Quantum Neural Network based Distinguisher for Differential Cryptanalysis on Simplified Block Ciphers

No Author Given

No Institute Given

Abstract. Differential cryptanalysis is cryptanalysis for block ciphers. It is a technique for inferring a key by analyzing the differential when the cryptographic algorithm is designed to be weak. It is more effectively compared to the brute force attack because the attack complexity is reduced. And, a neural distinguisher is neural network based distinguisher for differential neural networks. It makes the data complexity be reduced. In this work, we implemented quantum neural distinguisher for differential cryptanalysis. To the best of our knowledge, our work is the first neural distinguisher using quantum neural network. The quantum neural distinguisher is quantum-classical hybrid neural network based distinguisher, and it distinguishes the differential ciphertext pair from random data. We performed some experiments for simplified block ciphers (S-DES and S-AES), and the quantum neural distinguisher achieved quantum advantages (improved accuracy, less data, less parameters, faster convergence etc.) compared to classical neural networks. Our distinguisher achieved accracies of 98% for S-DES, 99% for S-AES. So our work can be used successfully as a distinguisher for differential cryptanalysis. In future work, we will apply our distinguisher for other block ciphers, and we will consider multiple differences.

Keywords: Hybrid Quantum-classical Neural Networks · Differential Cryptanalysis · Neural Distinguisher · Simplified Block Cipher.

1 Introduction

Differential cryptanalysis is one of cryptanalysis for block ciphers. It is a technique for inferring a key by analyzing the output difference according to the input difference when the cryptographic algorithm is designed to be weak. It works more effectively compared to the brute force because the attack complexity is reduced. Because, if the key used for brute force is the correct key, the output difference with respect to the input difference is satisfied with a high probability. Conversely, if the wrong key is used, the decrypted value does not satisfy the output difference. A large number of choosen plaintexts are required for this attack.

Also, block ciphers are designed to prevent differential attacks using wide-trail design strategies [1] and Shannon's principles [2]. Therefore, the method of

analyzing the differential trail requires a large number of data, which causes a bottleneck, which makes the differential trail analysis fail. If, in order to distinguish the r-round data of the n-bit block cipher from random data, a difference characteristic having a probability greater than $1 \div 2^n$ is required. That is, the probability of the output difference with respect to the input difference must be greater than the random probability $(1/2^n)$ [3]. Otherwise, we cannot succeed in differential cryptanalysis.

However, if a neural network-based distinguisher is used, it is possible to distinguish random data from differential data (differential data if the probability is greater than or equal to random probability), and data complexity for differential cryptanalysis can be reduced. This is called a neural distinguisher, and related works have been conducted.

In addition, as quantum computers have been developed in recent years, quantum neural network utilizing quantum computer is attracting attention. Quantum neural networks replaces the training process of classical neural networks with quantum circuits. In other words, the quantum circuit acts as a neural network. Simply, we use the rotation gate of a quantum circuit to update the output value of the circuit by changing the state of the qubit. This is similar to the traning process in which classical neural networks reduce losses while updating weights of network. Quantum neural networks have the advantage of being able to achieve fewer parameters and higher performance compared to classical neural networks. However, since current quantum computers are noisy intermediate-scale quantum computers, it is difficult to correct errors in qubits. Therefore, a hybrid neural network that combines a classical neural network and a quantum neural network is now stable in terms of performance.

In this work, we designed a quantum neural network-based distinguisher and compared it with existing classical neural distinguisher approaches. Our contribution is described in 1.1.

1.1 Contribution

First attempt on Quantum Neural Distinguisher based on Quantum-classical Hybrid Neural Network To the best of our knowledge, our work is the first quantum neural distinguisher using a quantum-classical hybrid neural network. We targeted S-DES and S-AES, which are simplified block ciphers, and achieved accuracies of 98% and 99%, respectively. So our work can be used successfully as a distinguisher.

