Quantum Artificial Intelligence on Cryptanalysis

Hyunji Kim[1], Sejin Lim[1], Anubhab Baksi[2], Dukyoung Kim[1], Seyoung Yoon[1], Kyungbae Jang[1], Hwajeong Seo[1][0000−0003−0069−9061]

[1] Hansung University, Seoul, South Korea;
khj1594012@gmail.com,dlatpwls834@gmail.com,dudejrdl123@gmail.com, sebbang99@gmail.com,starj1023@gmail.com, hwajeong84@gmail.com
[2] Nanyang Technological University, Singapore; anubhab001@e.ntu.edu.sg

Abstract. With the recent development of quantum computers, various studies on quantum artificial intelligence technology are being conducted. Quantum artificial intelligence can improve performance in terms of accuracy and memory usage compared to deep learning on classical computers. In this work, we proposed an attack technique that recovers keys by learning patterns in cryptographic algorithms by applying quantum artificial intelligence to cryptanalysis. Cryptanalysis was performed in the current practically usable quantum computer environment, and this is the world's first study to the best of our knowledge. As a result, we reduced 70 epochs and reduced the parameters by 19.6%. In addition, higher average BAP (Bit Accuracy Probability) was achieved despite using fewer epochs and parameters. For the same epoch, the method using a quantum neural network achieved a 2.8% higher BAP with fewer parameters. In our approach, quantum advantages in accuracy and memory usage were obtained with quantum neural networks. It is expected that the cryptanalysis proposed in this work will be better utilized if a larger-scale stable quantum computer is developed in the future.

Keywords: Key recovery attack, Quantum Neural Networks, Quantum cryptanalysis.

1 Introduction

In October 2019, Google developed a 54-qubit quantum processor Sycamore with a significantly reduced error rate, and at the same time published a paper related to achieving quantum supremacy [1]. They proved for the first time in the world that quantum computers can surpass the performance of existing supercomputers. This is the first result of verifying the practicality of quantum computers, which were considered unlikely to be realized until now, and provides a foundation for researchers around the world to focus on preparing for the upcoming quantum computer era without questioning the development of quantum computers anymore. Quantum computers have the ability to perform complex calculations in polynomial time that classical supercomputers could not perform for specific problems. Therefore, by using quantum computers, it is possible to solve difficult problems that have been impossible due to high computational load in cutting-edge fields such as artificial intelligence, simulation, and big data processing.

Recently, next-generation attack techniques for cryptanalysis such as the Grover algorithm and deep-learning-based attacks are being studied on quantum computers. Cryptanalysis using deep learning uses a heuristic technique that finds regularities contained in the randomness of cryptographic algorithms from a large amount of data. However, in cryptography with an infinite number of cases, approaches such as deep learning inevitably face limitations of high computational load and memory usage. Therefore, as a solution to this problem, artificial intelligence on quantum computers has been actively researched recently. Quantum artificial intelligence is a technology that combines quantum computers and deep learning [2]. By using the characteristics of quantum computers, it is possible to improve the accuracy and data storage space.

If cryptanalysis is attempted using quantum neural network technology, it is believed that the limitations of deep learning-based cryptanalysis that have been studied so far can be overcome. However, since this field is the latest field, research on this has not been conducted yet. In this work, we would like to confirm for the first time in the world cryptanalysis using quantum artificial intelligence.

2 Related Works

2.1 Classical Neural Networks

Artificial neural networks are learning algorithms inspired by neural networks in biology. 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 Computing and Quantum Neural Networks

Unlike the bits of classical computers, quantum computers use qubits based on quantum mechanical phenomena, resulting in fast computation speeds. Therefore, it can overcome the computation limitations of classical computers and is used for cryptanalysis and artificial intelligence.

Quantum Computing Qubits in quantum computers have properties of superposition and entanglement.

- **Superposition:** Superposition is a state that can have a state of 0 and 1 at the same time with probability. Bit of classical computers can only represent the value 0 or 1, but a qubit can represent any value on the qubit's sphere (Bloch sphere). A value on the qubit's sphere (if it is not 0 or 1) is a state with a probability of being 0 and 1, so it is a superposition state. Thus, while a classical computer can represent n values with n bits, a quantum computer can simultaneously represent 2^n values with n qubits. Because of these characteristics, it is possible to calculate all cases at the same time, so the calculation speed is faster than that of classical computers.
- **Entanglement:** Also, since qubits have entanglement properties, a change in the state of one qubit affects the states of other qubits. That is, qubits can be entangled using quantum gates, and through this, each qubit can be interconnected. this characteristic is important in quantum neural networks.

Quantum computers use quantum gates, similar to logic gates in classical computers, to change and control the state of qubits. A quantum gate can be represented as a matrix, and the state of a qubit can be changed by performing a matrix multiplication between the state of a qubit and the quantum gate. Figure 1 shows representative quantum gates and their respective characteristics are as follows.

