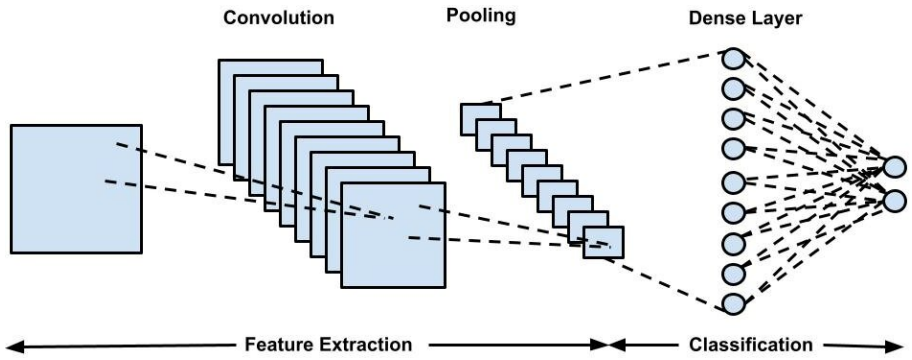

Fig. 1: CNN architecture

Convolutional Neural Network (CNN)

CNN is the most renowned and important Deep Neural Network. The CNN architecture is comparable to the way the neurons are connected in the Human Brain. The emergence of CNN was a motivation by the organization of the visual cortex. Each neuron responds to impulses only in a confined portion of the visual area called the receptive field. A set of such fields overlap to fill the whole visual region. The architecture of a CNN mainly consists of the following layers,

- **Convolution Layer:** The convolution layer has following characteristics,
  i) Unlike the feed-forward network, it follows sparse connectivity. The sparse connectivity reduces the number of parameters in the model.
  ii) Weight sharing is an important property of CNN. The kernel weight needs to be the same for different portions of the same image.
  The purpose of the convolution layer is to take out the high-level features, like edges, from an image. The convolution layer in the first level helps to find the low-level features like color, edges, etc. But as the level goes deeper, it grabs a good knowledge of the high-level features and provides a network with a wholesome perception of images in the dataset.

- **Pooling Layer:** The objective of pooling is to decrease the dimension of the feature map that we get from the convolution layer. By dimensionality reduction, it reduces the learning parameters and the computational power which is required to process the data. The pooling layer also helps to fetch the dominant features, which are rotational and positional invariant. Max pooling and average pooling are two types of pooling used in CNN. Max pooling fetches the maximum value from that part that is covered by the kernel. Average pooling computes and returns the average of all the values.
- **Fully Connected Layer:** This layer densely connects all the neurons of the last layer with the next layer. The effect of this layer can be easily calculated by using matrix multiplication and then adding a bias.
- **Non-linear Layer:** A vital layer called activation functions adds non-linearity to the CNN architecture. Depending on the value of the activation function, this layer predicts if the neuron of the next layer will be active or not. Primarily used activation functions are sigmoid, ReLU, softmax, tanh, etc.

Figure 1 depicts a top-level view of CNN architecture. After adding a sequence of convolution layer and pooling layer consecutively, the model is ready to understand the features. The model will distinguish between dominating and certain low-level features and classify them over a series of epochs.

Long Short-Term Memory (LSTM)

LSTM is a variant of RNN which reduces the problem of short-term memory. In RNN, the state s_i records information from the earlier time steps. For each new time instant, the past information gets morphed by the current information as the size of the state is of finite dimension. On the one hand, the information needs to get recorded from all the previous time steps. Still, on the other hand, the memory amount is finite, so it is bound to get overridden. The information will get morphed so much that it will be completely impossible to say the original contribution at step 1 or 2 after 20 or 30-time steps. This is the main problem of RNN, which the LSTM model partially resolves. Figure 2 narrates the architecture of the LSTM model. LSTM resolves the short-term memory problem of RNN. To resolve the problem of short-term memory, three items were included: selective read, selective write, and selective forget.

