Deep Learning based Cryptanalysis of Lightweight Block Ciphers, Revisited

Hyunji Kim, Sejin Lim, Yeajun Kang,
Wonwoong Kim, Hwajeong Seo[0000−0003−0069−9061]

IT Department, Hansung University, Seoul, South Korea,
{khj1594012, dlatpwls834, etus1211, dnjsdndeee, hwajeong84}@gmail.com

Abstract. Cryptanalysis is to infer the secret key of cryptography algorithm. There are brute-force attack, differential attack, linear attack, and chosen plaintext attack. With the development of artificial intelligence, deep learning-based cryptanalysis has been actively studied. There are works in which known-plaintext attacks against lightweight block ciphers, such as S-DES, have been performed.
In this paper, we propose a cryptanalysis method based on the-state-of-art deep learning technologies (e.g. residual connections and gated linear units) for lightweight block ciphers (e.g. S-DES and S-AES). The number of parameters required for training is significantly reduced by 93.16 % and the average of bit accuracy probability increased by about 5.3 %, compared with previous work.

Keywords: Cryptanalysis · Deep Learning · Lightweight Block Ciphers.

1 Introduction

Cryptanalysis for block ciphers has been studied and is still receiving high attention. Cryptanalysis is an attack that infers key of cryptographic algorithms. In cryptanalysis, there are various methods, such as ciphertext only attack, chosen-plaintext attack, chosen-ciphertext attack, known-plaintext attack, differential analysis, and side-channel analysis. A ciphertext-only attack is a technique that decrypts the ciphertext by examining the statistical characteristics of the ciphertext or performing a brute-force attack while the attacker has only the ciphertext. The chosen-plaintext attack is performed in a state, where an attacker can encrypt a large number of random plaintext. In other words, the key can be inferred by comparing the ciphertexts generated by encrypting the random plaintext. The known-plaintext attack is a cryptanalysis technique that uses a number of known plaintext and ciphertext pairs to infer key. Linear attack is a cryptanalysis technique developed by [1] in 1993. This is a known-plaintext attack, and it is a method of finding a key by linearizing the non-linear structure inside the cryptographic algorithm. Differential analysis [2] is an attack technique that can be used in chosen-plaintext attacks. After the plaintext is divided into smaller units, substitution and permutation are repeatedly applied

for security. In the case of the substitution process, it is non-linear. It is difficult to find out the key unless a brute-force attack. If the substitution is not designed with high security level, the key can be inferred through differential analysis using the property that the differential is maintained. There are variants of differential analysis, such as higher-order differential cryptanalysis and truncated differential cryptanalysis.

In this paper, we present deep learning-based cryptanalysis for lightweight block ciphers including Simplified DES (S-DES) and Simplified AES (S-AES). In the case of cryptanalysis for S-DES, it was firstly performed in [3] but it has still room to improve with the-state-of-art deep learning techniques. We performed the attack with fewer parameters ensuring higher accuracy by applying the-state-of-art deep learning technology. Furthermore, to the best of our knowledge this work is the first known-plaintext attack based on deep learning for S-AES.

1.1 Contribution

Designing artificial neural networks considering the characteristics of cryptographic algorithms In order to design an optimal artificial neural network for cryptanalysis, we constructed a neural network considering the characteristics of cryptographic algorithm. Various types and options of neural networks were tested, and we selected the neural network structure with the best performance.

Improving performance for S-DES cryptanalysis We applied the-state-of-art artificial neural network techniques that were not applied in the previous work for deep learning-based cryptanalysis. The neural network implemented in our work achieved 5.3 % higher accuracy with 93.16 % fewer parameters compared to previous work in cryptanalysis for S-DES.

The first cryptanalysis based on deep learning for S-AES We are the first to attempt a known-plaintext attack against S-AES, to the best of our knowledge. We confirmed that cryptanalysis based on deep-learning up-to 12-bit key space is possible. Finally we compared result of cryptanalysis for S-DES and S-AES.

The remainder of this paper is organized as follows. In Section 2, related technologies, such as artificial neural network, deep learning-based cryptanalysis, and previous work, are presented. In Section 3, the proposed cryptanalysis based on artificial neural network is introduced. In Section 4, the evaluation of our deep learning-based advanced cryptanalysis technique implementation is discussed. Finally, Section 5 concludes the paper.