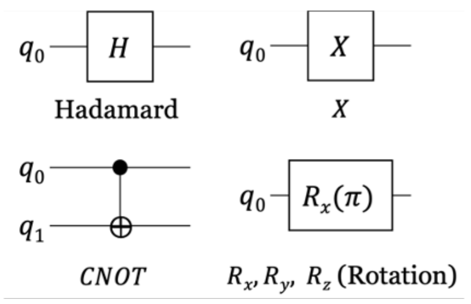


Fig. 1: Representative quantum gates.

- **Hadamard gate:** It operates on a single qubit and changes the target qubit to a superposition state.
- **X gate:** This inverts the state of a single qubit. If the state of a qubit is 1, it becomes 0, and if it is in a superposition state, it changes the probability of the qubit.
- **CNOT gate:** It operates on two qubit inputs. One qubit is the control qubit and the other is the target qubit. If the value of the control qubit is 1, the value of the target qubit is inverted, and if the value of the control qubit is

0, the state of the target qubit is maintained. That is, the value of the target qubit is changed according to the value of the control qubit, and the value of the control qubit is maintained.
– **Rotation gate:** It performs rotation operations on single qubits. The rotation gate takes the rotation angle (θ) as a factor and rotates the qubit about the x, y, or z axis by the input θ. There are Rx, Ry, and Rz gates along the base axis. If the value input to the Rx gate is π, it rotates by π with the x-axis as the rotation axis. In addition, there are controlled rotation gates (CRx, CRy, CRz) that rotate according to the state of the control qubit by targeting two qubits.

Quantum Neural Network Quantum artificial intelligence learns the characteristics of data based on quantum mechanical phenomena, enabling probabilistic inference like classical artificial intelligence. Quantum neural network can be implemented as a quantum circuit and operated on a quantum computer. As mentioned above, quantum mechanical phenomena include entanglement and superposition. Quantum artificial intelligence (quantum neural network) uses quantum circuits configured to learn based on these two quantum mechanical phenomena. First, entanglement, which can create correlations between two or more qubits, is used in quantum neural networks. Just as each neuron is connected by a weight in a classical neural network, each qubit is connected through entanglement in a quantum neural network. In addition, a quantum circuit for quantum artificial intelligence can be configured by setting rotation and entanglement after superposition. In addition, qubits can represent all points on the Bloch sphere, unlike classical bits that can only represent 0 and 1. Thus, qubit can gain quantum advantages based on the fact that it can express a wider range of values than classical bits [3]. In order to implement such quantum artificial neural network, quantum circuit must be designed, and rotation gates and entanglement are mainly used to implement similarly to existing neural networks. There are no rules for designing quantum neural networks, and an optimal circuit configuration can be obtained through several experiments. In addition, although a quantum neural network can operate in the Noisy Intermediate Scale Quantum era (NISQ), it is difficult to correct errors, so many calculation errors occur and many quantum resources cannot be used. Therefore, hybrid methods combining classical neural networks and quantum neural networks are now more stable in terms of performance. In addition, in order to construct and execute a quantum neural network, it is necessary to design a quantum circuit in consideration of various factors.

Quantum-only Neural Networks There are two types of quantum neural networks. There is a quantum-only neural network constructed using only quantum resources and a quantum-classical hybrid neural network that uses a combination of classical and quantum neural networks. A typical quantum-only neural network is the Quantum Support Vector Machine (QSVM) [4]. This is a quantum circuit implementation of the nonlinear kernel operation of the Support Vector

Machine (SVM) [5], a machine learning algorithm that can operate on a classical computer. In other words, QSVM is a supervised machine learning algorithm that finds the optimal boundary between data on a quantum computer. It has the advantage of being able to process high-dimensional data better than classical SVMs, so it has benefits for kernel optimization. It consists of a hadamard gate for superposition, a rotation gate for rotation, and a CNOT gate for entanglement. Here, a nonlinear kernel $((\pi - x) \cdot (\pi - y))$ used in classical SVM is used as the rotation angle of the rotation gate. In other words, since the rotation angle (parameter) changes as learning progresses, the output of the entire circuit changes as the qubit rotates by a different angle each time. The optimization function used for QSVM is SPSA (simultaneous perturbation stochastic approximation). In the QSVM training process, these quantum circuits are used to perform supervised learning so that the labels are finally classified as correct.

Quantum-classical Hybrid Neural Networks The quantum-classical hybrid method [6,7] combines a quantum circuit and a classical neural network and uses the quantum circuit as one of the layers of the entire neural network. Figure 2 shows the process of quantum-classical hybrid neural networks. The quantum circuit design for this is similar to the quantum-only method. The overall structure consists of a part for data embedding and a circuit for learning (Parameterized Quantum Circuit, PQC). And it can be configured to utilize rotation and entanglement and superposition properties. The training process consists of calculating the loss and adjusting the parameters based on the loss value, like in a classical neural network. So the whole process is the same as for classical neural networks, but with quantum circuits as one layer. In the process of executing a quantum circuit, like a quantum-only neural network, the state of a qubit is measured after sequentially passing through all the gates constituting the circuit. Finally, the expected value of the quantum circuit is obtained and used as the final output of the quantum layer. The value can be input to the next classical layer. If a quantum layer is used as an output layer, the loss is calculated and then the existing parameter (the rotation angle of the qubit) is changed through a parameter update method (e.g. parameter shift). As mentioned above, there are no strict rules for designing quantum circuits, and the designer must find an optimal circuit and rotation angle (parameters) that can be learned well. Also, since it is a hybrid method, it is used with libraries for classical neural networks such as Tensorflow [3] and Pytorch [4]. And their loss function (e.g. binary cross-entropy, categorical cross-entropy, mean squared error (MSE)) and optimization function (e.g. Adam, SGD) provided by the library can be applied.