- **Selective Write** In case of RNN the information of s_{t-1} is being used to compute s_t , $s_t = \sigma(Ws_{t-1} + Ux_t)$. But here in LSTM, only some portion of the information is passed from the previous state s_{t-1} to the next state s_t. A gate o_{t-1} is introduced, and it is an output gate. Then, an elementwise product is performed between the output gate and the cell state s_{t-1}, and the new product is written into h_{t-1}. The output gate o_{t-1} is computed as $o_{t-1} = \sigma(W_o h_{t-1} + U_o x_{t-1} + b_o)$. W_o, U_o are the parameters, and b_o is the bias. The sigmoid function ensures that all the values are between 0 to 1.
- **Selective Read** By using h_{t-1}, the new state at the next time step needs to be computed. From the previously computed h_{t-1} and the new information

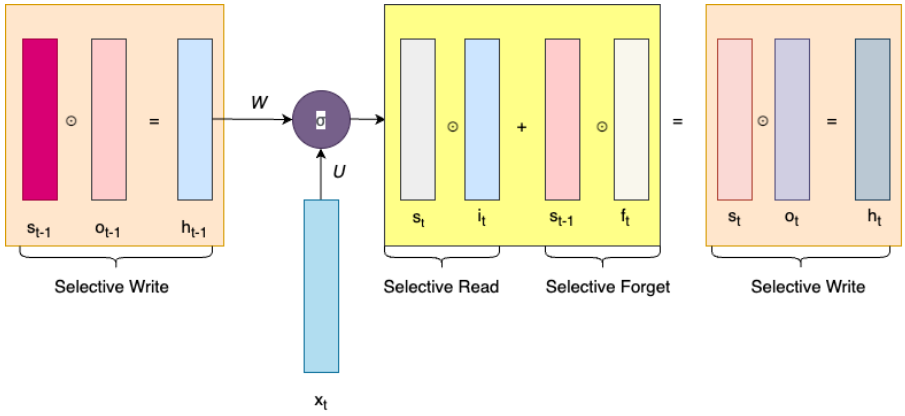

Fig. 2: LSTM architecture

x_t at the time step t, intermediate state s'_t is computed. s'_t captures all the information from the last state h_{t-1} and the next input x_t. A new gate is introduced as i_t. i_t is computed as $i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$. i_t multiplied by s'_t is the selectively read information.

– **Selective Forget** In selective forget, s_{t-1} and s'_t are both combined to get the new state s_t. Before combining, some part of s_{t-1} needs to get forgotten. So, the forget gate f_t is being introduced. $f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$. Finally s_t is computed as $s_t = f_t \bigodot s_{t-1} + i_t \bigodot s'_t$.

The whole architecture of LSTM using all three above-mentioned items is sketched out in Figure 2.

Light Gradient Boosting Machine (LGBM)

LGBM is one of the boosting algorithms. Boosting algorithm is an ensemble technique. When there are multiple models such as M_1, M_2, ..., M_n and there is a baseline model M_0, at first M_0 is made, the prediction on a finite set record and whatever errors are achieved from that, the next model is trained with that and so on and so forth it continues. In the end, all the models are combined, and a final model is made. LGBM consists of the following characteristics,

– The training speed is really high. It is very much efficient.
– It can handle large-scale data.
– Memory usage is less.
– LGBM model provides parallel, distributed, and GPU learning.

2.4 Differential Cryptanalysis of PRIDE and RC5

Differential cryptanalysis[5] is a chosen-plaintext attack that, by speculating on a key establishes a probabilistic relationship between the second last round state difference and the ciphertext difference.

Algorithm 2 Generate Data for Neural Distinguisher

Inputs: An input differences(Δ_i) for the differential distinguisher
Outputs: The Training/Validation Dataset $DATA_T/DATA_V$

procedure GENERATEDATASET(Δ_i, Rnd, Itr)
  $DATA \leftarrow \{\}$ ▷ Empty set
  for i = 1 to Itr **do**
    $MK \leftarrow$GenerateRandomKey()
    $PT_1 \leftarrow$GenerateRandomPlaintext()
    if $i \mod 2 = 0$ **then**
      $PT_2 \leftarrow$GenerateRandomPlaintext()
      $CT_1 \leftarrow$ RANDOM_ORACLE(PT_1, MK, Rnd) ▷ Encryption oracle of a cipher
      $CT_2 \leftarrow$ RANDOM_ORACLE(PT_2, MK, Rnd)
      $DATA \leftarrow DATA \cup (CT_1, CT_2, CT_1 \oplus CT_2, 0)$
    else
      $PT_2 = PT_1 \oplus \Delta_i$
      $CT_1 \leftarrow$ RANDOM_ORACLE(PT_1, MK, Rnd)
      $CT_2 \leftarrow$ RANDOM_ORACLE(PT_2, MK, Rnd)
      $DATA \leftarrow DATA \cup (CT_1, CT_2, CT_1 \oplus CT_2, 1)$
    end if
  end for
  Return $DATA$
end procedure

