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

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

IT Department, Hansung University, Seoul, South Korea,
{khj1594012, starj1023, dlatpwls834, etus1211, dnjsdndeee,
hwajeong84}@gmail.com

Abstract. Differential cryptanalysis is a block cipher analysis technology that infers a key by using the difference characteristics. Input differences can be distinguished using a good difference characteristic, and this distinguishing task can lead to key recovery. Artificial neural networks are a good solution for distinguishing tasks. For this reason, recently, neural distinguishers have been actively studied. We propose a distinguisher based on a quantum-classical hybrid neural network by utilizing the recently developed quantum neural network. To our knowledge, we are the first attempt to apply quantum neural networks for neural distinguisher. The target ciphers are simplified ciphers (`S-DES`, `S-AES`, `S-PRESENT-[4]`), and a quantum neural distinguisher that classifies the input difference from random data was constructed using the Pennylane library. Finally, we obtained quantum advantages in this work: improved accuracy and reduced number of parameters. Therefore, our work can be used as a quantum neural distinguisher with high reliability for simplified ciphers.

Keywords: Quantum Neural Network · Simplified Block Ciphers · Distinguisher.

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. Block ciphers are designed to prevent differential attacks using wide-trail design strategies [9] and Shannon's principles [22]. 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 $(1 \div 2^n)$ [26]. Otherwise, we cannot succeed in differential cryptanalysis. If the good difference characteristic is used, the input difference can be distinguished, and when the difference characteristic is not satisfied, the random data and the ciphertext data are not distinguished. Distinguishing differential data can lead to key recovery attacks [18]. For this distinguishing task, artificial neural networks can be a good solution. An artificial neural network-based distinguisher 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.

1.1 Our Contribution

In this work, we designed a quantum neural network-based distinguisher and compared it with existing classical neural distinguisher approaches. The contribution of this paper can be summarised as follows:

1. **First attempt on Quantum Neural Distinguisher based on Quantum-classical Hybrid Neural Network** As far as we know, this work is the first attempt for a distinguisher using a quantum neural network. We targeted the simplified block ciphers `S-DES`, `S-AES`, and `S-PRESENT-[4]`, and achieved accuracies of 98%, 99%, and 94%, respectively. Therefore, our work can be successfully used as a quantum neural distinguisher with high confidence.
2. **Quantum Advantage** We obtained quantum advantages in terms of accuracy and the number of parameters by applying a quantum neural network to the distinguisher. In 2-round `S-DES`, we achieved 2%

higher test accuracy and reduced the number of parameters by 29%. In 2-round `S-AES`, we got 1% higher accuracy and reduced the number of parameters by 29%. Finally, in 3-round `S-PRESENT-[4]`, we achieved 7% higher accuracy and reduced parameters by 32%. As can be seen from these results, the largest quantum benefit was obtained in the most complex data, 3-round `S-PRESENT-[4]`. In summary, in this work, we designed a quantum neural distinguisher with quantum advantages over classical neural networks by utilizing a quantum neural network that can represent more sophisticated and wide range of data.

2 Background

2.1 Classical Neural Networks

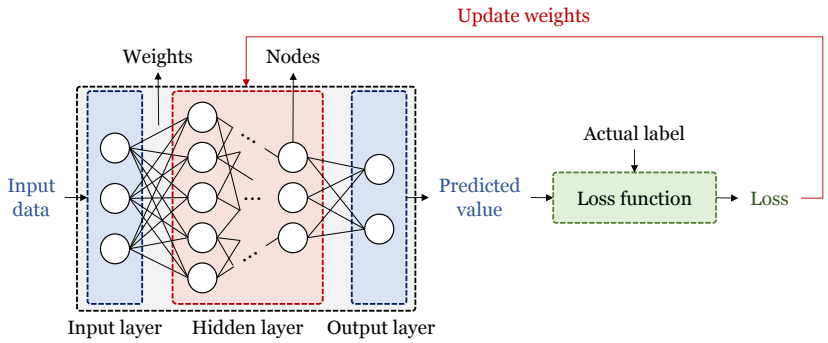

Fig. 1: Training process of artificial neural network.