Considerations for Implementing Quantum Neural Network We have used quantum neural networks in this work, and we think that the following points should be considered.

[3] https://www.tensorflow.org/
[4] https://pytorch.org/

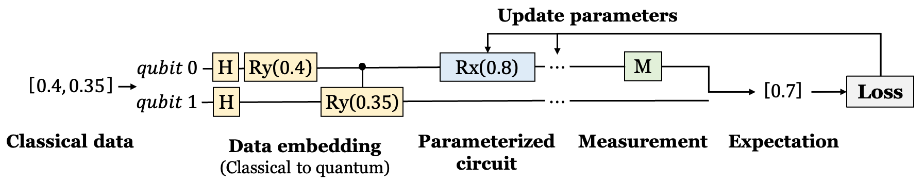


Fig. 2: The process of quantum-classical hybrid neural networks.

- **Quantum computer supporting quantum circuit:** Quantum circuits are used in quantum neural networks, and CNOT gates, rotation gates, and hadamard gates are mainly used. However, since the circuit cannot be executed in a quantum computer that supports quantum annealing, the quantum neural network does not operate. Therefore, in order to operate a quantum neural network, a general-purpose quantum computer capable of running quantum circuits must be used, and quantum gates (e.g. CNOT, rotation, hadamard) must be supported.
- **Stable device:** Quantum neural networks can be run on real quantum hardware or simulators provided by quantum programming platforms. However, current quantum processors cannot use many qubits for public people. Also, training and inference can be difficult because error correction is difficult in NISQ. Therefore, for now, we need to use a simulator supported by quantum computing platform. The simulator must be reliably run circuits with a depth of 30 or more, and provide an appropriate number of qubits (10-qubits).
- **Support for classical neural network frameworks and related libraries:** Since it is currently a noisy intermediate scale quantum computer, it is recommended to use a quantum-classical hybrid neural network in consideration of quantum resources, time required for training, and stable performance. Indeed, many studies have adopted this method. To do this, it is necessary to be able to use classical neural network frameworks such as Tensorflow and Pytorch with quantum circuits. There are quantum computing frameworks supporting classical neural network frameworks and related libraries. They provide loss functions, optimization functions of classical method, parameter updates of quantum circuits, and weight updates of classical method.

Research trends on quantum neural networks We briefly reviewed research [8,9,10,11,12,13] using quantum neural networks. There are quantum-only method and quantum classical hybrid neural network. Most of the studies were conducted within the last two years and continue to be actively studied.

Studies on performance improvement for well-known datasets such as the MNIST dataset or classification work for data with small dimensions have been mainly performed. These works have obtained quantum advantages (reduced number of layers, improved accuracy, etc.) compared to classical neural networks. However, most of the studies did not use many qubits, and since ibm's quantum hardware can only be used up to 7-qubit by general users, all studies using the quantum computer used 4 to 5-qubits. Research using 8 to 14-qubits allowed experiments on a large number of qubits using a classical simulator rather than actual hardware. In addition, there are many studies that add noise to a simulator or operate it in an actual quantum computer to consider the actual noise of a quantum computer.

As quantum hardware, IBM's quantum computers are widely used, and in the case of hybrid methods, Pennylane libraries are mostly used. In addition to these studies, there were many cases in which NISQ used a more stable hybrid method. In addition, various models such as quantum convolutional neural network (QCNN), quantum recurrent neural network (QRNN), and quantum generative adversarial network (QGAN) are being developed. However, it is difficult to use many qubits due to the lack of quantum resources, so classification tasks using relatively simple models are mainly performed.

2.3 Comparison and analysis of quantum computing environments and frameworks for quantum neural networks

There are various quantum frameworks that support quantum neural networks, including IBM Qiskit [5], Amazon Braket [6], Microsoft Azure [7], and Pennylane [8]. It is also possible to use simulators (running quantum programs on classical computers) and quantum hardware of other frameworks in a plug-in method, such as the Qiskit-braket plug-in and Pennylane-qiskit plug-in. In addition, for now, there are not many quantum resources available, it is possible to construct a hybrid neural network by combining it with Tensorflow or Pytorch. In this chapter, we analyze the quantum computer environment and software development kit (SDK) for the quantum neural networks.