Let (P, P') represent the plaintext pair and (C_m, C'_m) represent the ciphertext pair after m^{th} round. The conditional probability $Pr(\Delta C_m = \beta \mid \Delta P = \alpha)$ is then used to calculate the differential likelihood of an m-round differential $\alpha \rightarrow \beta$, where $\Delta P = P \oplus P'$ and $\Delta C_m = C_m \oplus C'_m$ and the sub-keys $K_1, \ldots, K_m$ are independent and uniformly random. The attacker determines the differential probability for each round in preparation for mounting an attack.

PRIDE Utilizing the shortcomings of the SBox and linear layer of PRIDE [1], Zhao [18] introduce 16 alternative two round differential trails and found multiple 15-round differential characteristics. They mount a differential attack on the 18-round PRIDE based on one of these differentials. They use 2^{60} chosen plaintexts. Applying an automatic tool Yang et al. [16] reports 56 differential characteristics, where also exists 24 one-round differential trails. With the help of three of them, they create a 15-round differential before launching a differential attack on the 19-round PRIDE, with data complexity 2^{62}.

RC5 Kaliski Jr et al. [10] mount differential attack on nine-round RC5(64-bit block size), using 2^{45} chosen plaintext pairs. For 12-round RC5 they need 2^{62} pairs. Alex Biryukov et al. [6] uses partial differential technique and improve the 12-round attack for RC5 with 32-bit block size. They needs only 2^{44} chosen pairs. Knudsen [11] demonstrates the shortcomings of Kaliski [10] proposed differential

Algorithm 3 Training Algorithm for Neural Distinguisher

Inputs: Training Data $DATA_T$
Outputs: Training accuracy AC_T

procedure TrainNeuralDistinguisher($DATA_T$)
  Generate the DL model DL_{Δ_i}.
  Perform training for DL_{Δ_i} with dataset $DATA_T$.
  Let AC_T be the training accuracy that the training algorithm returns.
  if $AC_T > 0.5$ **then**
    Generate new dataset $DATA_V$ for validation algorithm.
    Call $ValidationNeuralDistinguisher(ML_{\Delta_i}, DATA_V)$
  else
    Return "Distinguisher can't be identified"
  end if
  Return AC_T
end procedure

analysis. They improved the complexity of the differential attacks by a factor of up to 512. Additionally, they demonstrate that RC5 has a lot of weak keys in terms of differential attacks.

Algorithm 4 Validation Algorithm for Neural Distinguisher

Inputs: Validation Data $DATA_V$ and the pre-trained model DL_{Δ_i}
Outputs: Validation Accuracy AC_V

procedure ValidationNeuralDistinguisher($DL_{\Delta_i}, DATA_V$)
  Run the pretrained model DL_{Δ_i}.
  Perform validation for DL_{δ_i} by applying the validation data $DATA_V$.
  Let AC_V be the validation accuracy.
  if $AC_V > 0.5$ **then**
    Distinguisher identified.
  else
    Return "Distinguisher can't identified"
  end if
  Return AC_V
end procedure

3 Deep Learning to Model Classical Differential Cryptanalysis

In Gohr's algorithm, the input is a plaintext difference Δ_i. Let (PT_1, PT_2) be a plaintext pair, and the corresponding ciphertext pair after Rnd rounds is (CT_1, CT_2). Now for each plaintext and ciphertext pair, create an output label L such that,

$$\begin{cases} L = 0 & \text{if } PT_1 \oplus PT_2 \neq \Delta_i \\ L = 1 & \text{if } PT_1 \oplus PT_2 = \Delta_i \end{cases}$$

Classifying the real ciphertext pairs from random ciphertext pairs is the primary intention behind this. Following this idea, we create the dataset by choosing the appropriate plaintext difference for a cipher with a predetermined round. Next, we construct a deep learning model and train the model by applying the dataset. Check the training accuracy if it exceeds 50%, then generate the validation dataset applying the same plaintext difference. If the validation accuracy is greater than 50% then we conclude that a differential classifier is found for the corresponding cipher with given rounds.