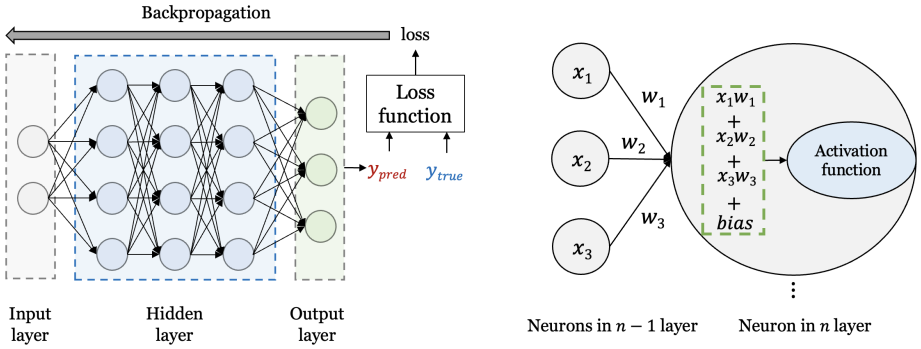

Fig. 1: Architecture of artificial neural networks.

2 Related Works

2.1 Artificial Neural Network

Artificial neural networks are learning algorithms inspired by neural networks in biology. As shown in Figure 1, a neural network is constructed in the form of stacked layers of multiple nodes. As shown on the right of the Figure 1, neurons (i.e. nodes) in each layer perform a weighted sum operation using the node values (x) and weights (w) of the previous layer connected to them, and add a bias. Then, it is input to the non-linear activation function, and computed as a single value. In this way, the loss value is obtained after passing through all the layers. Then, the weights inside the neural network are updated to minimize the loss through the backpropagation process. By repeating this process, a neural network that guarantees generalization performance for untrained data is constructed. When the trained model is used for actual inference, the inference proceeds by inputting data with the weights of the fixed neural network. Through this, it is possible to learn, classify, and predict by extracting features of input data (e.g. image, time-series, language, and graph).

Residual Connection in Neural Network Figure 2 shows residual connection (i.e. skip connection) [4] in artificial neural network. As shown in the Figure 2, a residual connection has a structure in which the output of the previous layer is added to the input of the next layer after skipping several layers. This structure allows us data to go deeper into the neural network by following skip connections instead of following the main path where data flows by default. In other words, the skip connection means skipping layers to propagate information to a deeper layer. If the network gets deeper, the gradient may vanish as the gradient converges to 0 as it goes to the input layer in the backpropagation process. The residual connection can solve this gradient vanishing problem. Larger gradients can be propagated to the initial layer, and the initial layers are trained as fast as the output layer. This structure allows deeper networks to be trained.

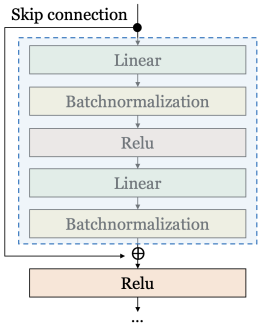

Fig. 2: Skip connection for residual network.

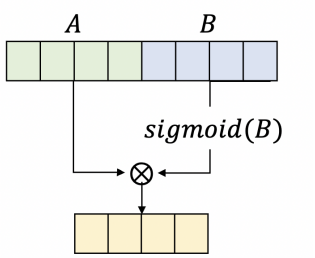

Fig. 3: Gated Linear Units(GLU).

Additional information can be learned and the overfitting of a neural networks can be prevented.

Gated Linear Units Figure 3 shows gated linear units (GLU) [5]. GLU controls the input data like a gate. In each layer, a matrix multiplication operation of input data and weights is performed, and then it is input to the activation function. In the case of GLU, A and B are constructed as shown in Figure 3 before input to the activation function. After that, the B part is input to the sigmoid activation function. Since the sigmoid function is a bipolar function, important data survives in the multiplication process. Conversely, insignificant data with a generally small value has smaller values. Finally, a point-wise multiplication on A and B is performed. In other words, GLU allows us to focus on more important information and enables faster and more stable training.

2.2 Artificial Neural Network based Cryptanalysis

In the artificial neural network-based cryptanalysis, known-plaintext attacks, ciphertext only attacks, and differential attacks against Caesar cipher, Vigenere cipher, Simplified-DES, round-reduced SPECK, and round-reduced SIMON were

Table 1: Comparison of cryptanalysis methods based on deep learning.