D-wave Leap D-wave provides IDE and Jupyter notebook environment on the browser through Leap. It provides Python-based solutions for various types of problems, and sample codes are frequently updated as research is being actively conducted. About 48 sample codes are provided, and there are 4 samples related to machine learning. D-wave itself provides actual quantum hardware. (Advantage model supporting more than 5000 qubits and D-wave 2000Q supporting more than 2000 qubits.) Since D-wave supports more qubits than any other framework, we checked if it was applicable to our work. We obtained the QPU usage time by providing the github repository required by D-wave. As a service related to quantum neural networks, functions such as data point separation and classifier are provided in the form of modules, and they can be easily applied to specific tasks by modifying parameters. However, it is difficult to modify the entire structure for application to other tasks. We approached D-wave's QPU as a

[5] https://qiskit.org/
[6] https://aws.amazon.com/ko/braket/
[7] https://azure.microsoft.com/ko-kr/products/quantum/
[8] https://pennylane.ai/

plug-in method in the Amazon Braket framework without using their examples. As a result of operating the quantum neural network based on the quantum circuit using the device, an error occurred saying that the type of work performed was different from the work that the QPU of D-wave could perform. The work supported by D-wave's QPU is Quantum Annealing (QA). Quantum annealing has different characteristics from general-purpose quantum computers based on circuit architecture. Tasks to which quantum annealing techniques can be applied include optimization and sampling. Optimization is the problem of finding an optimal solution from a finite set of options, and sampling is the problem of improving model performance by providing state information of a model for a given parameter during machine learning. That is, it is a QPU suitable for parameter optimization of classical neural networks. To do our work, quantum circuits must be able to run. Therefore, the QPU of D-wave using the quantum annealing technique cannot be used in our work.

IBM Qiskit Qiskit, an open-source quantum software framework developed by IBM, provides a cloud service for quantum computing and a locally available quantum simulator. Qiskit can be implemented using Python and the quantum command QpenQASM. Qiskit can implement quantum programs using Python and QpenQASM, a quantum assembly language. Qiskit provides various functions necessary for quantum programming, focusing on quantum circuit-related functions, and provides more convenient visualization than other platforms. Analysis of the simulator, hardware, and quantum neural network library provided by Qiskit are as follows. Analysis of the simulator and hardware is the result of running the same circuit (circuit depth=30, shots=1024) in IBM Q Experience (cloud service).

- **Simulator:** Qiskit provides a quantum simulator and hardware for up to 127 qubits. But these hardware are only licensed to some users. The Qiskit's QPUs accessible to public users are provided up to 7 qubits. In addition, Qiskit provides a noisy quantum simulator that can simulate noise similar to real quantum devices. Qiskit provides simulators such as QASM and State vector that support up to 32 qubits, and can be used according to the purpose and situation.
  The QASM simulator works similarly to how it works in hardware, and the measurement probability depends on the number of shots. That is, if 90 times out of 100 are measured as 0, the probability that the value of the corresponding qubit is 0 is 0.9. In addition, high-quality noise modeling and various rotation gates are provided.
  The state vector simulator can observe the state vector of a qubit and returns a state vector of length 2^n for n-qubits. Therefore, it requires more time and memory compared to the QASM simulator. It supports noise modeling and various rotation gates.
  Table 1 shows the performance of the simulator according to the number of qubits. Qiskit said these simulators can provide up to 32 qubits. But we can see different results. They simulated 28 qubits using QASM, and 24 qubits

using a state vector simulator. Both simulators supported the quantum gates required for quantum neural networks, and the QASM simulator was faster than the state vector simulator.

Table 1: Performance of the Qiskit simulator according to the number of qubits. (units:s)

Qubit	10	20	28	29
Time (QASM)	0.047	0.085	4.144	Dead

Qubit	10	20	24	25
Time (Statevector)	0.395	119.092	2319.701	Dead

– **Hardware:** Qiskit provides a number of hardware (e.g. ibmq_manila and ibmq_nairobi). Running a quantum program using quantum hardware requires waiting in the task queue. After being allocated quantum hardware, the program is executed, and this waiting time differs depending on the time and type of hardware. There is also a device called fake hardware that imitates the operation of IBM quantum processor without using actual hardware. This provides noise simulation for 5 to 127 qubits.
  Table 2 shows the results for ibmq_manila and ibmq_nairobi. Both hardware were operated up to their maximum qubits. Execution time is the time excluding waiting time in the queue, and ibmq_manila showed faster speed for the same qubit.

Table 2: Performance of the Qiskit hardware according to the number of qubits. (units:s)

Qubit	3	5
Time (ibmq_manila)	5.6	6

Qubit	5	7
Time (ibmq_nairobi)	9.16	51.2

– **Library for quantum neural network:** Qiskit provides libraries and tutorials related to quantum neural networks. Typically, quantum-only methods such as Quantum Support Vector Classifier (QSVC) are provided. In addition, a quantum-classical hybrid neural network can be constructed by combining classical neural network frameworks such as Tensorflow, Pytorch, and Cirq with Qiskit quantum circuits. As an example, they provide sample code for a simple binary classification problem focused on presenting a quantum artificial intelligence architecture. In this sample, a circuit with a single qubit and no entanglement is used, and it can be easily changed to a circuit with entanglement. However, if the circuit is changed to use multiple qubits, the problem of calculating the state of each qubit as a single decimal

number arises. For example, if two qubits measure 1 and 0 respectively, it calculates the result as decimal 10. Libraries related to quantum neural networks can be used in local environments and other cloud environments, and are actively developed, so deletions and updates occur frequently.
Finally, using the sample code for binary classification mentioned above, we experimented with the maximum number of operable qubits in a quantum neural network. As a result, up to 27 qubits were operable. That is, it can use fewer qubits than general quantum circuits.

