

Deep Learning based Cryptanalysis of Lightweight Block Ciphers, Revisited

Hyunji Kim¹, Sejin Lim¹, Yeajun Kang¹,
Wonwoong Kim¹, and Hwajeong Seo¹

Hansung University, Seoul, South Korea

khj1594012@gmail.com, dlatpwls834@gmail.com, etus1211@gmail.com,
dnjsdndeee@gmail.com, hwajeong84@gmail.com

Abstract. Cryptanalysis is to infer the secret key of a cryptography algorithm. There are brute-force attacks, differential attacks, and linear attacks. 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 state-of-art deep learning technologies (e.g. residual connections and gated linear units) for lightweight block ciphers (e.g. S-DES, S-AES, and S-SPECK). 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 the-state-of-art work. Cryptanalysis for S-AES and S-SPECK was possible up to 12-bit and 6-bit keys, respectively. Through this, we confirmed that deep learning-based key recovery of modern ciphers with the full round and the full key is practically impossible.

Keywords: Cryptanalysis · Deep Learning · Lightweight Block Ciphers · S-DES · S-AES · S-SPECK.

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 [15] 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 [3] 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 a 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), Simplified AES (S-AES), and Simplified SPECK (S-SPECK). In the case of cryptanalysis for S-DES, it was first performed in [22] 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 key recovery attack based on deep learning for S-AES.

1.1 Contribution

The following are the major contributions of this work.

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 the 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 key recovery (random bit key) based on deep learning for S-AES and S-SPECK We are the first to attempt a known-plaintext attack (for random bit key) against S-AES and S-SPECK, to the best of our knowledge. We confirmed that cryptanalysis is possible for S-AES and S-SPECK up to 12-bit and 6-bit key spaces. Finally, we compared the results of cryptanalysis for S-DES, S-AES, and S-SPECK.

The remainder of this paper is organized as follows. In Section 2, related technologies, such as artificial neural networks, deep learning-based cryptanalysis, and previous work, are presented. In Section 3, the proposed cryptanalysis based on an 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.

2 Related Works

2.1 Artificial Neural Network

Artificial neural networks [9] 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 Figure 1, neurons (i.e. nodes) in each layer perform a multiplication and sum operation using the node values (x) and weights (w) of the previous layer connected to them and add a bias [14]. Then, it is input to the non-linear activation function [6], 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). The type of neural network is appropriately selected according to the characteristics of the input data. There is the most basic network which is Multi-Layer Perceptron (MLP), Convolutional Neural Networks (CNN) [1] which is good for image learning, and Recurrent Neural Network (RNN) [19] which is effective for time series data prediction. There are also Generative Adversarial Networks (GANs) [8], which are used for tasks that generate data, and techniques such as Reinforcement learning [23], which allow the optimal action to be selected based

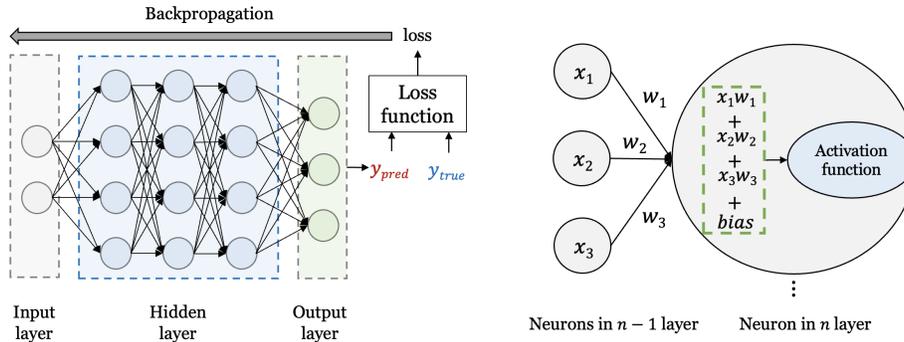


Fig. 1: Architecture of artificial neural networks.

on the concept of a reward. A loss function and an optimization function are used in the training process for an artificial neural network. The loss function calculates the loss, which is the difference between the actual answer and the predicted value. Loss function includes binary cross-entropy, categorical cross-entropy, mean squared error, etc., and are selected according to the problem to be solved (e.g. Binary cross-entropy is used for binary classification). The neural network needs to minimize the loss calculated in this way, and an optimization function is used for this. Optimizer [20](e.g. Stochastic gradient descent (SGD), RMSprop, and Adam) is used to effectively find the minimum.

Residual Connection in Neural Network Figure 2 shows residual connection (i.e. skip connection) [10] in an artificial neural network. As shown in 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. Additional information can be learned and the overfitting of neural networks can be prevented.