Category	Known Plaintext Attack	Differential Attack
Target cipher	S-DES, SPECK and SIMON[3]	Round-reduced SPECK[6], SIMON[7], PRESENT[8], GIMLI, ASCON, KNOT, CHASKEY (round-reduced)[9]
Descriptions	Successful for text-based key (Successful for random bit keys only in S-DES)	Impossible to attack full-round cipher

mainly performed. In addition, the work of predicting the number of active S-boxes required for cryptanalysis was also studied. It is possible for cryptanalysis, such as classical ciphers, S-DES, SPECK, SIMON, and PRESENT. Table 1 shows research trends for deep learning-based cryptanalysis. There are two types of cryptanalysis using artificial neural networks (i.e. known-plaintext attack and differential attack).

Known Plaintext Attack For known-plaintext attacks, S-DES, SPECK, and SIMON block cipher algorithms are targeted [3]. They performed cryptanalysis on text-based keys and random bit keys. However, in both cases, only S-DES could be attacked, and SPECK and SIMON could only analyze text-based keys.

Differential Cryptanalysis Research on differential attacks is receiving high attention. In [6], differential analysis for round-reduced SPECK is presented for the first time at Crypto2019. They proposed a neural network distinguisher with performance exceeding the existing differential distinguisher for SPECK32/64 in round 11. They showed that artificial neural networks can be applied significantly in differential attacks. In order to utilize the differential properties of round reduced SPECK, the proposed method trains to distinguish between a ciphertext pair and a random pair for a given input difference. It is a method of determining whether the decryption result using the guessed key is an actual ciphertext or a randomly generated value. If the result of classification is ciphertext, the corresponding key is determined as the correct key. However, this work raised the question of the interpretability of the artificial neural network, a black-box model. The interpretation of [6] was published in Eurocrypt2021 [10]. The artificial neural network-based differential analysis method is in progress for ciphers, such as SIMON [7] based on [6]. Inspired by the research results of [6], [9] proposed an artificial neural network-based cryptanalysis of the all-in-one differentials for non-Markov ciphers through machine learning. The cryptographic algorithms analyzed are GIMLI, ASCON, KNOT, and CHASKEY, all of which are round-reduced versions.

Most of the currently performed artificial neural network-based differential cryptanalysis works correspond to lightweight ciphers and round-reduced ciphers. In the future, research on full-round attacks and cryptographic interpreta-

tions should be conducted. Researches to improve the performance of cryptanalysis as well as minimize the structure of artificial neural networks while guaranteeing similar performances are performed. In the future, there are opinions that the application of explainable artificial intelligence (i.e. XAI) to interpret the results derived from the artificial neural network from a cryptographic point of view. And cryptanalysis attacks should be performed in non-lightweight ciphers and in full rounds. In addition, artificial neural networks can be applied to cryptanalysis in various aspects, such as research on predicting the number of active S-boxes [11].

2.3 Previous Work

As mentioned above, known-plaintext attacks for S-DES, Speck, and Simon have been proposed in [3]. In this approach, plaintext and ciphertext pairs are expressed as bits, concatenated, and then input into a neural network. Then, the neural network predicts the key corresponding to the pair of plaintext and ciphertext by comparing it with the real key. Finally it calculates the loss through the MSE loss function. Training is carried out to minimize the loss to predict the correct key corresponding to the pair of plaintext and ciphertext. The weights of the neural network trained to achieve sufficient performance through this process are fixed. In the inference phase, the key can be predicted using a trained neural network. The number of pairs of plaintext and ciphertext used for training and validation of S-DES is 50,000 and 10,000, respectively. The number of pairs of plaintext and ciphertext used in Speck and Simon's training and validation is 500,000 and 1 million. In their experiment, they used the random bit key and text key. The random bit key has the same probability of occurrence of all bits, the key space for n-bit is 2^n, and the text-based key uses only 64 ASCII codes. The probability of occurrence of all bits is not the same. That is, the text key is easier to predict because text-based keys have a different probability of occurrence for each bit, and the key space is smaller than for random bit keys. Bit Accuracy Probability (BAP) and Deviation were used to evaluate the performance of the previous work. If the probability of occurrence for each bit is different, it is easier to predict. For example, if the first bit is 1 with a probability of 1.0, the neural network can predict the first bit as 1 without difficulty. In other words, the difference between the BAP and the probability of occurrence is calculated to fairly evaluate the performance when the probability of occurrence of the key for each bit is different. The difference between the two values is calculated, and if the value is a positive number, it is determined that cryptanalysis is possible. In the case of the random bit key, since the key has the same probability of occurrence for all bits, the deviation is the value obtained by subtracting 0.5 from BAP.