Quantum Advantage We not only achieved successful classification accuracy, but also implemented a quantum neural distinguisher that uses less data and fewer parameters compared to classical neural distinguisher. In S-DES, there was a 2% improvement in accuracy for the same number of epochs and data, and a 28.7% decrease compared to the parameters of the classical distinguisher. In S-AES, the accuracy was improved by 18% for 1000 data, and the number of parameters decreased by 43% compared to the classical method. Also, for 2000

data, the accuracy was improved by 1% and the number of parameters decreased by 28.6%. Briefly explaining the reason, quantum neural networks can represent a wider range of data, and perform training while changing the state of qubits with a small number of quantum gates. In addition, quantum circuits require fewer parameters because they are not fully-connected and require as many parameters as the number of rotation gates. In other words, we implemented a neural distinguisher with quantum advantages thanks to the characteristics of quantum neural networks.

The remainder of this paper is organized as follows. In Section 2, related works are presented such as quantum neural network, previous works, quantum cryptanalysis using quantum neural network. In Section 3, the proposed method based on quantum neural network is introduced. In Section 4, the evaluation of our implementation is discussed. Finally, Section 5 concludes the paper.

2 Related Works

2.1 Classical Neural Networks

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. Neurons (i.e. nodes) in each layer perform a weighted sum operation using the node values and weights 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).

2.2 Quantum Neural Networks

A quantum neural network is an artificial intelligence that utilizes quantum mechanics phenomenon (entanglement and superposition). A quantum neural network consists of qubits and quantum gates on a quantum computer. Therefore, it learns quantum state data (parameterized quantum circuit) by encoding the classical data into quantum data. The parameter of the circuit is the rotation angle of the rotation gate, and the input data is used as the rotation angle. After encoding, the constructed quantum circuit is executed. At this time, each qubit state is changed as it passes through the quantum gate. Finally, when a qubit is measured, the state of the qubit is determined to be a classical value. The loss is calculated based on that value and the rotation angles of the quantum gate

are changed. The parameterized quantum circuit is trained by re-executing the quantum circuit to which the changed parameter is applied and repeating this process. Quantum neural networks require fewer parameters and fewer training data than classical neural networks. It also has the advantage of being able to perform better than classical neural networks.

Quantum-classical Hybrid Neural Networks A classical-quantum hybrid neural network is a neural network that uses a combination of classical and quantum neural networks. In other words, a parameterized quantum circuit is used as one of the layers that make up a classical neural network. Therefore, elements such as loss functions, optimization functions, metrics, and epochs are used the same as in classical neural networks. The training process consists of calculating the loss as a forward propagation process like a classical neural network and adjusting parameters through a backpropagation process based on the loss value. As such, the whole process is the same as for classical neural networks, but with quantum circuits as one layer. In quantum circuit execution, the state of qubits is measured after sequentially passing all gates constituting the circuit in the same way as in a quantum neural network. Finally, the expected value of the quantum circuit is obtained and used as the final output of the quantum layer. After calculating the loss based on the corresponding value, the existing parameter (the rotation angle of the qubit) is changed through a parameter shift method, etc. Through this process, the hybrid neural network is trained. Currently, there are many studies using a hybrid neural network, and it has the advantage of being more stable and performing better than a neural network using only quantum.

2.3 Differential Cryptanalysis

Differential cryptanalysis is a representative cryptanalysis method of block ciphers. The input difference (δ) is the XOR between the plaintext pairs (P_0, P_1), and the output difference (Δ) is the XOR between the ciphertext pairs. That is, as in Equation 1, if the delta is XORed to the random plaintext, it is calculated as P_1. Also, the results of encrypting (E) P_0 and P_1 are C_0 and C_1, respectively. Finally, by XORing C_0 and C_1, the output difference (Δ) can be calculated. A pair of input and output differences $((\delta, \Delta))$ is called differential. If encrypted data can be distinguished from random using the difference characteristic, data complexity is significantly lower than that of exhaustive search. In the case of an ideal encryption algorithm, when plaintext with any input difference is encrypted, the output difference should be uniform. Conversely, a weak cipher has a certain output difference. If there is a difference characteristic in which the value of the output difference with respect to the input difference is greater than the random distribution probability, the random distribution and the ciphertext can be distinguished from uniform distribution. In this case, the encryption algorithm is considered weak.

$$\begin{aligned} &P_1 = P_0 \oplus \delta, \\ &C_0 = E(P_0), C_1 = E(P_1), \\ &\Delta = C_0 \oplus C_1 \end{aligned} \tag{1}$$

2.4 Neural Network based Distinguisher for Differential Cryptanalysis

The neural network based distinguisher (neural distinguisher) is used for distinguishing the ciphertext data from random data. That is, it is a binary classification of random data and ciphertext data satisfying the difference. If there is only one input difference, it is binary classification and should achieve an accuracy of 0.5 or better. The neural distinguisher makes ciphertext data be distinguished. Then, the data complexity is lower than attacking for all data.