Amazon Braket Braket is a cloud service for quantum computing developed by Amazon. The Braket cloud service allows users to design quantum algorithms and perform quantum simulations. In addition, actual quantum resources and technologies of companies that provide quantum hardware can be used. Braket also offers a notebook format like Qiskit's Experience Q.

– **Simulator:** Local, State Vector (SV1), Density Matrix (DM1), and Tensor Network (TN1) simulators are provided. If you run all simulators locally, the maximum runtime is 6 hours. Local and SV1 simulators can perform state vector simulation for free and support general-purpose quantum circuits. The number of executable qubits of those simulators is 25-qubit and 34-qubit, respectively. The DM1 simulator provides 17 qubit, and uses a density matrix considering noise, so noise simulation is possible. Finally, the TN1 simulator using graphs is specialized for large-scale simulations and is suitable for regular circuit simulations such as quantum Fourier transforms. The circuit with 50 qubits and a depth of 1000 can be operated using TN1. Table 3 shows the execution time according to the number of qubits on the local simulator. The simulator can run the quantum circuit with 25 qubits, the maximum number of qubits supported.

Table 3: Performance of the Amazon braket's local simulator according to the number of qubits. (units:s)

Qubit	10	20	25
Time (Local simulator)	0.02	0.48	22.86

– **Hardware:** Braket provides real quantum hardware (e.g. IonQ, D-wave) that supports 8-qubit to 5760-qubit for a fee.
– **Library for quantum neural network:** Amazon Braket provides an example to implement a simple binary classifier by constructing a quantum circuit and loss function without using Tensorflow and Pytorch. And, we experimented about the maximum number of available qubits using this example. As a result, it was possible to operate up to 20 qubits. Most of the cases using Amazon Braket constructed a quantum-classical hybrid neural network using the Pennylane plug-in.

Microsoft Azure Quantum Azure Quantum is a service for quantum computing developed by Microsoft. You can use the Azure quantum service through the Quantum Development Kit, a quantum computer programming development tool provided by Microsoft. Models based on quantum circuits are written in Q, and tasks written in other languages (e.g. Qiskit, Cirq) can also be executed. Users can also work in Visual Studio, Visual Studio Code (local) and Azure Quantum (cloud). Azure quantum has a total of 8 providers that provide simulators and hardware.

– **Simulator:** ionq.simulator is IonQ's free cloud-based simulator, and all gates provided by quantum hardware are available. Quantinuum.hqs-lt-s2-sim is an emulator that is more realistic than a simulator (using a real physical model and noise model of the H1-2 hardware). Currently, accessible simulators provide 11 qubits to 29 qubits. In the case of ionq.simulator, it is written that up to 29 qubits is supported in the document, but in reality, only up to 22-qubit was available. In addition, the performance was different for each measurement, and the gap was very large.
– **Hardware:** Quantum hardware such as IonQ, Quantinuum, Pasqal, Rigetti, and qci are provided. Currently, accessible hardware provides 11 qubits to 20 qubits. An error occurred saying that Quantinuum's hardware supporting 20 qubits was unavailable, and only hardware with 12 qubits was available. The hardware was also unstable to run the circuit like ionq.simulator.
– **Library for quantum neural network:** Through the Microsoft.Quantum.MachineLearning library, models that can solve a simple binary classification problem are provided. However, since grammar and libraries have been updated until recently, errors such as import errors in modules exist even if the environment is set according to Azure Quantum's official documentation.

Pennylane Pennylane is a library designed for quantum-classical hybrid neural networks. In other words, it is a quantum neural network library, and can be used with most quantum frameworks such as Qiskit, Braket, Q, Cirq, and Rigetti. In addition, it can be combined with classical neural networks, so we can use the loss function and optimization function of Tensorflow and Pytorch.