As a result of the experiment, in the case of S-DES, cryptanalysis was possible for both the random bit key and the text key. In addition, the 5-th and 8-th bits in the random bit key and the text key were vulnerable to attack. And the 6-th bit was relatively safe in cryptanalysis. Next, Speck and Simon achieved an average prediction probability of 0.5 for the case of using a random bit key, and

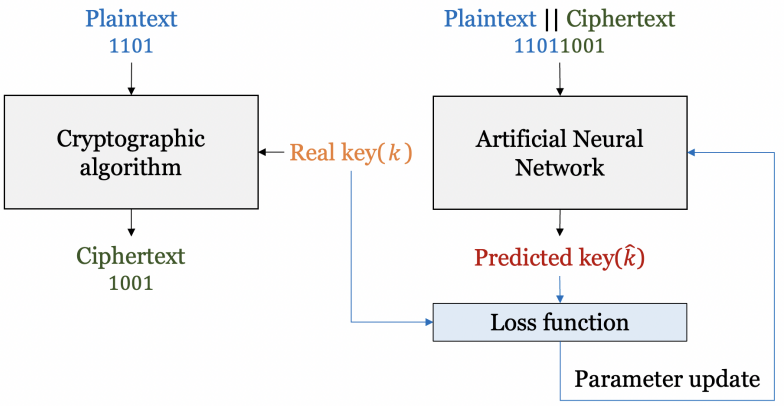

Fig. 4: System diagram for proposed method.

there were bits with a negative deviation. Cryptanalysis is failed for random bit keys and succeeded for text keys (keyspace is 2^{48} and key occurrences are different).

In this paper, the performance of cryptanalysis is improved by applying the-state-of-art artificial neural network structure and technology with well selected parameters.

3 Proposed Method

In this paper, we propose an advanced cryptanalysis technique based on deep learning for S-DES. Several deep learning technologies that can improve performance compared to previous work was applied. We also firstly perform cryptanalysis for S-AES. Figure 4 shows the system diagram for proposed method. First, a cryptographic algorithm is used to obtain a pair of ciphertexts and plaintext. The real key used is called k. The real key k is used as a label for the data to be trained on. Next, we concatenate plaintext and ciphertext, which are then input to an artificial neural network for supervised learning. The neural network learns characteristics of the input data. It can predict the correct label. The output of the neural network is $\hat{k}$, which is the predicted key, and it is input to the loss function to compare with the real key k. As the real key and the predicted key become similar, the loss becomes minimized, and the neural network updates the weight to minimize the loss. The neural network are trained by repeating this process to make correct predictions.

3.1 Data Generation

Figure 5 shows the data set for training and test, and Table 2 shows details of data set for S-DES and S-AES. The data type is bits. Plaintext and ciphertext

Table 2: Details of dataset.

Algorithm	N_{tr}	N_{val}	N_{ts}	$m(bit), l(bit)$	$Rounds$
S-DES	55,000	30,000	15,000	8, 10	2
S-AES	900,000	500,000	200,000	16, 16	2

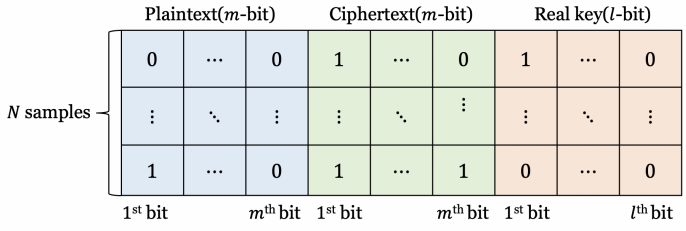

Fig. 5: Dataset.

pairs and key are expressed as bits. When saving these as a csv file format for training, one bit is inputted per one column. When the length of the plaintext is m, the data has a length of $2m$, and when input to the neural network, one bit is assigned to each neuron. The key bit of l-bit is used as label. In other words, it is not a classifier classified as a value from 0 to $2^l - 1$, but predicting each bit of the key.

We sampled plaintext and keys, randomly. In the case of S-DES, an 8-bit plaintext and a 10-bit key are randomly selected and then encrypted to make a data set. S-AES randomly samples 16-bit plaintext and 16-bit key, then encrypts it to form a dataset. The probability of occurrence of all bits is the same, because we generate the random number in the range of 2^n. Let the number of dataset for training be N_{tr}, the number of dataset for validation as N_{val}, and the number of datasets for testing as N_{ts}. The number of data is 100,000 for S-DES, and the number of data is about 1.6 million for S-AES. In addition, S-AES has a longer plaintext and a longer key length than S-DES, and requires a large number of data.