Many related works have been conducted. In Gohr's work [4], they designed a neural distinguisher for 7-round speck32/64. They distinguished multiple input differences from random data using a neural network classifier. And, compared to the existing distinguisher (not neural network), it achieves less data complexity for key recovery attacks. And, the interpretation of [4] was published in Eurocrypt2021 [5]. The artificial neural network-based differential analysis method is in progress for ciphers, such as SIMON [6] based on [4]. In Baksi et al. [7], they also designed a machine learning-based distinguisher. However, the target ciphers are the GIMLI, ASCON, KNOT, and CHASKEY, and two models that consider multiple differences and single difference were proposed. 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 interpretations 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.

2.5 Quantum Cryptanalysis using Quantum Neural Networks

On the other hand, machine learning and deep learning are recently applied to classical cryptanalysis. Cryptanalysis using quantum neural networks has only one case[8]. They performed a known-plaintext attack on the caesar cipher. They used a quantum support vector machine(QSVM) and could attack up to a 3-bit key due to the lack of resources such as qubits. As a result of the experiment, 100% accuracy was achieved when the shot was 5 for the 2-bit key, and 84% accuracy was achieved when the shot was 150 for the 3-bit key. In addition, when the attack on the 2-bit key was performed using a real quantum processor, there was a loss of accuracy of 7%. In addition, a real quantum computer requires about 5.5 times longer learning time than a simulator. Therefore, cryptanalysis using only quantum computers still has limitations due to a lack of resources.

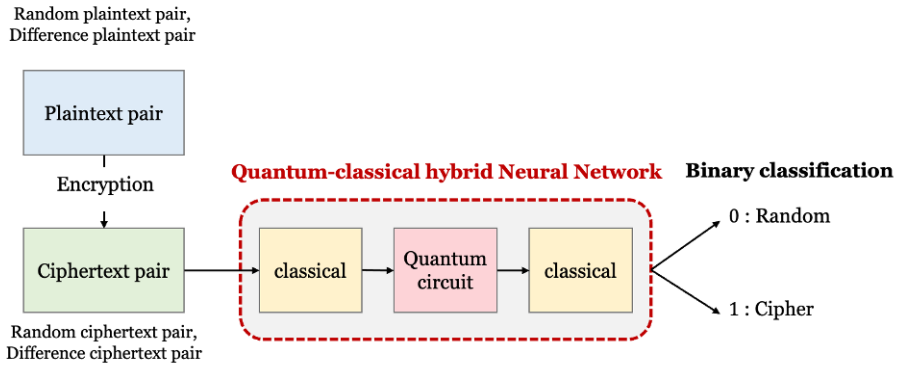

Fig. 1: Diagram of proposed method.

3 Proposed Method

In this paper, we present quantum neural distinguisher for differential cryptanalysis on simplified block ciphers. We implemented neural distinguisher using quantum-classical hybrid neural network, and the target ciphers are S-DES and S-AES. The purpose of quantum neural distinguisher is distinguishing the ciphertext pair from random pair for differential cryptanalysis. In other words, by utilizing the difference characteristic, an attacker can distinguish between random data and differential data, thereby reducing data complexity.

Figure 1 shows the system diagram of quantum neural distinguisher. First, plaintext pairs with a difference and random plaintext pairs are encrypted. The generated data are input to the quantum-classical hybrid neural network.The network then distinguishes the differential ciphertext pair from the random pair.

3.1 Dataset

In this work, the target ciphers are S-DES and S-AES. S-DES has 8-bit plaintext and ciphertext, 10-bit key. And, S-AES has 16-bit plaintext, ciphertext and key. Also, both ciphers are designed for 2-rounds.

Input difference characteristic These input differences are the first round input difference for each cipher. To generate dataset for this work, we used these input differences. The input difference is equal to the length of the plaintext because it is a value obtained by XORing the plaintext and another plaintext.