– **Simulator:** Pennylane provides a default simulator, a C++-based accelerated simulator, and a density matrix-based noise simulator. In addition, simulators of other platforms such as Qiskit, Braket, Rigetti, and Cirq can be used in a plug-in method. Table 4 shows the execution time according to the number of qubits on the default simulator. Simulations of up to 29 qubits are possible. The execution speed is slower overall compared to other platforms as the number of qubits increases, but the same or faster up to 10 qubits.
– **Hardware:** Pennylane does not provide its own hardware, but devices from other platforms can be used through plug-in.
– **Library for quantum neural network:** Pennylane provides templates for embedding layers and templates for parameterized quantum layers used in

Table 4: Performance of Pennylane's default simulator according to the number of qubits. (units:s)

Qubit	10	20	25	28	29
Time (Default.qubit)	0.02	0.93	44.73	383.41	747.54

a hybrid quantum neural network. The embedding layer is a circuit for embedding classical data into quantum states, and Pennylane has 7 templates including amplitude embedding and angle embedding. The parameterized quantum layer is a circuit composed of rotation and entanglement gates for training, and 5 templates such as basic and random circuits are provided. Also, these quantum circuits can be converted into Tensorflow-Keras and Pytorch layers to combine classical and quantum layers. In addition, it provides various techniques necessary for training the quantum neural network, such as differentiating methods for backpropagation and simulators for automatic differentiation.
We investigated the number of available qubits using circuits provided by Pennylane. As a result, up to 16 qubits could be used. This is also less than the maximum number of qubits available in the general circuit.

2.4 Cryptanalysis using Classical and Quantum Neural Network

Table 5 shows the research of known-plaintext attacks using classical and quantum neural networks. In [14], they proposed key recovery attacks using a classical neural network for `S-DES`, round-reduced `SPECK` and `SIMON`. They experimented with attacks for bit keys and text-based keys. Both attacks were successful for `S-DES`. But, in round-reduced `SPECK` and `SIMON`, only the attack for text-based keys was successful. Since the text-based key has a total of 64 cases, it is not a 10-bit key space. Also, not all bits are used, and the probability of occurrence for each bit is different (i.e. the 1st bit is 0 with 80% probability). So, the text-based key is more vulnerable to the key recovery attack. Therefore, attacks on text-based keys were possible in round-reduced `SPECK` and `SIMON`, but this has limitations in terms of round reduction and text-based keys.

In [15], key recovery attacks using a classical neural network for `S-DES` and `S-AES` have been presented. They used bit keys. In the case of `S-DES` , higher accuracy was achieved than the key recovery probability in [14]. In addition, a key recovery attack for `S-AES` was attempted for the first time. For `S-DES` , it was possible to attack a 10-bit key, but there is a limitation in that it is impossible to attack a 16-bit key of `S-AES`.

In [16], a method of predicting plaintext with ciphertext for simplified `ASCON` (`S-ASCON`) using deep learning was proposed. The target cipher, `S-ASCON`, some processes are omitted from the full `ASCON` (i.e. finalization is omitted, and empty associated data is used). They predicted plaintext with 0.998% accuracy. However, since there are many constraints in the scenario, predicting plaintext will fail in an unconstrained real scenario.

In [17], they performed a key recovery attack on CAESAR using quantum support vector machine (QSVM), a quantum machine learning algorithm. However, due to the limitations of the current quantum computing environment, analysis of 2-bit and 3-bit keys was only possible.

Table 5: Comparison of existing cryptanalysis works (KR: Key recovery, PR: Plaintext recovery).

Ref.	Target cipher	Attack	Method
[14]	S-DES (Bit, Text), Round-reduced SPECK and SIMON (Text)	KR	Classical
[15]	S-DES (Bit), S-AES (Bit)	KR	Classical
[16]	S-ASCON (Bit)	PR	Classical
[17]	CAESAR (Bit)	KR	Quantum

3 Proposed Method

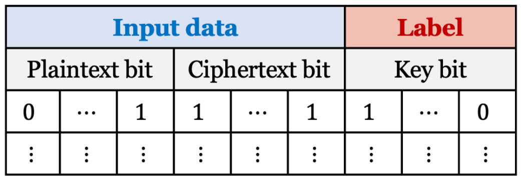
In this paper, we perform cryptanalysis on S-DES using quantum neural network. A quantum-classical hybrid neural network was implemented using the Pennylane library, and a key recovery attack for S-DES was performed. We used the basic simulator of Pennylane, and we also performed cryptanalysis considering the noise of the quantum computer using a noise simulator. Through this, we look at the quantum advantages for cryptanalysis and present the direction of cryptanalysis using a quantum neural network.

3.1 Dataset

Cryptanalysis based on a quantum hybrid neural networks also requires classical data like classical neural networks. Therefore, we implement S-DES, the target encryption algorithm, on a classical computer, and generate a dataset for plaintext, ciphertext, and used keys. Figure 3 shows the data set for cryptanalysis for S-DES. The plaintext, ciphertext, and key are all represented by bits, and the plaintext and ciphertext are concatenated. That is, the data in the dataset have dimensions (features) equal to twice the length of the plaintext. Also, the key bits are used as labels. Therefore, if the quantum hybrid neural network train this dataset, the used key bit can be recovered if only the plaintext and ciphertext pairs are known.

3.2 Quantum-classical Hybrid Neural Network

To construct a quantum neural network for cryptanalysis, quantum-only and hybrid methods can be used. However, quantum-only methods such as QSVM currently lack available quantum resources and are difficult to train from large



Input data						Label		
Plaintext bit			Ciphertext bit			Key bit		
0	···	1	1	···	1	1	···	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Fig. 3: Data Set configuration.

amounts of data. Therefore, a hybrid quantum neural network was constructed for stable and feasible cryptanalysis using the resources of the current quantum computer. For this, the Pennylane library and Tensorflow-Keras were used. Figure 4 is the system diagram, and the overall process is as follows.