3.2 Neural Network Structure for Cryptanalysis

There are various types of neural networks that can be used for cryptanalysis, such as Fully-Connected Neural Networks (FCNN), Convolution Neural Networks (CNN), and Recurrent Neural Networks (RNN). Among them, we use a fully-connected neural network to design an effective artificial neural network for cryptanalysis. The reason has to do with the properties of a cryptographic algorithm. An cryptographic algorithm has a property that most or all bits are affected when a single bit is changed, and half of the ciphertext is statistically changed when a single bit of the plaintext is changed [12]. In other words, the first bit of the plaintext can affect all bit of the ciphertext. Therefore, it is difficult for data for learning to have locality in which adjacent features have similar

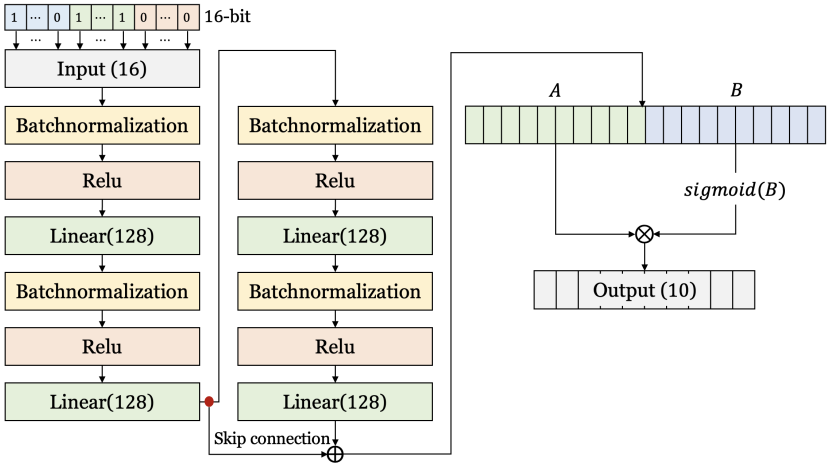

Fig. 6: Neural network structure for cryptanalysis for S-DES.

values, and it is not temporal data having time information. Therefore, instead of CNN and RNN, which are effective for training data with temporal locality, a fully-connected network suitable for considering global information of data is used.

Structure of Neural Network for Cryptanalysis for S-DES Figure 6 is neural network structure for S-DES cryptanalysis. We designed a neural network by applying the residual connection and GLU described above. Input data is 16-bit, because each bit of the data set concatenated 8-bit plaintext and 8-bit ciphertext. The number of neurons in the input layer was set to 16. That is, each bit is input to each neuron. Also, the same number of neurons was used in all hidden layers to minimize information loss. Finally, it goes through GLU which enables stable learning by controlling the information. The number of neurons in the output layer is 10 equal to the number of key bits. That is, each neuron in the output layer predicts each bit of the key.

Structure of Neural Network for Cryptanalysis for S-AES Figure 7 shows the structure of neural network for cryptanalysis for S-AES. For S-AES, we set the number of input neurons to be the same as the number of bits of input data in the same way as S-DES. In addition, 5 residual blocks and 1 GLU are applied, and the number of neurons in each hidden layer is larger than that of S-DES. In other words, it has a structure similar to the neural network used for S-DES, but it uses a deeper and larger neural network.

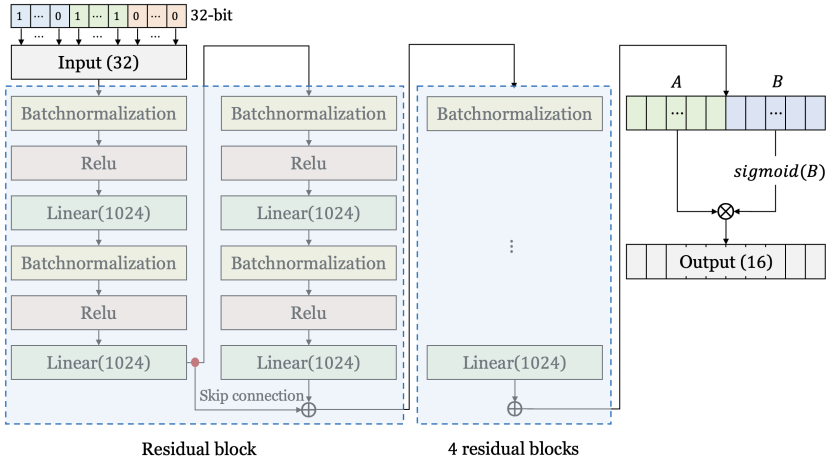

Fig. 7: Neural network structure for cryptanalysis for S-AES.