- **S-DES [9]:** 0x04 (0000 0100)
- **S-AES [10]:** 0x8000 (1000 0000 0000 0000)

Algorithm 1 Dataset preparation

Input: Input difference (δ), The number of data (N_{ds}), Encryption function(Enc)
Output: Dataset (DS)
for $i = 1$ **to** $N_{ds} \div 2$ **do**
  Choose random P_0, P_1 ($P_1 \neq P_0 \oplus \delta$)
  $P'_0 = P_0 \oplus \delta$
  $C_0 = Enc(P_0)$
  $C_1 = Enc(P_1)$
  $C'_0 = Enc(P'_0)$
  $(C_0||C_1)$ is labeled *class* 0 (Random ciphertext pair)
  $(C_0||C'_0)$ is labeled *class* 1 (Difference ciphertext pair)
  $DS_i \leftarrow (C_0||C_1)$
  $DS_{i+(N_{ds} \div 2)} \leftarrow (C_0||C'_0)$
end for
return DS

Dataset preparation Algorithm 1 shows the process for preparing dataset. First, we select random plaintext $P0$ and $P1$. The two plaintexts are not in a relationship that satisfies the input difference, but are random values. Next, P'_0 is obtained by XORing the input difference (δ) to P_0. After generating three plaintexts in this way, all plaintexts are encrypted, and these become $C_0, C_1, and C'_0$, respectively. C_0 and C_1 are random ciphertext pairs because they encrypt a random pair of plaintexts that are not related to each other. In addition, C_0 and C'_0 are differential ciphertext pairs obtained by encrypting plaintext pairs satisfying the input difference. Thus, the random ciphertext pair is labeled *class* 0, and the differential ciphertext pair is labeled *class* 1. In addition, if a total of N_{ds} data is to be generated, a random ciphertext pair and a differential ciphertext pair are equally generated in half.

3.2 Design of Quantum Neural Distinguisher

Training Algorithm 2 shows training process using quantum-classical hybrid neural network, and Figure 2 shows the architecture of quantum-classical hybrid neural network. The dataset (DS) generated in 3.1 is used for training. The overall flow is the same as the existing neural distinguisher [7]. The part that is different from the existing one is that the quantum circuit is used for training. First, input data (ciphertext pairs and random pairs) are input to the input layer. Since each input bit is assigned to each neuron, the number of the neuron of input layer is set to twice the block size. Then, the result obtained after passing through the input layer is output to 64 neurons and then input to the hidden layer. We used quantum circuits as hidden layer. Therefore, 64 neurons, the outputs of the input layer, are input to the quantum circuit used as the hidden layer. Quantum circuits consist of data embedding and quantum layer. We used amplitude embedding (Q_{amp}) as the data embedding layer. For amplitude embedding, $2^{N_{qubit}}$ features can be embedded using N_{qubit} qubits. Therefore, the

quantum circuit must be repeatedly executed as much as the number of neurons in the hidden layer ($Neuron_H$) divided by $2^{N_{qubit}}$. That is, data from the previous layer is divided and allocated to quantum circuit having N_{qubit} number of qubits. Since we used 4-qubits in our implementation, we input the 0th data (H_0) to the 15th data (H_{15}) into the first quantum circuit. Then, the 16th data (H_{16}) to the 31st data (H_{31}) are input to the second quantum circuit. This process is repeated 4 times to allocate all $Neuron_H$ hidden neurons.

After performing this embedding process, the parameterized quantum circuit for training is operated. In other words, implementing quantum circuits with rotating gates (with parameters). We used a strongly entangling layer as the quantum layer. In this quantum circuit, three quantum rotation gates are applied to each qubit, and entanglement is set by a certain rule. We will explain the details in 3.2.

After passing through a quantum neural network (layer), the output of each quantum neural network is merged back into one. In other words, each output vector of quantum circuit into one vector. By inputting this into the output layer, we get the final output of the quantum neural network. The loss is calculated using the final output value, and the parameters of the entire neural network including the quantum circuit are updated to minimize the loss. Finally, after the neural network is trained, we can get accuracy. If the accuracy is less than 0.5, this quantum neural distinguisher cannot distinguish ciphertext data from random data, so it is discarded. Conversely, if the classification probability is greater than 0.5, our model succeeds in distinguishing, so it can be used successfully as a quantum neural distinguisher.