Artificial neural networks [15] are supervised learning algorithms. As shown in Figure 1, a neural network is constructed in the form of stacked layers of multiple nodes, and multiple layers that exist between the input layer and the output layer are called hidden layers. Neurons (nodes) of each layer is multiplied by weights and the values of neurons in the previous layer connected to them, and then they are added [1]. The calculated values are input to the non-linear activation function, then the values are determined as the value of the current node. All nodes of the corresponding layer are input to the next layer and are used to calculate the values of the nodes. After performing this process in all layers, the final value of the network is output. The output is input to the loss function along with the label (the correct answer of the actual data).

A loss is then calculated that represents the difference between the label and the predicted value. In order to minimize the loss, the weights of the neural network are updated, and the neural network is trained to make a correct prediction. Artificial neural networks can solve classification and regression problems. Classifying into two classes is called binary classification, and classifying into multiple classes is multiple classification. In addition, for this task, various structures of neural network can be used. There are Multi Layer Perceptron (MLP) with the most basic structure, Convolutional Neural Network (CNN) [2] effective for image processing, Recurrent Neural Network (RNN) [25] suitable for processing time series data, and Generative Adversarial Network (GAN) [14] etc.

2.2 Quantum Neural Networks

Quantum Circuit Quantum circuits are constructed through bits and gates like classical logic circuits. However, instead of the classic bits and gates, quantum circuit use qubits and quantum gates that have the superposition and entanglement principles of quantum mechanics, and it works in quantum computers. A qubit is like a classical bit, but through superposition states, values exist as probabilities and are determined to be a single value after observation. In addition, since qubit can represent any value in the bloch sphere, it can represent a richer range of values than classical bits. In addition, multiple physical qubits are required for one logical qubit (without error), and error correction techniques are required. However, current quantum computers don't have enough resources and techniques to correct errors.

There are several types of gates used in quantum circuits [10], and the state of a qubit can be changed by applying quantum gates. Figure 2 shows some of the quantum gates. The hadamard (H) gate make superposition so that qubits can have both 0 and 1 states at the same time. Also, when the same qubit passes through the hadamard gate again, it is restored to its original state. The X gate changes the state of the qubit (e.g. from 0 to 1), and it changes the probability in the superposition state. The $CNOT$ gate uses the first qubit as the control qubit. When the control qubit is 1, the NOT operation is applied to the second qubit. Here, entanglement for two qubits is used. Next, there are Rx, Ry, and Rz rotation gates that rotate the qubits about the x, y, and z axes. These gates rotate the qubit by the rotation angle based on a specific axis.

Quantum circuits can be constructed by applying these quantum gates to qubits. At this time, an efficient circuit design is required considering the depth (length of the circuit) and width (the number of qubits) of

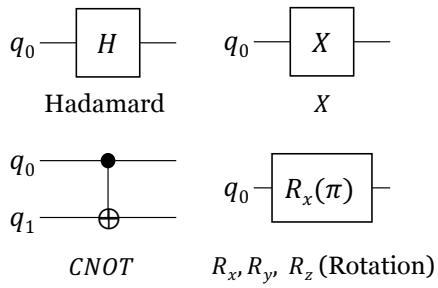

Fig. 2: Quantum gates.

the circuit. Quantum circuits can be implemented using the quantum language QASM and host languages such as Python and JavaScript in various quantum computing frameworks [12] such as Qiskit, ProjectQ, and Cirq.

Quantum Neural Network A quantum neural network [20] is an artificial intelligence that utilizes quantum mechanics phenomenon (entanglement and superposition). Figure 3 shows the training process of quantum neural network. A quantum neural network consists of qubits and quantum gates on a quantum computer. Therefore, it learns quantum state data by encoding the classical data into quantum data. There are several types of encoding. For angle embedding, n input data are used as rotation angles of the rotation gate for n qubits. For amplitude embedding, 2^n input data are converted into amplitude vectors of n qubits. After embedding, the quantum circuit is executed. In quantum circuit, the parameter for training is the rotation angle of the rotating gate. At this time, each qubit state is changed as it passes through the quantum gate. A quantum circuit with trainable parameters is called a parameterized quantum circuit. 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. However, with current quantum computers, it is difficult to use many qubits [5]. Therefore, the quantum-classical hybrid neural network [7,11] combined with the classical