3.3 Training and Testing

Training Training and testing are performed using a neural network designed for cryptanalysis and the prepared data. First, for training, a training data set is input to a neural network, and the neural network outputs predicted values. The output value and the real key are input to the loss function, and the loss representing the difference between the two value s (predicted value ($\hat{k}_{i,j}$) of the $j-th$ bit of the $i-th$ and real key (k)) is calculated. The loss function used is Mean Squared Error (MSE), which is shown in Equation 1. N is the number of data samples (N_{tr} or N_{val}), and L is the number of key bits. In order to minimize the loss, the training is performed while repeating the process of updating the weights of the neural network. If the training is not performed properly, the neural network will output a predicted value of about 0.5 or an incorrect value. If the training is performed properly, the neural network will predict the output values similar to the real key value.

$$\frac{1}{N \cdot L} \cdot \sum_{i=1}^{N} \sum_{l=1}^{L} \left(\hat{k}_{i,l} - k_{i,l}\right)^2 \tag{1}$$

Testing In the inference phase, a trained neural network that has the fixed weights is used. This neural network outputs predicted values ($\hat{k}$) when inputting test data. This value is a real number. We need to compare $\hat{k}$ with the real key (bit) for the test. In other words, $\hat{k}$ must be converted to 0 or 1, because it needs to be compared in the form of a bitstring. For this, the predicted key value ($k_{pred(i,l)}$) is calculated as in Equation 2 using the predicted value ($\hat{k}_{i,l}$) of the $l-th$ bit of the $i-th$ data sample in the test data set. In addition, BAP (the accuracy of each key bit for the entire data set) is calculated using the transformed predicted key and the real key as in Equation 3. If the training is

poorly performed, BAP is less than 0.5. That is, the neural network just guesses the result with one of two. With the proper training, the BAP will be greater than or equal to 0.5. If the result is 0.5 or greater, the neural network can predict the corresponding key bit.

$$k_{pred(i,l)} = \begin{cases} 0 & \hat{k}_{i,l} < 0.5, \\ 1 & \hat{k}_{i,l} \geq 0.5. \end{cases} \tag{2}$$

$$BAP_l = \frac{1}{N_{ts}} \cdot \sum_{i=1}^{N_{ts}} XNOR(k_{pred(i,l)}, k_{(i,l)}) \tag{3}$$

4 Evaluation

4.1 Experiment Environment

For the experiment, Google Co-laboratory PRO PLUS (commercial license), a cloud-based service, was utilized. It ran on Ubuntu 18.04.5 LTS and consists of an Nvidia GPU (Tesla T4) with 50GB RAM. In terms of the programming environment, Python 3.7.13, TensorFlow 2.8.0 and Keras 2.8.0 version were used. Due to the large amount of data and the growing size of the neural network, it took about 10 hours to train one time.

4.2 Experiments on S-DES

Training Result We performed training on S-DES using the data, and the structure of the neural network proposed in Section 3. We only show the results on our best neural network model. The loss function, optimizer, and epoch used for training are as follows.

- **Loss:** We used MSE loss function. As a result of training a neural network to which residual connection is applied, the training loss is 0.1656 and the validation loss achieves 0.1660. The result of training a neural network to both residual connection and GLU are applied. The training loss achieves 0.1774 and the validation loss achieves 0.1767.

- **Optimizer:** We used Adam optimizer. The optimizer is a function that finds the minimum value of the loss function (to minimize the loss). When the optimizer moves toward the minimum, its stride is called the learning rate. The learning rate of the optimizer is set as the learning rate exponential decay method. The range of learning rate from 0.001 to 0.1. Learning rate decay use the large learning rate value at first, and then the value gradually decreases. This allows the neural network to achieve optimal training result faster.
- **Epoch:** As a result of 100 epochs in the network that both techniques are applied, the loss was sufficiently reduced. The network that is applied skip connection requires 150 epochs. In the case of the previous work, 5,000 epochs

Table 3: Comparison with previous work (BAP and the number of parameters for S-DES).

Method	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	6^{th}	7^{th}	8^{th}	9^{th}	10^{th}	Parameters
Previous work [3]	0.64	0.74	0.71	0.58	0.64	0.8	0.54	0.6	0.84	0.8	805,930
This work (Res)	0.72	0.77	0.75	0.6	0.76	0.8	0.59	0.68	0.85	0.83	53,802
This work (Res+GLU)	0.72	0.79	0.77	0.62	0.75	0.81	0.59	0.66	0.87	0.85	55,092

were performed. Since GLU and residual connection technology were added in this work, stable and fast training was possible compared to the basic network.