Before using the quantum neural distinguisher, the test data must be generated in the same way as the training data. After preparing plaintext pairs and random pairs that satisfy the input difference, input them into the oracle to obtain ciphertext pairs. Then, the obtained ciphertext pairs are input into a trained quantum neural distinguisher. If the algorithm that generated the test data is an encryption algorithm, the distinguisher will be able to distinguish the ciphertext data from random, otherwise it will output an accuracy of 0.5 or less. That is, a well-trained quantum neural network can be used as a distinguisher.

Amplitude Embedding Figure 3 shows the 4-qubits quantum circuit for amplitude embedding. The amplitude embedding circuit consists of RY (for rotation) and CNOT (for entanglements) gates. A rotation gate is used, but it is different from the parameters of a quantum neural network because it is to embed the input data. Also, unlike angle embedding, amplitude embedding can use 4-qubits to embed 16 classical values. That is, 16 values among the input data are used to express the amplitude vector representing the state of the qubit. In this way, the input data in the classical state can be transformed into the quantum state.

Parameterized Circuit Figure 4 shows the part of 4-qubits quantum circuit with 10 quantum layer for strongly entangled quantum layer. That is, it shows

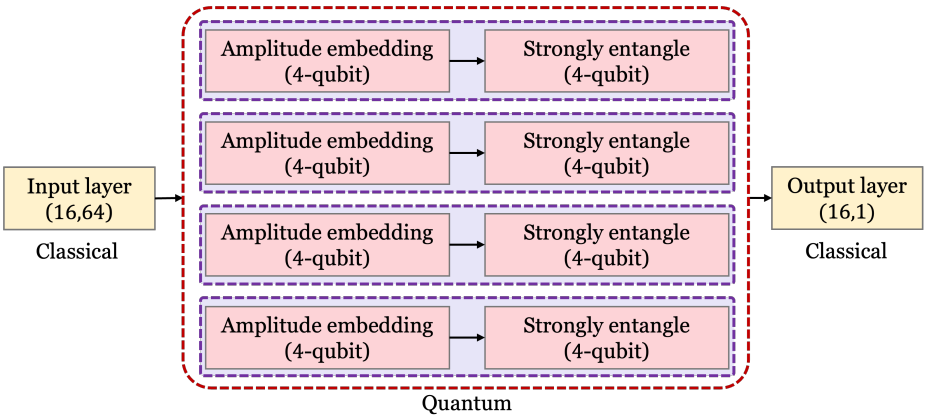

Fig. 2: Architecture of quantum-classical hybrid neural network.

Algorithm 2 Training process using quantum-classical hybrid network

Input: Dataset (DS), The number of qubits (N_{qubit}), Classical input, hidden and output layer ($Input, H, Output$), Quantum circuit for amplitude embedding (Q_{amp}), Quantum circuit for quantum layer (Q_{ent})

Output: Trained hybrid model (QC_{Hybrid})

$Neuron_H \leftarrow$ the number of neuron of hidden layer
$H_i \leftarrow i$-th neuron of classical hidden layer
$N_{qc} \leftarrow (Neuron_H \div 2^{N_{qubit}})$; the number of quantum circuit
$Q_{amp(i)}$ is i-th Q_{amp}
$Q_{ent(i)}$ is i-th Q_{ent}
$Q_{states(i)} \leftarrow$ states of qubits of i-th quantum circuit

for $i = 0$ **to** $Epoch - 1$ **do**
  $x \leftarrow Input(DS)$
  $x \leftarrow H(x)$
  for $i = 0$ **to** $N_{qc} - 1$ **do**
    $Q_{states(i)} \leftarrow Q_{amp(i)}(H_{2^{N_{qubit}}*i+0}, H_{2^{N_{qubit}}*i+1}, \cdots, H_{2^{N_{qubit}}*i+15})$
    $Q_{states(i)} \leftarrow Q_{ent(i)}(Q_{states(i)})$
    $x_i \leftarrow \text{measure}(Q_{states(i)})$
  end for
  $x \leftarrow (x_0||x_1||\cdots||x_{N_{qc}-1})$
  $outputs \leftarrow Output(x)$
  Compute $loss$ and $accuracy$
  Adjust the parameters of the quantum circuit
end for
if $accuracy < 0.5$ **then**
  Abort QC_{Hybrid}
else if $accuracy > 0.5$ **then**
  return QC_{Hybrid}
end if