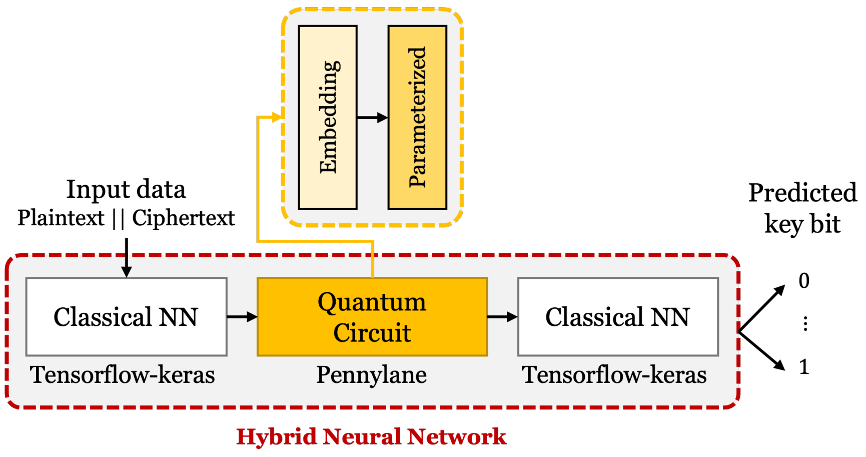


Fig. 4: Diagram of cryptanalysis using quantum-classical hybrid neural network.

1. The input data is input to the input layer (written in Tensorflow-Keras) of the quantum-classical hybrid neural network.
2. After passing through several classical layers (input layer, additional hidden layer (optional)), it is input to the quantum hidden layer.
3. The quantum hidden layer consists of the quantum circuit (using the Pennylane library), which consists of data embedding circuits and parameterized circuits (PQC). Therefore, after the outputs of the previous layer are input to the quantum hidden layer, they are processed as follows.
   (a) The input classical value is encoded into a quantum state through a data embedding circuit. The embedding circuits used in our work include amplitude embedding and instantaneous quantum polynomial (IQP) embedding methods.

   (b) After being encoded, each qubit changes its state as it passes through a parameterized quantum circuit (PQC). Then, at the end of the circuit, it is measured and determined as a classical value. For this, we used a random circuit and a strongly entangling circuit.

4. After the output of the quantum hidden layer is input to the classical output layer, the entire neural network predicts the key bits.
5. After the loss is calculated based on the predicted key bits, the parameters of the entire neural network (classical weights and rotation angles of quantum gates) are updated. For parameter update, existing methods are used in classical neural networks. And in quantum circuits, methods such as parameter shift and adjoint differentiation are used. As this process is repeated over and over again, it is trained.
6. After training is completed, data (plaintext and ciphertext pairs) is input to the trained hybrid neural network for inference. Then, the hybrid neural network can predict the key used by recognizing its characteristics.

Through this process, key recovery attacks using quantum neural networks become possible. In order to implement a quantum neural network for this, the number of qubits, the number of quantum layers, the embedding method, the type of parameterized circuit, and the data set must be set. However, since current quantum computers have resource limitations, we must select appropriately considering available quantum resources, performance, and training time. Table 6 shows details of each element. These are detailed below.

Table 6: Details of elements for the quantum circuit.

Elements of quantum circuit	Description
The number of qubits	4
Data embedding	Amplitude, IQP
Parameterized circuit	Strongly entangling circuit
The number of quantum layers	10

The number of qubits When embedding 2^n of data, angle embedding uses 2^n qubits, and amplitude embedding uses n qubits. We experimented with the maximum number of qubits executable in these two methods. For now, 16 qubits can be operated. If we use more than 17 qubits, memory shortage problems occur. In other words, all 28 qubits provided by Pennylane's simulator cannot be used, and 16 qubits can be used. However, it is not realistically easy to use all 16 qubits for training. Assuming all other things are equal, if the number of qubits increases n times, the circuit will take n times as long to run. Thus, an appropriate number of qubits must be used.

As a result of experiments with the number of qubits set to 2, 4, and 8 for our work, 2 qubits did not provide sufficient performance. In the case of 8-qubits, one epoch took too much time. So we couldn't get the result. 4-qubits also take a long time to learn (more than 20000 seconds for 1 epoch), but we used 4 qubits because cryptanalysis can be performed in a relatively realistic time.

Data Embedding Circuit For the training process described above, it is necessary to change from a classical layer to a quantum hidden layer. For this, classical data must be converted into quantum data. Figure 5 shows the embedding methods used in our work. There are other embedding methods, but we used the following two embedding circuits.