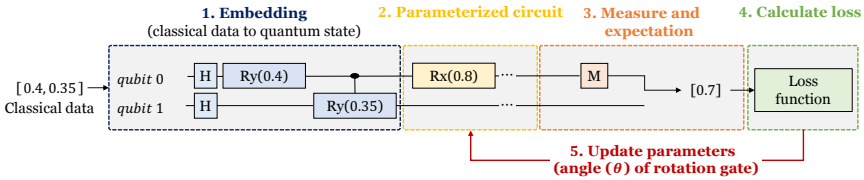

Fig. 3: Training process of quantum neural network.

neural network is now widely used. In quantum-classical hybrid neural network, a parameterized quantum circuit is used as one of the layers of a classical neural network. Therefore, elements such as loss functions, optimization functions, metrics, and epochs are used the same as in classical neural networks. Currently, hybrid neural networks have the advantage of being more stable and able to obtain higher accuracy than using only quantum circuits.

2.3 Differential Cryptanalysis

Differential cryptanalysis [16] 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

At CRYPTO 2019, Gohr [13] proposed a first deep learning-based neural distinguisher for round-reduced SPECK. They used a deep learning-based classifier to distinguish random data from input difference, and reduced data complexity for key recovery attacks compared to classical distinguisher. The accuracy of the neural network distinguisher for 8 round SPECK is about 0.514, and since we obtained an accuracy of 0.5 or more, we can classify cipher data from random data. After Gohr's work, in [17], differential cryptanalysis for SIMON was performed using the deep learning-based neural distinguisher. In [13], one ciphertext pair (two ciphertexts) was used as input data, whereas k ($k > 1$) ciphertext pairs were used as input data in [8]. That is, k ciphertext pairs are input to the neural distinguisher and classified. Their target ciphers are SPECK, CHASKEY, PRESENT, and DES. The experiment was conducted according to the k, and accuracy of 0.5 or more was obtained in all cases. In [6], they conducted experiments on multiple input differentials in 5 round SPECK. It was analyzed that there are difference characteristics with a higher probability than 0x00400000, which is the input difference of the 5 round SPECK used in Gohr's work. They experimented with the top 25 input differences. In most cases, an accuracy of 75% or better was achieved, and for 0x28000010, an accuracy of 90% or more was achieved. In [4], the target ciphers are GIMLI, ASCON, KNOT, and CHASKEY, and two models were proposed considering multiple input differences and single difference. The neural distinguisher for multiple input differences performs multi classification by setting each input difference as a class, and the neural distinguisher for single input difference performs binary classification on random data and ciphertext data. Baksi et al. experimented using MLP, CNN, and LSTM (Long Short Term Memory) for GIMLI-CIPHER. In this task, MLP performed the best, and CNN could not achieve an accuracy higher than 0.5. Therefore, in their work, the MLP model was used for other ciphers. However, the existing deep learning-based neural distinguisher has limitations in round expansion due to memory lack and data complexity. To overcome this, a new approach combining classical distinguisher and neural distinguisher has been proposed in [26]. Input the input difference to the classical distinguisher to find the output difference for r rounds. Then, the distinguisher for the extended round was proposed using the output difference of the r round as the input difference of the neural distinguisher. As such, research on various ciphers, input differences, neural network structures, etc. is being actively conducted using neural

distinguisher, and current studies are the result of round reduced ciphers, not full rounds.

2.5 Quantum Neural Networks based Quantum Cryptanalysis

On the other hand, machine learning and deep learning are recently applied to classical cryptanalysis. Cryptanalysis using quantum neural networks has only one case [21]. They performed a known-plaintext attack on the caesar cipher. They used a quantum support vector machine(QSVM) [19] 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.

2.6 Simplified Block Ciphers

`S-DES` `S-DES` [24] is a reduced version of the `DES` algorithm, and has an 8-bit block size and a 10-bit key size. In key scheduling, two 8-bit sub-keys are generated by permutation and shift operations on a 10-bit key.