These results show that the neural network to which GLU is applied can converge more stably and faster than when it is not applied.

Bit Accuracy Probability We performed inference using a trained neural network. Table 3 shows the result of inference for cryptanalysis. For comparison of the number of parameters with the previous work, the neural network described in [3] was implemented identically, and similar performance was obtained. However, for the BAP values excluding the number of parameters, the values written in the paper were used.

The 7-th bit achieves less than 60 % accuracy. They are relatively safe for cryptanalysis. However, the 6-th, 9-th, and 10-th bits exceed 80 % in accuracy. These bits are vulnerable to cryptanalysis attacks. A similar pattern to ours can be seen in previous work as well. The proposed method with both residual connection and GLU achieved higher overall accuracy. As a result of calculating the average over all bits, the accuracy is 5.3 % higher. A slightly higher BAP was achieved than previous works when the only residual connection was applied, and the BAP for the 4-th bit exceeded 60 %. Therefore, we can see that the neural network to which both techniques are applied is more effective for cryptanalysis.

Finally, with our neural network applying both techniques, we reduced the number of parameters by 93.16 % compared to the previous work. As with the neural networks in the previous work, all layers of the neural network are fully connected layers. However, these are the results obtained by reducing the number of neurons in each layer from 512 to 128 and applying residual connection and GLU. In other words, by applying this new deep learning technique, it was possible to achieve higher accuracy with a smaller neural network.

4.3 Experiments on S-AES

Training Result S-AES data was learned using the neural network structure proposed in Section 3. The loss function, optimizer, and epoch are as follows.

– **Parameters**: Cryptanalysis for S-AES requires much larger parameters than S-DES. The number of parameters for S-AES with 12-bit key space

Table 4: BAP for S-AES (9∼12-bit key space).

Key	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	6^{th}	7^{th}	8^{th}	9^{th}	10^{th}	11^{st}	12^{nd}	13^{rd}	14^{th}	15^{th}	16^{th}
9-bit	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.7	0.7	0.69	0.69	0.7	0.69	0.69	0.7	0.69
10-bit	1.0	1.0	1.0	1.0	1.0	1.0	0.63	0.63	0.63	0.64	0.63	0.63	0.6	0.6	0.6	0.6
11-bit	1.0	1.0	1.0	1.0	1.0	0.52	0.53	0.52	0.53	0.52	0.52	0.53	0.52	0.51	0.52	0.52
12-bit	1.0	1.0	1.0	1.0	0.51	0.51	0.5	0.5	0.5	0.5	0.5	0.51	0.5	0.5	0.5	0.5

is 11,636,832.

- **Loss**: In S-AES, the MSE loss function is also used. As a result of training, a training loss of 0.1826 and a validation loss of 0.1923 were achieved.
- **Optimizer**: We use Adam optimizer, and it was also set to the learning rate exponential decay method (Range of learning rate is 0.001 to 0.1).
- **Epoch**: The epoch was set to 150. There was no decrease in loss even after training more than 150 epochs.

Bit Accuracy Probability Table 4 shows the result of crytanalysis for S-AES. We trained from 9-bit to 12-bit key spaces. For proper training, the capacity of the neural network must be large enough according to the complexity of the data. In case of the 13-bit key space, even if the network is scaled as large as possible to accommodate the increased data complexity, training is hardly performed. In addition, training on very large neural networks and large dataset was not possible due to the constraints of the environment. This shows that deep learning-based cryptanalysis requires large data set, GPU, and memory environments. Through this experiment, we show that cryptanalysis of S-AES is possible up to a 12-bit key space. In the case of S-AES, unlike S-DES, the BAP of all bits is a similar value. When the key is increased by 1-bit, the accuracy tends to decrease by about 10 % even if the capacity of the network sufficiently increases.