Table 2: Performance of DL-based distinguishers for cipher PRIDE

DL Technique	No of Rounds	AC_T	AC_V	TPR	TNR
CNN	3	84.23	84.29	0.822	0.810
	4	54.07	54.51	0.267	0.728
	5	50.21	50.53	0.479	0.520
	6	50.31	50.45	0.181	0.811
LGBM	3	63.6	54.62	0.482	0.611
	4	63.59	50.13	0.584	0.418
	5	63.69	50.00	0.493	0.504
LSTM	3	51.08	50.53	0.813	0.195
	4	50.62	50.67	0.745	0.267

Algorithm 2 explains the process of data generation for training and testing. The input is the plaintext difference Δ_i, the number of data elements Itr(number of iterations), and the round number Rnd up to which we want to find the neural classifier for the given cipher. Take two plaintexts PT_1 and PT_2 is selected in such a way that $\Delta_i = PT_1 \oplus PT_2$. The pair (PT_1, PT_2) is called real pair. Also consider one random plaintext pair (PT_1', PT_2'). Now encryption is performed using random oracle (with the round number Rnd) to get the ciphertext pair (CT_1, CT_2) from (PT_1, PT_2) and (CT_1', CT_2') from (PT_1', PT_2'). Next store the real ciphertext pair (CT_1, CT_2) with label 1 and the random ciphertext pair (CT_1', CT_2') with label 0. We also store the corresponding ciphertext differences along with the ciphertexts. Repeat this process for random keys for each iteration to generate a training dataset $DATA_T$ or validation dataset $DATA_V$.

Algorithm 3 explains the training process. The input for training is the training data $DATA_T$, and the output is the training accuracy AC_T. We first construct the DL model DL_{Δ_i} and train applying $DATA_T$. Check the training accuracy

AC_T. If AC_T is less than 50% distinguisher is not found for the current dataset else, create the validation data $DATA_V$ and call the validation algorithm.

Algorithm 4 describes the validation process after training. The input for the validation algorithm is validation data $DATA_V$, and the output is validation accuracy AC_V. During validation, load the pre-trained model DL_{Δ_i} and run this for validation data $DATA_V$. If validation accuracy AC_V is greater than 50% distinguisher found for $DATA_T$ and $DATA_V$ corresponding to input difference Δ_i. Save the model DL_{Δ_i}. But if AC_V is less than 50%, distinguisher is not possible.

Table 3: Performance of DL based distinguishers for cipher RC5

DL Technique	No of Rounds	AC_T	AC_V	TPR	TNR
CNN	2	81.48	81.36	0.629	0.998
	3	62.96	58.64	0.425	0.704
	4	56.46	51.2	0.513	0.508
	5	59.14	50.33	0.489	0.509
	6	56.30	50.16	0.499	0.504
	7	53.61	50.18	0.463	0.533
LGBM	2	75.48	74.14	0.558	0.922
	3	62.18	54.02	0.290	0.788
	4	64.57	50.02	0.573	0.427
	5	64.95	50.00	0.496	0.498
	6	64.73	50.15	0.585	0.416
LSTM	2	68.30	67.48	0.518	0.833
	3	50.80	51.30	0.324	0.695
	4	50.35	51.33	0.390	0.609

We use the concept of a generic neural classifier proposed by Pal et al. [12]. We search an x round differential trail for the given cipher and use the corresponding output difference as an input difference for the neural classifier. In this context, if the classifier achieves good accuracy up to y rounds, then we found a new neural classifier with $(x + y)$ rounds.

4 Experiments and Observations

We use CNN, LGBM, and LSTM models to generate deep learning-based distinguishers. For all our experiments, we use google colab with installed GPU (Nvidia T4, 16GB of memory, clock size 1.59GHz). For checking each distinguisher we create a total of 10^5 data elements, out of which 50000 are applied for training, and the rest for validation.

During the training of the models setting up the value of the batch size and number of epochs was a challenging task for us to get the appropriate validation accuracy. We use batch size 30, and the number of epochs is set to nine for both CNN and LSTM models.