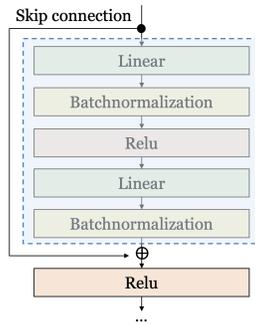


Fig. 2: Skip connection for the residual network.

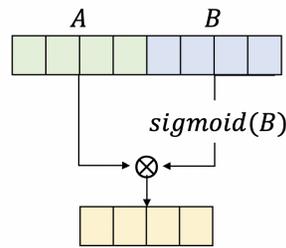


Fig. 3: Gated Linear Units(GLU).

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

The development of artificial intelligence technology could provide a new approach for cryptanalysis [4]. 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 mainly performed [7, 8, 9, 10]. In addition, the work of predicting the number of active S-boxes required for cryptanalysis was also studied [11]. There are mainly two types of cryptanalysis using artificial neural networks (i.e. known-plaintext attack and differential attack). For known-plaintext attacks, S-DES, SPECK, and SIMON block cipher algorithms are targeted [22]. 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. In [13], a known-plaintext attack on Caesar cipher was performed using quantum support vector machine (QSVM) [12] through quantum machine learning. QSVM has a feature map of the existing support vector machine (SVM) [17] designed as a quantum circuit on a quantum computer, and like SVM, it is a machine learning technique that finds the optimal boundary between data points through a hyperplane. They performed cryptanalysis for 2-bit and 3-bit plaintext and ciphertext pairs and keys as resource and memory problems such as qubits required for the quantum circuits.

2.3 Previous Work

As mentioned above, known-plaintext attacks for S-DES, Speck, and Simon have been proposed in [22]. 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 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 try cryptanalysis for S-AES and S-SPECK with the random-bit key. Figure 4 shows the system diagram for the 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 inputted to an artificial neural network for supervised learning. The neural network learns the 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 is trained by repeating this process to make correct predictions.

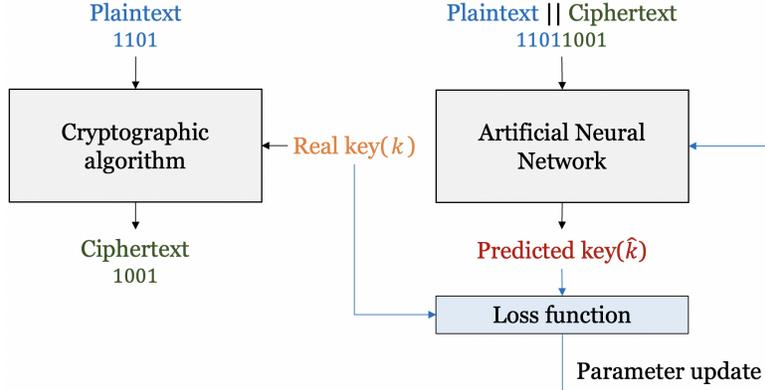


Fig. 4: System diagram for the proposed method.

3.1 Data Generation

Figure 5 shows the data set for training and test, and Table 1 shows details of data set for S-DES [18], S-AES [16], S-SPECK [2]. The data type is bits. Plaintext and ciphertext pairs and keys 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 a 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 block 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. S-SPECK has 32-bit plaintext and a 64-bit key. They are randomly chosen. 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} . In addition, S-AES and S-SPECK have a longer plaintext and a longer key length than S-DES. So, they require a large number of data.

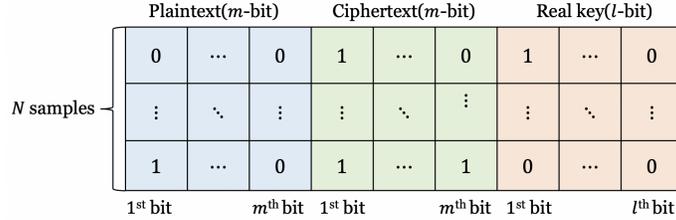


Fig. 5: Dataset.

Table 1: 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
S-SPECK	10,000,000	2,000,000	100,000	32, 64	22

3.2 Neural Network Structure for Cryptanalysis

There are various types of neural networks that can be used for cryptanalysis, such as MLP, CNN, and 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. A 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 [21]. In other words, the first bit of the plaintext can affect all bits of the ciphertext. Therefore, it is difficult for data for learning to have a locality in which adjacent features have similar 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 the 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.