4.4 Comparison of Cryptanalysis for S-DES and S-AES

We obtained the following results through the experiments described above. First, S-DES has a specific vulnerability pattern compared to S-AES. In addition, S-AES has similar accuracy for all bits. As shown in Figure 8, the number of parameters required for cryptanalysis for S-AES is much larger and increases, significantly. In S-DES, when the key space increases by 1-bit, the number of parameters increases by about 1.5 times, but in case of S-AES, it increases by about 2 to 3 times. Considering that the accuracy is lower and network requires much more parameters than that of S-DES when using the same key space as S-DES, it can be seen that S-AES is more difficult for neural networks to

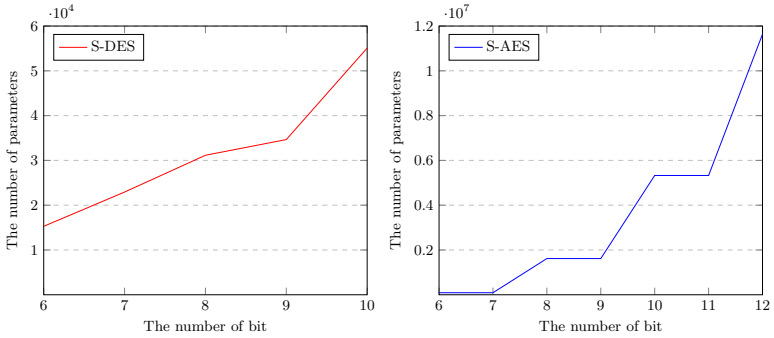

Fig. 8: Parameters of neural network for cryptanalysis (left: S-DES, right: S-AES).

learn and is designed to be more secure than S-DES. S-DES consists of initial permutation, expansion/permutation, key addition, s-box (i.e. substitution), and swap operations. In addition, S-AES is composed of substitution nibbles, shift rows, mixcolumns, and key addition, and the key space is larger than that of S-DES. This result seems to have been derived because the S-box of S-AES is designed to be more confused than S-DES, so the relationship between the key and the ciphertext is not well revealed, and diffusion is better achieved through mixcolumns and permutation. That is, S-DES uses less key space and has poor diffusion and confusion characteristics than S-AES. Some key bits can be inferred.

Finally, many parameters are required for cryptanalysis of the Simplified cipher as shown in Figure **??** and many resources are required for training. Therefore, there is a limit to analyzing cryptographic algorithms such as AES with the current deep learning-based technology, and it can be seen that text key-based cryptanalysis is possible as in the previous work.

5 Conclusion

In this paper, we proposed an improved deep learning-based cryptanalysis. In this improved model, skip connection and gated linear units are applied to the basic neural network structure, enabling more stable learning. As a result, 5.3 % higher accuracy was achieved on average and the number of parameters was reduced by 93.16 % compared to previous work in S-DES. We also performed deep learning-based cryptanalysis for S-AES for the first time. In future work, we will apply deep learning based cryptanalysis for other block ciphers.

References

1. M. Matsui, "Linear cryptanalysis method for DES cipher," in *Workshop on the Theory and Application of of Cryptographic Techniques*, pp. 386–397, Springer, 1993. 1

2. E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," *Journal of CRYPTOLOGY*, vol. 4, no. 1, pp. 3–72, 1991. 1
3. J. So, "Deep learning-based cryptanalysis of lightweight block ciphers," *Security and Communication Networks*, vol. 2020, 2020. 2, 5, 6, 12
4. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. 3
5. Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *International conference on machine learning*, pp. 933–941, PMLR, 2017. 4
6. A. Gohr, "Improving attacks on round-reduced Speck32/64 using deep learning," in *Annual International Cryptology Conference*, pp. 150–179, Springer, 2019. 5
7. Z. Hou, J. Ren, and S. Chen, "Cryptanalysis of round-reduced Simon32 based on deep learning," *Cryptology ePrint Archive*, 2021. 5
8. A. Jain, V. Kohli, and G. Mishra, "Deep learning based differential distinguisher for lightweight cipher PRESENT," *Cryptology ePrint Archive*, 2020. 5
9. A. Baksi, "Machine learning-assisted differential distinguishers for lightweight ciphers," in *Classical and Physical Security of Symmetric Key Cryptographic Algorithms*, pp. 141–162, Springer, 2022. 5
10. A. Benamira, D. Gerault, T. Peyrin, and Q. Q. Tan, "A deeper look at machine learning-based cryptanalysis," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 805–835, Springer, 2021. 5
11. M. F. Idris, J. S. Teh, J. L. S. Yan, and W.-Z. Yeoh, "A deep learning approach for active s-box prediction of lightweight generalized feistel block ciphers," *IEEE Access*, vol. 9, pp. 104205–104216, 2021. 6
12. C. E. Shannon, "Communication theory of secrecy systems," *The Bell system technical journal*, vol. 28, no. 4, pp. 656–715, 1949. 8