The encryption procedure can be summarized as $C = E(P, K) = IP^{-1}(\rho_2(\rho_1(IP(P)))$. `S-DES` consists of initial permutation (IP), Cipher function f (expansion, key addition, substitution and permutation), and swap operations. Before the 1-round, the plaintext that has been subjected to initial permutation is divided into L_0 and R_0 (They are plaintext to be encrypted). The round function(ρ) is performed twice in total and is calculated as follows $L_i = R_{i-1}$, $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$.

`S-AES` `S-AES` [23] is composed of nibbles substitution(NS), shift rows(SR), mixcolumns(MC), and key addition(A_K). The `S-AES` key expansion and encryption algorithms are all based on an S-box (`94ABD1856203CEF7`) that depends on a finite field with 16 elements. This cipher has a 16-bit key, operates on 16-bit plaintext, and goes through 2 rounds to generate 16-bit ciphertext. The encryption algorithm consists of 8 functions ($A_{K_2} \circ SR \circ NS \circ A_{K_1} \circ MC \circ SR \circ NS \circ A_{K_0}$) for 2 round, and A_{K_0} is applied first.

S-PRESENT-[4] S-PRESENT-[4] [3], the small scale variant of the lightweight block cipher PRESENT, is based on 80-bit key. It was designed by reducing the block size to $4n$-bits while maintaining the PRESENT structure. Therefore S-PRESENT-[4] means block size of 16-bit. In other words, when n is 16, it becomes a PRESENT with an 80-bit key and a 64-bit block size. The algorithm is an substitution-permutation network (SPN) structure, and each round consists of AddRoundKey, S-Box Layer (Substitution layer), and P-Layer (Permutation layer). S-Box Layer is based on a single 4-bit S-box (C56B90AD3EF84712). The bit permutation function of P-Layer follows $P_i = [0, 4, 8, 12, 1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11, 15]$.

3 Design of Quantum Neural Network for Distinguisher

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, S-AES and S-PRESENT-[4]. 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 4 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

Input difference characteristic In this work, the target ciphers are S-DES, S-AES, and S-PRESENT-[4]. S-DES has 8-bit plaintext and ciphertext. And S-AES and S-PRESENT-[4] have 16-bit plaintext and ciphertext. Therefore, the input differences is 8-bit for S-DES, 16-bit for S-AES and S-PRESENT-[4]. 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 [24]: 0x04 (0000 0100)
- S-AES [23]: 0x8000 (1000 0000 0000 0000)
- S-PRESENT-[4] [3]: 0x0007 (0000 0000 0000 0111)

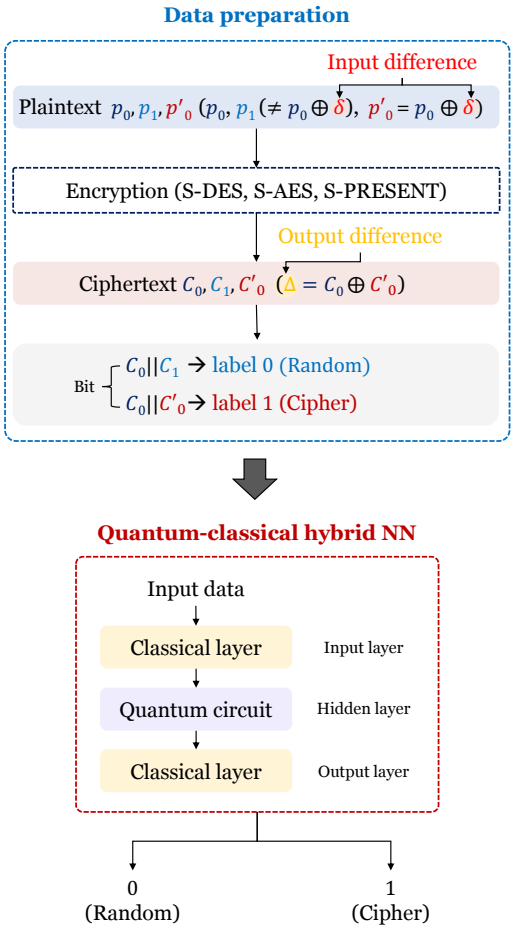

Fig. 4: Diagram of proposed method.