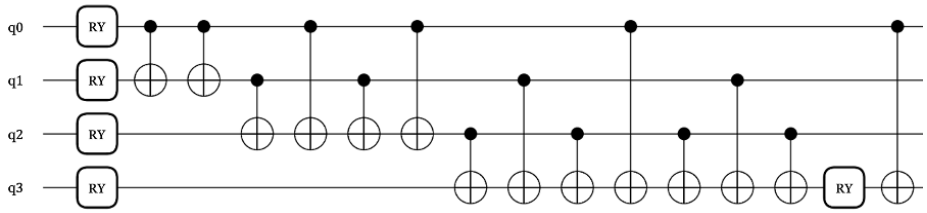

Fig. 3: Quantum circuit for amplitude embedding.

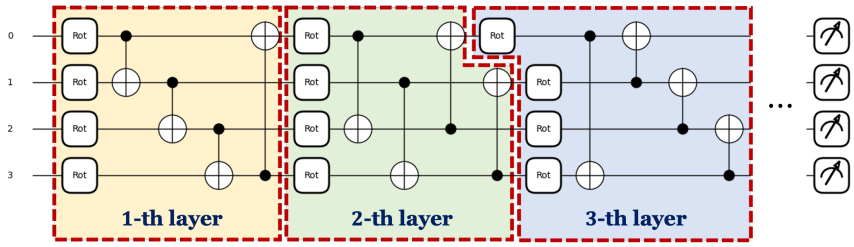

Fig. 4: Quantum circuit for strongly entangled quantum layer.

one strongly entangled layer and is composed of Rot (RZ+RY+RZ) gates and CNOT gates. This circuit is designed for rich entanglement and rotation, and the number of layer iterations (r) is required to design such entanglement. First, when designing a quantum neural network, we can set how many quantum layers to stack. Let the number of quantum layers and the number of qubits be N_{ql} and N_{qubit}, respectively. Also, when the quantum circuit is executed, it is assumed that the currently operating layer is the lth layer ($l < N_{ql}$). Then, as in Equation 2, the number of layer repetitions (r) can be obtained. The obtained r value is used to design the entanglements. If the second layer is operating in a 4-qubit quantum circuit, $r = 2$. As a result, the i-th qubit is entangled with the $(i + r \ mod \ N_{qubit})$-th qubit. In other words, rather than being entangled with adjacent qubits, one qubit is evenly entangled with other qubits, resulting in more influence.

Also, the circuit uses rotation gates and CNOT gates. The rotation gates used here requires angle of rotation. These are called parameters of a quantum neural network, and these values are updated through training. That is, the output of the circuit is changed while changing the rotation angle of the rotation gate to minimize the loss.

$$r = l \mod N_{qubit} \tag{2}$$

4 Evaluation

We used AMD Ryzen 7 4800h with radeon graphicx16 processor, 15GB RAM, Ubuntu 20.04.2 LTS, Python 3.9, Tensorflow 2.9.1 and keras 2.9.0. For design of the quantum circuit, Pennylane, a library for hybrid neural networks, was used, and the 'mixed.qubit' simulator provided by pennylane was used as a device for circuit execution.

4.1 Quantum-classical Hybrid Neural Network

Table 1 shows the detail of quantum-classical hybrid neural network for differential cryptanalysis on S-DES, S-AES and S-PRESENT. The following is the details of Table 1.