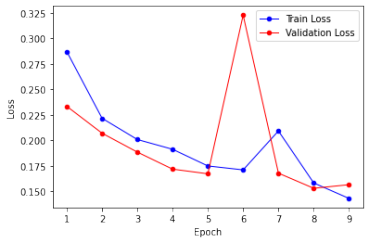

Fig. 3: Epoch Vs Loss for PRIDE (CNN)

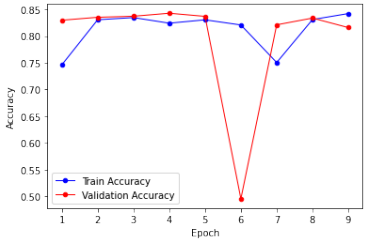

Fig. 4: Epoch Vs Accuracy for PRIDE (CNN)

4.1 PRIDE

Zaho et al. [18] reports differential characteristics of 15 and 18 for PRIDE. We take the input difference of 18 round distinguisher (08000000) as input to our neural classifier. We achieve good accuracy in up to six rounds, which results in a distinguisher for 23 rounds. The training and validation accuracy with the true positive rate(TPR) and true negative rate(TNR) of these distinguishers for different models are enlisted in Table 2. We are getting distinguishers up to six rounds for CNN, up to five rounds for LGBM, and up to four rounds for LSTM. The change of training and validation accuracy with increasing epochs for CNN and LSTM is depicted in Figure 4 and 6. The change of training and validation loss with increasing epochs for CNN and LSTM is depicted in Figure 3 and 5.

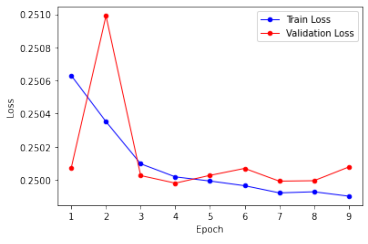

Fig. 5: Epoch Vs Loss for PRIDE (LSTM)

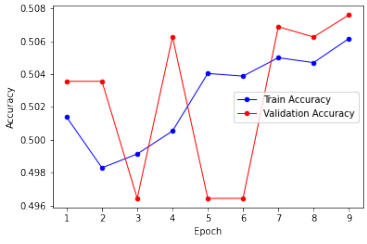

Fig. 6: Epoch Vs Accuracy for PRIDE (LSTM)

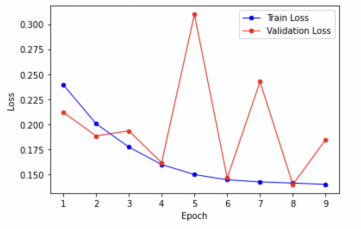

Fig. 7: Epoch Vs Loss for RC5 (CNN)

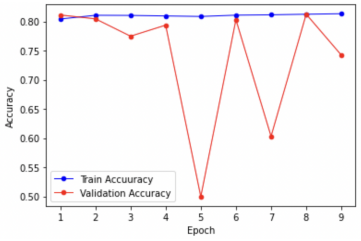

Fig. 8: Epoch Vs accuracy for RC5 (CNN)

4.2 RC5

Birukov et al.[6] propose eight rounds of differential characteristics for RC5. We take the input difference of third round distinguisher (0000080000000000) as input to our neural classifier and achieve good accuracy up to seven rounds, which provides a distinguisher for nine rounds. The training and validation accuracy with the corresponding TPR and TNR of these distinguishers for different models are provided in Table 3. We are getting distinguishers up to seven rounds for CNN, up to six rounds for LGBM, and up to four rounds for LSTM. The training and validation accuracy change with increasing epochs for CNN and LSTM is depicted in Figure 8 and 10. The change of training and validation loss with increasing epochs for CNN and LSTM is depicted in Figure 7 and 9.

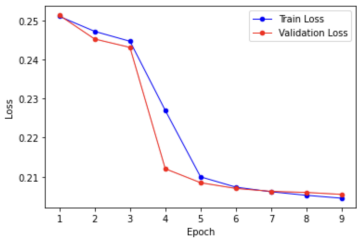

Fig. 9: Epoch Vs Loss for RC5 (LSTM)

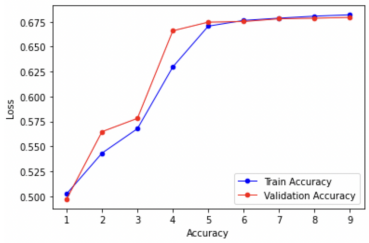

Fig. 10: Epoch Vs Accuracy for RC5 (LSTM)

5 Conclusion

This paper reports the deep learning-based differential classifiers for PRIDE and RC5. To our knowledge, this is the first deep learning-based distinguisher for PRIDE and RC5. We have applied three deep learning models: CNN, LGBM, and LSTM, for searching neural classifiers. For PRIDE, the classifier works up to 23 rounds; for RC5, it works up to nine rounds, which is better than the existing reported results. CNN performs better with accuracy and distinguishers (with higher rounds) when compared with LGBM and LSTM. In future work, we

want to apply partial key recovery attacks for PRIDE and RC5. One can use the techniques for searching neural classifiers to other lightweight block ciphers.