Dataset preparation Algorithm 1 shows the process for preparing dataset. First, we select random plaintext P_0 and P_1. 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

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

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 5 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 [4]. 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 a quantum circuit as a hidden layer and optionally a classical hidden layer (only for S-PRESENT). Therefore, 64 neurons, the outputs of the input layer, are input to the quantum circuit used as the hidden layer after going through the classical hidden layer (optional). 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 6 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 values. That is, 16 values among the input data are used to express the amplitude vector

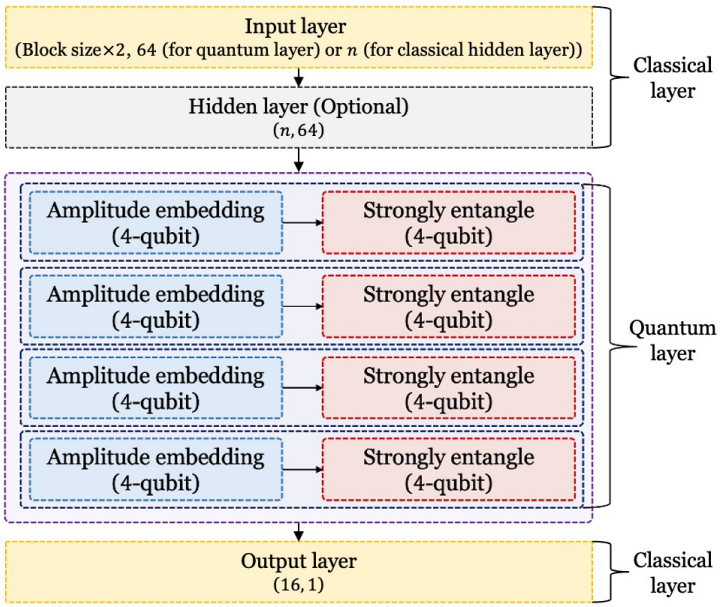

Fig. 5: Architecture of quantum-classical hybrid neural network.

representing the state of the qubit. In this way, the input data in the classical state can be transformed into the quantum state.

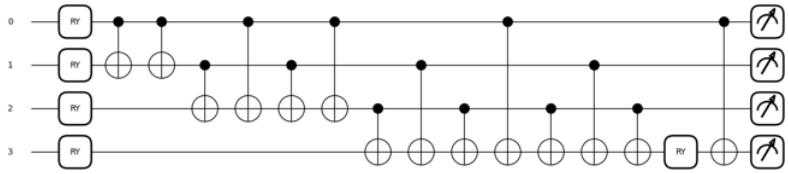

Fig. 6: Quantum circuit for amplitude embedding.

Parameterized Circuit Figure 7 shows the part of 4-qubits quantum circuit with 10 quantum layer for strongly entangling quantum layer. That is, it shows one strongly entangling layer and is composed of Rot (RZ+RY+RZ) gates and CNOT gates. And, the entire circuit is constructed by stacking several quantum layers. This circuit is designed for rich entanglement and rotation, and the number of layer iterations (r) is

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
$Nq \leftarrow 2^{N_{qubit}}$
$N_{qc} \leftarrow (Neuron_H \div Nq)$; 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**
        $Q_{states_{(i)}} \leftarrow Q_{amp_{(i)}}(H_{Nq*i+0}, \cdots, H_{Nq*i+15})$
        $Q_{states_{(i)}} \leftarrow Q_{ent_{(i)}}(Q_{states_{(i)}})$
        $x_i \leftarrow \text{measure}(Q_{states_{(i)}})$
    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

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.

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

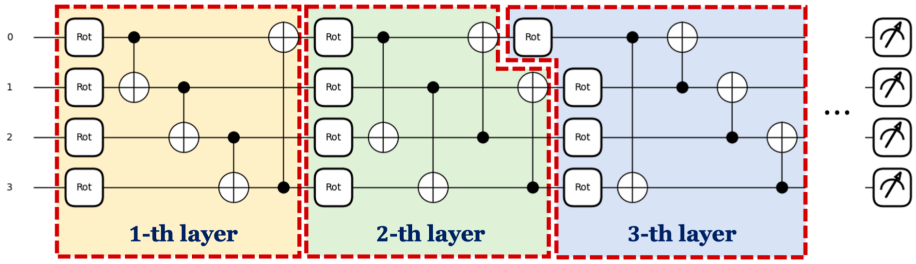

Fig. 7: Quantum circuit for strongly entangling quantum layer.