- **The number of qubits:** We use 4-qubits for both ciphers. As a result of experimenting with circuits of 2-qubits, sufficient performance could not be obtained. When using 4-qubits, it took about 5300 seconds for 1 epoch. If the number of qubits used doubles, the training time also doubles. Therefore, the execution time is too long if more than 4-qubits are used. So we used 4-qubits.
- **Quantum embedding:** 4-qubits are a level that can be used without difficulty even with current quantum computers. This is possible because we use amplitude embedding and hybrid neural network. Amplitude embedding allocates $2^{N_{qubit}}$ features to N_{qubit}, so fewer qubits can be used. In addition to this method, we also experimented with a quantum machine learning approach called Quantum Support Vector Machine (QSVM). However, QSVM only uses quantum, not quantum-classical hybrid approach, which uses angle embedding. Therefore, if the ciphertext pair is a total of 16 bits, 16 qubits are needed to embed one feature to one qubit. However, as mentioned earlier, we used amplitude embeddings and quantum-classical hybrid neural networks. Since we used amplitude embeddings we can use fewer qubits. Also in hybrid neural networks we can add or reduce classical hidden layers. So we can adjust the dimensions of the hidden layers and use fewer qubits accordingly.
- **Quantum layer:** Next, the strongly entangling quantum layer was used as a parameterized quantum circuit for training. This is a quantum circuit with strong entanglement and rotation, as mentioned in 3.2. Random quantum circuits or quantum circuits entangled with adjacent qubits can also be designed, but as a result of our experiments, this quantum circuit achieved the most stable and best performance. In particular, in the case of a random circuit, the performance was not stable because the quantum gate and entanglement were set at random.
- **The number of quantum circuit, layer, rotation gate:** In S-DES and S-AES, 1 quantum circuit and 15 quantum layers, 4 quantum circuits and 10 quantum layers were used, respectively. The reason is that S-AES is a more complex cryptographic algorithm than S-DES. Therefore, more circuits (more parameters) are considered necessary. Also, considering the number

Table 1: Details of quantum-classical hybrid neural network for differential cryptanalysis.

	S-DES	S-AES
The number of qubits	4	4
Quantum embedding	Amplitude	Amplitude
Quantum layer	Strong entangle	Strong entangle
The number of quantum circuit	1	4
The number of quantum layer	15	10
The number of rotation gate	180	480
The number of parameters	457	777
The number of data	1000	2000
Epoch	25	25
Test accuracy	98%	99%

of circuits and layers, the number of rotation gates can be obtained. As in Equation 3 the number of rotation gates can be calculated. $N_{Rotation}$, N_{qc}, N_{qubit} and N_{ql} are the number of rotation gate (e.g. Rx, Ry, Rz), quantum circuit, qubit and quantum layer, respectively. And $N_{Rotation}$ is rotation gate with RZ-RY-RZ combined. Also, $N_{Rotation}$ is equal to the total number of parameters used in the quantum circuit (not the entire quantum neural network). CNOT is also quantum gate, but it doesn't have parameters.

- **The number of parameters:** The quantum neural distinguisher for S-DES and S-AES consisted of 457 and 777 parameters, respectively. For S-AES, the number of parameters of the entire neural network is larger in the case of S-AES because more quantum resources are used. Since parameters of the input and the output layer exist in addition to the parameters of the quantum circuit, the difference between the quantum parameters for both ciphers is not the difference between the overall parameters of quantum-classical hybrid nerual network.
- **The number of data:** The distinguisher for S-DES, which is a simple cipher with a relatively small key space, required 1000 data, and twice that number was used for S-AES.
- **Epoch:** we used 25 epochs for both ciphers. When more than 25 epochs were used, little convergence was performed and the overall accuracy was not significantly affected.
- **Test accuracy:** Finally, for the test data of S-DES and S-AES, accuracies of 98% and 99% were achieved, respectively. When other values were used as input differences, a result barely exceeding 0.5 was obtained, but as a result of using the correct input difference, it was successfully distinguished with high accuracy.

$$N_{Rotation} = N_{qc} \cdot (3 \cdot (N_{qubit} \cdot N_{ql})) \tag{3}$$

4.2 Comparison with Quantum-classical Hybrid Network and Classical Neural Network

We compared our quantum neural distinguisher with classical neural distinguisher. Table 2 shows the comparison result for S-DES, and Table 3 is about S-AES. In these tables, Tr, Val, Ts are training, validation and test accuracy, N_{Params} is the number of parameters, N_{Data} is the number of data.

In Table 2, for the same epoch and the number of training data, the quantum neural distinguisher achieved 2% higher accuracy and required fewer parameters. However, since S-DES is a relatively simple cryptographic algorithm, it has reached sufficient performance even with classical neural networks.

In Table 3, we tested 1000 and 2000 data for 25 epochs. In (25,1000) case, the accuracy of the quantum version achieved 18% higher accuracy than the classical version. And in (25,2000) case, the classical and quantum versions achieved similar performance because the number of data increased, but the accuracy of the quantum neural distinguisher was 1% higher. In addition, the quantum version used about 70% of the number of parameters of the classical neural distinguisher. That is, S-DES obtained 99% accuracy with only 1000 data, and S-AES, which is a slightly more complex cipher, required twice as much data.