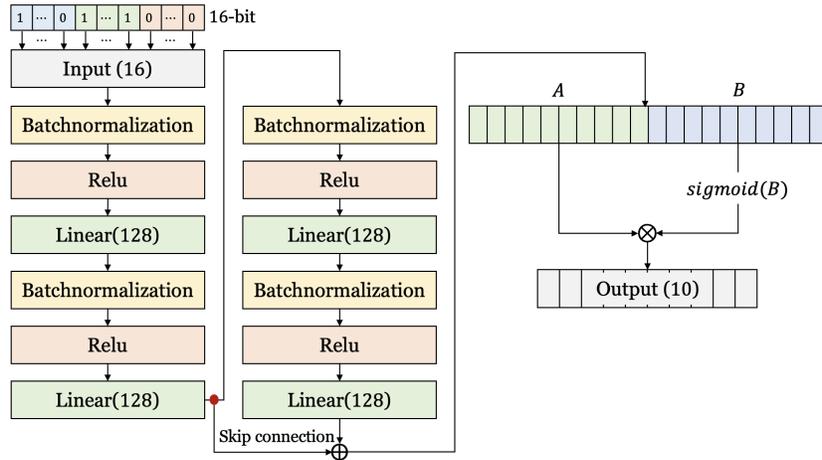


Fig. 6: The neural network structure for cryptanalysis for S-DES.

Structure of Neural Network for Cryptanalysis for S-AES Figure 7 shows the structure of the 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.

Structure of Neural Network for Cryptanalysis for S-SPECK Figure 8 shows the architecture of the neural network for key recovery attack for S-SPECK. The model for S-SPECK is much deeper and larger than the neural networks for S-DES and S-AES. The hidden layer consists of 25 residual blocks (including linear, batch normalization, and Relu). The units of the linear layer of each block are 256. Unlike the previous two models, the output layer has k neurons. Since the key length of SPECK is 64, the number of output units should originally be 64. But then the size of the model becomes too large. So, the number of output neurons was reduced to predict only the reduced key bits.

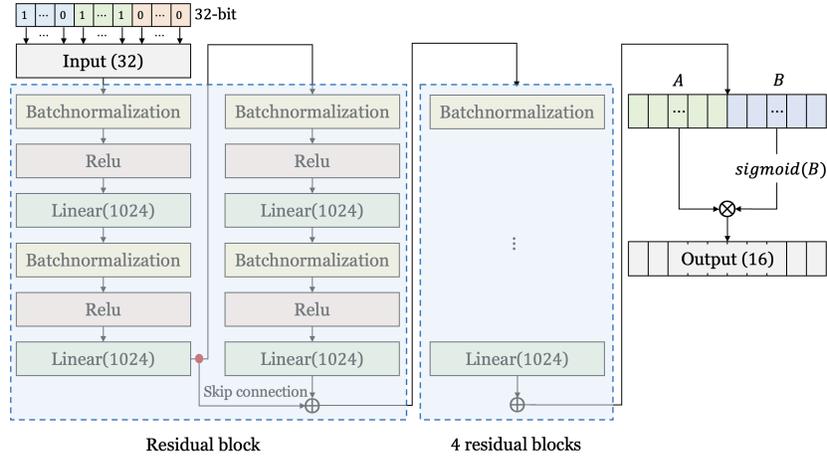


Fig. 7: The neural network structure for cryptanalysis for S-AES.

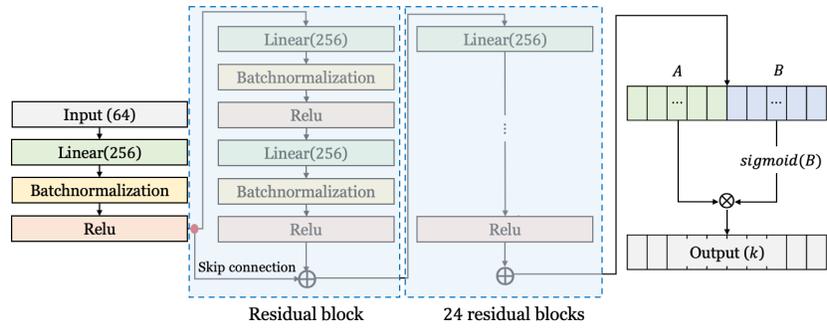


Fig. 8: The neural network structure for cryptanalysis for S-SPECK.

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 represents the difference between the two values (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.

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

Testing In the inference phase, a trained neural network that has 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.

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

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

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.

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

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 2 shows the result of inference for cryptanalysis. For comparison of the number of parameters with the previous work, the neural network described in [22] 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

Table 2: 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
[22]	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

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.