4 Experiment and 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 [7], a library for hybrid neural networks, was used, and the 'default.mixed' 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-[4]`. The following is the details of Table 1.

- **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

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

	S-DES	S-AES	S-PRESENT-[4]
Quantum embedding	Amplitude		
Quantum layer	Strong entangling		
N_{qubit}	4		
N_{qc}	1	4	4
N_{ql}	15	10	10
$N_{Rotation}$	180	480	480
N_{Params}	457	777	25393
N_{Data}	1300	2500	9000
Rounds	2	2	3
Epoch	25	25	20
Test accuracy	98%	99%	94%

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 rich rotation, as mentioned in 3.2. Random quantum circuit or basic quantum circuit (entangled with adjacent qubits) can also be used, but as a result of our experiments, this quantum circuit achieved the most stable and best performance.
- **The number of qubits (N_{qubit}):** 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.
- **The number of quantum circuit (N_{qc}), layer (N_{ql}), rotation gate ($N_{Rotation}$):** In `S-DES`, 1 quantum circuit and 15 quantum layers were used, and 4 quantum circuits and 10 quantum layers were used in `S-AES` and `S-PRESENT-[4]`. N_{qc} means the number of times the same quantum circuit is repeatedly executed. If the 4-qubit circuit is executed 4 times, the number of qubits used is 4 instead of 16, and

N_{qc} is 4. However, the number of all gates for 4 iterations should be considered as quantum resource. The reason is that the rotation gate of the circuit has parameters, and the parameters are stored and combined into one vector and then input to the classical output layer, so the number of quantum parameters as many as the number of rotation gates must be learned. Considering N_{qc}, N_{qubit} and N_{ql}, the number of rotation gates can be obtained as in Equation 3. The reason why it is multiplied by 3 is that the rotation gate used in a strongly entangling circuit is a combination of three rotation gates (RZ+RY+RZ). In addition, CNOT is also quantum gate, but it doesn't have parameters. Therefore, $N_{Rotation}$ is equal to the number of quantum parameters for the parameterized circuit (not the entire quantum neural network). `S-DES`, which is a relatively simple cipher, required the least amount of quantum resources, and the same number of quantum resources was used in `S-AES` and `S-PRESENT-[4]`.

- **The number of parameters:** N_{Params} is the number of parameters of the entire neural network combined with quantum and classical network. As mentioned earlier, since the classical layers have parameters, N_{Params} and quantum parameters depends on the number of neurons and the number of layers. The smallest 457 parameters were used in `S-DES`, and 777 parameters were used in `S-AES`. `S-PRESENT-[4]` requires the same number of quantum resources as `S-AES`, but more classical parameters are used, so N_{Params} is larger than `S-AES`.
- **The number of data and rounds:** For the 2-round `S-DES`, 1000 training data, 200 validation data, and 100 test data were used, and for the 2-round `S-AES`, 2000, 400, and 100 data were used for training, validation and test, respectively. In 3-round `S-PRESENT-[4]`, we used 7000 training data, 1900 validation data, and 100 test data.
- **Epoch:** We used 25 epochs for `S-DES` and `S-AES`. When more than 25 epochs were used, little convergence was performed and the overall accuracy was not significantly affected. For `S-PRESENT-[4]`, in 20 epochs, loss was reduced enough.
- **Test accuracy:** Finally, for the test data of `S-DES`, `S-AES` and `S-PRESENT-[4]`, accuracies of 98 %, 99 % and 94 % were achieved, respectively. When other values were used as input differences, the accuracy was barely over 0.5, or lower than our results, but as a result of using the correct input differences, we 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, Table 3, Table 4, and Table 5 show the comparison result for `S-DES`, `S-AES` and `S-PRESENT-[4]`. 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.

Result for `S-DES` 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.

Table 2: Comparison between classical and quantum classical neural networks for `S-DES` (25 epochs, 1000 data).