References

1. Albrecht, M.R., Driessen, B., Kavun, E.B., Leander, G., Paar, C., Yalçin, T.: Block ciphers - focus on the linear layer (feat. PRIDE). In: Garay, J.A., Gennaro, R. (eds.) Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I. Lecture Notes in Computer Science, vol. 8616, pp. 57–76. Springer (2014). https://doi.org/10.1007/978-3-662-44371-2_4, https://doi.org/10.1007/978-3-662-44371-2_4
2. 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
3. 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
4. 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
5. 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
6. Biryukov, A., Kushilevitz, E.: Improved cryptanalysis of RC5. In: Nyberg, K. (ed.) Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding. Lecture Notes in Computer Science, vol. 1403, pp. 85–99. Springer (1998). https://doi.org/10.1007/BFb0054119, https://doi.org/10.1007/BFb0054119
7. 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
8. Gohr, A., Leander, G., Neumann, P.: An assessment of differential-neural distinguishers. Cryptology ePrint Archive, Paper 2022/1521 (2022), https://eprint.iacr.org/2022/1521, https://eprint.iacr.org/2022/1521
9. Hou, Z., Ren, J., Chen, S.: Cryptanalysis of round-reduced simon32 based on deep learning. Cryptology ePrint Archive, Paper 2021/362 (2021), https://eprint.iacr.org/2021/362, https://eprint.iacr.org/2021/362

10. Kaliski, B.S., Yin, Y.L.: On differential and linear cryptanalysis of the rc5 encryption algorithm. In: Coppersmith, D. (ed.) Advances in Cryptology — CRYPT0' 95. pp. 171–184. Springer Berlin Heidelberg, Berlin, Heidelberg (1995)
11. Knudsen, L.R., Meier, W.: Differential cryptanalysis of RC5. Eur. Trans. Telecommun. **8**(5), 445–454 (1997). https://doi.org/10.1002/ett.4460080503, https://doi.org/10.1002/ett.4460080503
12. Pal, D., Mandal, U., Chaudhury, M., Das, A., Chowdhury, D.R.: A deep neural differential distinguisher for ARX based block cipher. IACR Cryptol. ePrint Arch. p. 1195 (2022), https://eprint.iacr.org/2022/1195
13. Rivest, R.L.: Cryptography and machine learning. In: Imai, H., Rivest, R.L., Matsumoto, T. (eds.) Advances in Cryptology - ASIACRYPT '91, International Conference on the Theory and Applications of Cryptology, Fujiyoshida, Japan, November 11-14, 1991, Proceedings. Lecture Notes in Computer Science, vol. 739, pp. 427–439. Springer (1991). https://doi.org/10.1007/3-540-57332-1_36, https://doi.org/10.1007/3-540-57332-1_36
14. Rivest, R.L.: The RC5 encryption algorithm. In: Preneel, B. (ed.) Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings. Lecture Notes in Computer Science, vol. 1008, pp. 86–96. Springer (1994). https://doi.org/10.1007/3-540-60590-8_7, https://doi.org/10.1007/3-540-60590-8_7
15. 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
16. Yang, Q., Hu, L., Sun, S., Qiao, K., Song, L., Shan, J., Ma, X.: Improved differential analysis of block cipher PRIDE. In: López, J., Wu, Y. (eds.) Information Security Practice and Experience - 11th International Conference, ISPEC 2015, Beijing, China, May 5-8, 2015. Proceedings. Lecture Notes in Computer Science, vol. 9065, pp. 209–219. Springer (2015). https://doi.org/10.1007/978-3-319-17533-1_15, https://doi.org/10.1007/978-3-319-17533-1_15
17. Zhang, L., Wang, Z.: Improving differential-neural distinguisher model for des, chaskey, and PRESENT. CoRR **abs/2204.06341** (2022). https://doi.org/10.48550/arXiv.2204.06341, https://doi.org/10.48550/arXiv.2204.06341
18. Zhao, J., Wang, X., Wang, M., Dong, X.: Differential analysis on block cipher PRIDE. IACR Cryptol. ePrint Arch. p. 525 (2014), http://eprint.iacr.org/2014/525