Also, in the result when the number of data of S-AES is 1000, the same epoch is used, but the accuracy of the quantum distinguisher is higher. In other words, it can be observed that the convergence of the quantum neural distinguisher is performed faster and more stably. In addition, the reason for using less data and fewer parameters but having higher performance is thought to be that qubits have a wider and more sophisticated data representation than classical bits.

Quantum Advantage for quantum neural distinguisher In summary, we can obtain the following quantum advantages in a quantum neural distinguisher. As mentioned above, due to the characteristics of quantum neural networks, they require less data and fewer parameters, and higher performance with more fast convergence can be achieved compared to classical neural networks. In S-DES, there was a 2% improvement in accuracy for the same number of epochs and data, and a 28.7% decrease compared to the parameters of the classical distinguisher. In S-AES, the accuracy was improved by 18% for 1000 data, and the number of parameters decreased by 43% compared to the classical method. Also, for 2000 data, the accuracy was improved by 1% and the number of parameters decreased by 28.6%. The reason fewer parameters are required is, in a classical neural network, most of the nodes in layers are connected to each other, and the number of necessary parameters increases dramatically as the number of neurons increases and the depth of the network increases. However, since quantum neural networks require as many parameters as the number of rotation gates used, relatively few parameters are required. In the neural network used in this experiment, the difference in the number of parameters is not large, but the difference will become larger as the size of the classical neural network grows.

Table 2: Comparison between classical and quantum classical neural networks for differential crypanalysis for S-DES.

$(Epoch, N_{Data})$	(25, 1000)	
Target	S-DES (Classical)	S-DES (Quantum, Ours)
Tr	96	97
Val	97	97
Ts	96	98
N_{Params}	641	457

Table 3: Comparison between classical and quantum classical neural networks for differential crypanalysis for S-AES.

$(Epoch, N_{Data})$	(25, 1000)		(25, 2000)	
Target	S-AES (Classical)	S-AES (Quantum, Ours)	S-AES (Classical)	S-AES (Quantum, Ours)
Tr	68	92	92	100
Val	75	86	99	99
Ts	65	83	98	99
N_{Params}	1089	617	1089	777

5 Conclusion

In this work, we implemented the first neural distinguisher for simplified block ciphers based on quantum-classical hybrid neural networks. The neural distinguisher can reduce the data complexity of differential cryptanalysis and is a task that distinguishes ciphertext from random data. Our work obtained higher accuracies (2% for S-DES, 18% (1000 data) and 1% (2000 data) for S-AES) than classical neural classifiers. In addition, it required a reduced number of parameters by 28.6% to 43% compared to the classical neural distinguisher. That is, a quantum neural distinguisher with quantum advantage was designed using a neural network based on quantum characteristics. As a future work, we will design a distinguisher for other ciphers and multiple input differentials.

References

1. J. Daemen and V. Rijmen, *The design of Rijndael*, vol. 2. Springer, 2002. 1
2. L. R. Knudsen and M. Robshaw, *The block cipher companion*. Springer Science & Business Media, 2011. 1
3. T. Yadav and M. Kumar, "Differential-ml distinguisher: Machine learning based generic extension for differential cryptanalysis," in *International Conference on Cryptology and Information Security in Latin America*, pp. 191–212, Springer, 2021. 2
4. A. Gohr, "Improving attacks on round-reduced speck32/64 using deep learning," in *Annual International Cryptology Conference*, pp. 150–179, Springer, 2019. 5

5. 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
6. Z. Hou, J. Ren, and S. Chen, "Cryptanalysis of round-reduced Simon32 based on deep learning," *Cryptology ePrint Archive*, 2021. 5
7. 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, 7
8. H.-J. Kim, G.-J. Song, K.-B. Jang, and H.-J. Seo, "Cryptanalysis of caesar using quantum support vector machine," in *2021 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, pp. 1–5, IEEE, 2021. 5
9. K. Ooi and B. C. Vito, "Cryptanalysis of s-des," *Cryptology ePrint Archive*, 2002. 6
10. M. A. Musa, E. F. Schaefer, and S. Wedig, "A simplified aes algorithm and its linear and differential cryptanalyses," *Cryptologia*, vol. 27, no. 2, pp. 148–177, 2003. 6