Target	`S-DES` (Classical)	`S-DES` (Quantum)
Tr	96	97
Val	97	97
Ts	96	98
N_{Params}	641	457

Result for `S-AES` In Table 3 and 4, we tested 1000 and 2000 data for 25 epochs. In (25 epochs, 1000 data) case, the accuracy of the quantum version achieved 18% higher accuracy than the classical version. And in (25 epochs, 2000 data) 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 other words, a larger quantum benefit (a significant improvement in accuracy) can be achieved when the less data is used. Also, the quantum version used fewer parameters than the classical neural distinguisher. In 1000 data, the number of parameters was reduced by about 43%, and in 2000 data it was reduced by 29%. Therefore, it can be seen that by using a quantum neural network, higher accuracy can be achieved despite using fewer parameters.

Result for `S-PRESENT-[4]` Table 5 shows the accuracy when training 20 epochs using 9000 data for `S-PRESENT-[4]`. Compared to the classical

Table 3: Comparison between classical and quantum classical neural networks for `S-AES` (25 epochs, 1000 data).

Target	`S-AES` (Classical)	`S-AES` (Quantum)
Tr	68	92
Val	75	86
Ts	65	83
N_{Params}	1089	617

Table 4: Comparison between classical and quantum classical neural networks for `S-AES` (25 epochs, 2000 data).

Target	`S-AES` (Classical)	`S-AES` (Quantum)
Tr	92	100
Val	99	99
Ts	98	99
N_{Params}	1089	777

method, the test accuracy is increased by 7% and the number of parameters is reduced by 32%. Also, 3-round `S-PRESENT-[4]` was tested, 1-round more than the other two ciphers, and more data and parameters were used because the plaintext size was longer and more complex than `S-DES`.

Table 5: Comparison between classical and quantum classical neural networks for `S-PRESENT-[4]` (20 epochs, 9000 data).

Target	`S-PRESENT-[4]` (Classical)	`S-PRESENT-[4]` (Quantum)
Tr	84	99
Val	80	96
Ts	87	94
N_{Params}	37377	25393

4.3 Quantum Advantage

Table 6 shows the comparison of quantum advantage (compared with classical method) for `S-DES`, `S-AES` and `S-PRESENT-[4]`. $I(Tr)$ means the improvement rate in training accuracy obtained by using a quantum neural network, and $I(Val)$ and $I(Ts)$ mean the improvement rate in validation and test. $R(N_{Params})$ is the reduction rate of the parameter obtained by using a quantum neural network.

Table 6: Comparison of quantum advantage for S-DES, S-AES and S-PRESENT-[4].

Target	S-DES	S-AES	S-PRESENT-[4]
$I(Tr)$	1%	8%	15%
$I(Val)$	0%	0%	16%
$I(Ts)$	2%	1%	7%
$R(N_{Params})$	29%	29%	32%

S-DES and S-AES achieved similar levels for $I(Ts)$ and $R(N_{Params})$. It can also be seen that the quantum benefit of S-PRESENT-[4] is greater because the $I(Ts)$ and $R(N_{Params})$ of 3-round S-PRESENT-[4] are large compared to 2-round S-DES and S-AES. In addition, in case of S-AES, a greater quantum advantage can be obtained when the number of data is smaller for the same epoch and round. This shows that the quantum method converges better (faster) than the classical method when less data is used. However, when the data is sufficiently increased, classical method and quantum method achieve similar accuracy. Therefore, we confirmed that for the same epoch, a larger quantum advantage occurs when using more complex data or less data. The reason for higher accuracy while using fewer parameters and fewer data is thought to be because qubit has a wider and more sophisticated data representation than classical bits.

5 Conclusion and Future Work

In this work, we implemented the first neural distinguisher for simplified block ciphers based on quantum-classical hybrid neural networks. The neural distinguisher is the task of classifying ciphertext from random data using differential characteristics, which can be developed into a key recovery attack. Our work obtained higher accuracies (2% for S-DES, 18% (1000 data) and 1% (2000 data) for S-AES, 7% for S-PRESENT-[4]) than classical neural distinguisher. In addition, it required a reduced number of parameters by 29% to 32% compared to the classical method. That is, we designed a distinguisher with quantum advantage in terms of parameters and accuracy by using a quantum neural network. As a future work, we will design a distinguisher for other ciphers and multiple input differences.