Bit Accuracy Probability by Epoch Table 3 shows the BAP by Epoch. In every epoch, bits 4 and 7 are safe. In other words, it is a bit difficult to predict because the accuracy is low even if it is sufficiently trained. In addition, since the 6-th and 9-th bits in all epochs are vulnerable bits, they can be easily inferred even with little training. Also, as the epoch increases, the number of safe bits are decreased. That is, as the training progresses further, the accuracy of them increases so that they are no longer safe bits. The 1-st and 2-nd bits are dropped from the safe bit from 20 epochs. Finally, after training with 100 epochs, the 10-th bit, which was not vulnerable, is detected as a vulnerable bit. This result shows that the 6-th and 9-th bits are the most vulnerable, and the 4-th and 7-th bits are the most secure.

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.

Table 3: Bit accuracy probability by epoch for S-DES (V : vulnerable bit, S : safe bit).

Epoch	1 st	2 nd	3 rd	4 th	5 th	6 th	7 th	8 th	9 th	10 th
15	S	S		S	S	V	S	S	V	
25				S	S	V	S	S	V	
35				S	S	V	S	S	V	
95				S	S	V	S	S	V	
100				S		V	S		V	V

- **Parameters:** Cryptanalysis for S-AES requires much larger parameters than S-DES. The number of parameters for S-AES with 12-bit key space is 5,326,944. And, the number of parameters for S-AES with 12-bit key space is 11,636,832. When the key space is increased by 1 bit, the number of parameters increases significantly compared to S-DES.
- **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 [20], and it was also set to the learning rate exponential decay method (The 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 cryptanalysis 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 12-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 the large dataset was not possible due to the constraints of the environment. This shows that deep learning-based cryptanalysis requires a large data set, GPU, and memory environments. Through this experiment, we show that cryptanalysis of S-AES is possi-

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	-	-	-	-	-	-	-	0.7	0.7	0.69	0.69	0.7	0.69	0.69	0.7	0.69
10-bit	-	-	-	-	-	-	0.63	0.63	0.63	0.64	0.63	0.63	0.6	0.6	0.6	0.6
11-bit	-	-	-	-	-	0.52	0.53	0.52	0.53	0.52	0.52	0.53	0.52	0.51	0.52	0.52
12-bit	-	-	-	-	0.51	0.51	0.5	0.5	0.5	0.5	0.5	0.51	0.5	0.5	0.5	0.5

ble up to an 11-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 Experiments on S-SPECK

Training Result In S-SPECK, the loss function, optimizer, and epoch are as follows.

- **Parameters:** The number of parameters of our model is 3,333,894. When a network smaller than this was used, there were cases where the BAP was less than 0.5. Also, since the size of the dataset is large, out-of-memory (OOM) occurs. Therefore the size of the model cannot be increased any further.
- **Loss:** We used the MSE loss function. The loss decreased to 0.2499. This result shows that it is barely trained considering the initial loss of 0.2548.
- **Optimizer:** We used the Adam optimizer with a cyclic learning rate (The range of learning rate is 0.001 to 0.002).
- **Epoch:** We set the number of epochs to 20. In order to the limitations of the experiment environment, we cannot use larger neural networks. Therefore, the accuracy did not improve even after learning more than 20 epochs.

Bit Accuracy Probability Table 5 shows the result of cryptanalysis for S-SPECK. In this experiment, more than 10,000,000 data were used. And to analyze more than a 6-bit key, more data must be required. However, we can't use more data because of environmental constraints. To

Table 5: BAP for S-SPECK (4~6-bit key space).

Key	1 st	2 nd	3 rd	4 th	5 th	6 th
4-bit	-	-	0.54	0.53	0.53	0.53
5-bit	-	0.52	0.51	0.51	0.52	0.52
6-bit	0.51	0.51	0.51	0.51	0.50	0.51

overcome this, we tried different methods (increasing the capacity of the neural network and trying several methods to solve underfitting). However, analysis was not possible because the number of data was too small compared to the complexity of the data. In previous work, key recovery was possible because certain bits were determined with high probability when the text-based key was used. However, they failed analysis when they use random key bits. As shown by our results and previous work, it is difficult to infer a key when all key bits have a random probability. Therefore, if these resource and time limitations are not addressed, deep learning-based key recovery is inevitably difficult.