6 Data Availability

Source codes of proposed method are available in https://github.com/khj1594012/QND.

References

1. Abiodun, O.I., Jantan, A., Omolara, A.E., Dada, K.V., Mohamed, N.A., Arshad, H.: State-of-the-art in artificial neural network applications: A survey. Heliyon **4**(11) (2018) e00938 3
2. 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 4
3. Albrecht, M.R., Leander, G.: An all-in-one approach to differential cryptanalysis for small block ciphers. In: International Conference on Selected Areas in Cryptography, Springer (2012) 1–15 9
4. Baksi, A.: Machine learning-assisted differential distinguishers for lightweight ciphers. In: Classical and Physical Security of Symmetric Key Cryptographic Algorithms. Springer (2022) 141–162 7, 11
5. Beer, K., Bondarenko, D., Farrelly, T., Osborne, T.J., Salzmann, R., Scheiermann, D., Wolf, R.: Training deep quantum neural networks. Nature communications **11**(1) (2020) 1–6 5
6. Benamira, A., Gerault, D., Peyrin, T., Tan, Q.Q.: A deeper look at machine learning-based cryptanalysis. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer (2021) 805–835 7
7. Bergholm, V., Izaac, J., Schuld, M., Gogolin, C., Alam, M.S., Ahmed, S., Arrazola, J.M., Blank, C., Delgado, A., Jahangiri, S., et al.: Pennylane: Automatic differentiation of hybrid quantum-classical computations. arXiv preprint arXiv:1811.04968 (2018) 5, 15
8. Chen, Y., Yu, H.: A new neural distinguisher model considering derived features from multiple ciphertext pairs. IACR Cryptol. ePrint Arch. **2021** (2021) 310 7
9. Daemen, J., Rijmen, V.: The design of Rijndael. Volume 2. Springer (2002) 1
10. DiVincenzo, D.P.: Quantum gates and circuits. Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences **454**(1969) (1998) 261–276 4
11. Farhi, E., Neven, H.: Classification with quantum neural networks on near term processors. arXiv preprint arXiv:1802.06002 (2018) 5
12. Fingerhuth, M., Babej, T., Wittek, P.: Open source software in quantum computing. PloS one **13**(12) (2018) e0208561 5
13. Gohr, A.: Improving attacks on round-reduced speck32/64 using deep learning. In: Annual International Cryptology Conference, Springer (2019) 150–179 7
14. 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 4
15. Haykin, S.: Neural networks and learning machines, 3/E. Pearson Education India (2009) 3
16. Heys, H.M.: A tutorial on linear and differential cryptanalysis. Cryptologia **26**(3) (2002) 189–221 6
17. Hou, Z., Ren, J., Chen, S.: Cryptanalysis of round-reduced Simon32 based on deep learning. Cryptology ePrint Archive (2021) 7
18. Kaplan, M., Leurent, G., Leverrier, A., Naya-Plasencia, M.: Quantum differential and linear cryptanalysis. arXiv preprint arXiv:1510.05836 (2015) 2
19. Kariya, A., Behera, B.K.: Investigation of quantum support vector machine for classification in nisq era. arXiv preprint arXiv:2112.06912 (2021) 8

20. Killoran, N., Bromley, T.R., Arrazola, J.M., Schuld, M., Quesada, N., Lloyd, S.: Continuous-variable quantum neural networks. Physical Review Research **1**(3) (2019) 033063 5
21. 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 8
22. Knudsen, L.R., Robshaw, M.: The block cipher companion. Springer Science & Business Media (2011) 1
23. Musa, M.A., Schaefer, E.F., Wedig, S.: A simplified aes algorithm and its linear and differential cryptanalyses. Cryptologia **27**(2) (2003) 148–177 8, 9
24. Ooi, K., Vito, B.C.: Cryptanalysis of s-des. Cryptology ePrint Archive (2002) 8, 9
25. Petneházi, G.: Recurrent neural networks for time series forecasting. arXiv preprint arXiv:1901.00069 (2019) 4
26. Yadav, T., Kumar, M.: Differential-ml distinguisher: Machine learning based generic extension for differential cryptanalysis. In: International Conference on Cryptology and Information Security in Latin America, Springer (2021) 191–212 2, 7