4.5 Comparison of Cryptanalysis for S-DES, S-AES and S-SPECK

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 9, 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 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

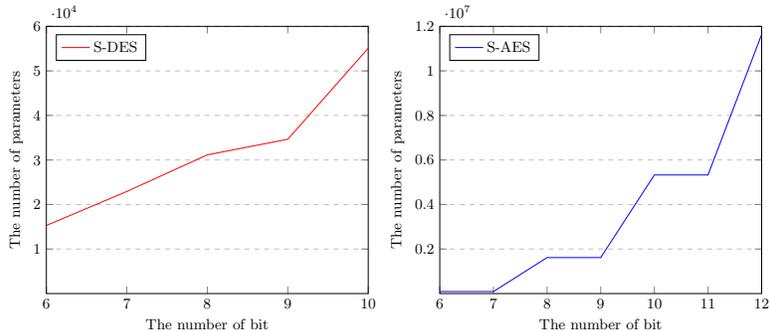


Fig. 9: Parameters of neural network for cryptanalysis (left: S-DES, right: S-AES).

than S-AES. Some key bits can be inferred. These results also apply to S-SPECK. It has a larger block size and key size than S-DES and S-AES, and is designed with a much more complex structure. Therefore, even attacks on small key bits are not easy. That is, the larger the block and key size and the more complex the structure, the more difficult it is to infer the key using plaintext and ciphertext.

Finally, many parameters are required for cryptanalysis of the Simplified cipher as shown in Figure 9, 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 and S-SPECK. Key recovery was possible up to a 12-bit key for S-AES and a 6-bit key space for S-SPECK. Through two experiments, we confirmed that it is practically impossible to recover the entire key for a modern cipher. In future work, we will apply deep learning-based cryptanalysis for other block ciphers.

References

1. Albawi, S., Mohammed, T.A., Al-Zawi, S.: Understanding of a convolutional neural network. In: 2017 international conference on engineering and technology (ICET), Ieee (2017) 1–6
2. 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. (2015) 1–6
3. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. *Journal of CRYPTOLOGY* **4**(1) (1991) 3–72
4. Blackledge, J., Bezobrazov, S., Tobin, P.: Cryptography using artificial intelligence. In: 2015 International Joint Conference on Neural Networks (IJCNN), IEEE (2015) 1–6
5. Dauphin, Y.N., Fan, A., Auli, M., Grangier, D.: Language modeling with gated convolutional networks. In: International conference on machine learning, PMLR (2017) 933–941
6. Farsal, W., Anter, S., Ramdani, M.: Deep learning: An overview. In: Proceedings of the 12th International Conference on Intelligent Systems: Theories and Applications. (2018) 1–6
7. Focardi, R., Luccio, F.L.: Neural cryptanalysis of classical ciphers. In: ICTCS. (2018) 104–115
8. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks. *Communications of the ACM* **63**(11) (2020) 139–144
9. Haykin, S.: *Neural networks and learning machines*, 3/E. Pearson Education India (2009)
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 770–778
11. Idris, M.F., Teh, J.S., Yan, J.L.S., Yeoh, W.Z.: A deep learning approach for active s-box prediction of lightweight generalized feistel block ciphers. *IEEE Access* **9** (2021) 104205–104216
12. Kariya, A., Behera, B.K.: Investigation of quantum support vector machine for classification in nisq era. arXiv preprint arXiv:2112.06912 (2021)
13. Kim, H.J., Song, G.J., Jang, K.B., Seo, H.J.: Cryptanalysis of caesar using quantum support vector machine. In: 2021 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), IEEE (2021) 1–5
14. Kukreja, H., Bharath, N., Siddesh, C., Kuldeep, S.: An introduction to artificial neural network. *Int J Adv Res Innov Ideas Educ* **1** (2016) 27–30
15. Matsui, M.: Linear cryptanalysis method for DES cipher. In: *Workshop on the Theory and Application of Cryptographic Techniques*, Springer (1993) 386–397
16. Musa, M.A., Schaefer, E.F., Wedig, S.: A simplified aes algorithm and its linear and differential cryptanalyses. *Cryptologia* **27**(2) (2003) 148–177
17. Noble, W.S.: What is a support vector machine? *Nature biotechnology* **24**(12) (2006) 1565–1567
18. Ooi, K., Vito, B.C.: Cryptanalysis of s-des. *Cryptology ePrint Archive* (2002)
19. Petneházi, G.: Recurrent neural networks for time series forecasting. arXiv preprint arXiv:1901.00069 (2019)
20. Ruder, S.: An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747 (2016)

21. Shannon, C.E.: Communication theory of secrecy systems. The Bell system technical journal **28**(4) (1949) 656–715
22. So, J.: Deep learning-based cryptanalysis of lightweight block ciphers. Security and Communication Networks **2020** (2020)
23. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press (2018)