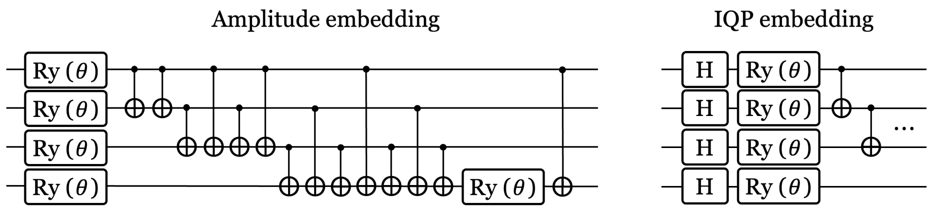


Fig. 5: Quantum circuit for data embedding.

– **Amplitude embedding:** Amplitude embedding is a method of expressing a vector representing the amplitude state of a qubit using input data. That is, through this embedding circuit, it is possible to express the amplitude of the state of a qubit corresponding to a point on the Bloch sphere. Also, rotation gates and CNOT gates for entanglement are used to make the amplitude embedding circuit. 2^n data can be embedded in n qubits. If 4 qubits are used, the amplitude vector of the qubits is represented by a data vector of length 16. Therefore, this method is suitable for current quantum neural networks where it is difficult to use many qubits.
– **IQP embedding:** This method uses Hadamard gates for superposition, and embeds data through rotation and entanglement in superposed qubits. This method can only embed n data in n qubits. In addition, since the number of CNOT gates for entanglement applied after the rotation gate can be adjusted by the user, the depth of the circuit can be adjusted. However, fewer data features can be reflected when using the same number of qubits as amplitude embedding. Therefore, if we want to embed more data, we need to increase the number of qubits. However, the number of qubits that can be operated in the current quantum neural network (with Pennylane) is up to 16 regardless of the embedding method. If embedding for 16 data requires 16 qubits, and since Hadamard gates are also used for each qubit, the circuit takes longer to run. Therefore, data embedding using superposition

is possible, but currently, it is difficult to reflect high-dimensional data. This is also a problem directly related to the performance of the quantum neural networks. Using low-dimensional data can degrade performance for complex tasks.

Parameterized Circuit After the data has been successfully embedded, a parameterized quantum circuit is required for training. Parameterized quantum circuits are generally composed of rotation and entanglement, and the rotation angles of the rotation gate are repeatedly updated during the training process. In other words, the rotation angle (parameter) of the circuit is changed like the weight of a classical neural network. The embedding circuit is not a parameterized quantum circuit because the input data is embedded whenever training proceeds. Pennylane provides various circuit samples to configure parameterized circuits, but user-defined circuits can also be used. In this work, we used a strongly entangling circuit as shown in Figure 6. In this circuit, each qubit is regularly and very strongly entangled. Therefore, it is designed so that all qubits can be evenly entangled as a quantum layer (meaning a layer in a parameterized circuit, not a quantum hidden layer) is added. In addition, the rotation gates used in this circuit are a combination of three general rotation gates (Rot gate is combined in the order of Rz, Ry, and Rz). Therefore, three times the parameter is required compared to the general rotation gate. As a result, the more layers are stacked, the more rich rotation and strong entanglement are possible.

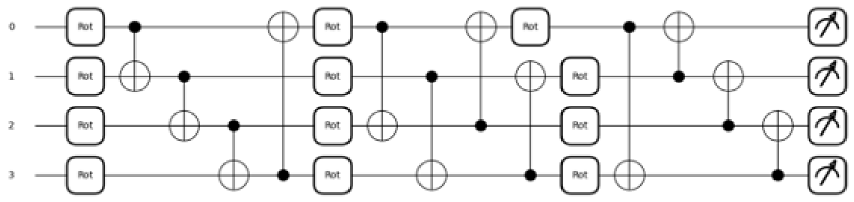


Fig. 6: Strongly entangling circuit for the parameterized circuit.

Moreover, these parameterized quantum circuits consist of several layers of quantum layers. A quantum layer may have a pattern according to a rule or may be random. As the number of these quantum layers increases, the depth of the circuit increases, and the number of parameters increases, so it must be selected appropriately. As a result of our experiment, cryptanalysis was possible with 5 quantum layers up to the 5-bit key of `S-DES`, but more layers were required from the 6-bit key. For a 10-bit key, 10 quantum layers were required. And when 10 to 15 quantum layers were used, it took about 20000 to 30000 seconds for one epoch. We used 10 quantum layers because increasing the number of quantum layers to improve performance significantly increases the training time.

Combining Classical and Quantum Neural Network Quantum hidden layers (data embedding + parameterized quantum circuits) should be combined with classical layers to be constructed in the hybrid method. Figure 7show the overall structure and the process of combining the two types of layers.

First, the input layer is a classic layer, and the number of nodes is twice the block size. That is, each bit of plaintext and ciphertext is assigned to each node. After passing through this input layer, it is input to the hidden layer, and a quantum circuit can be used as one of the hidden layers. For this, classical layers and quantum circuits must be combined, so neurons from the previous classical layer are divided and input into each quantum circuit. For example, in the case of a circuit using 4-qubit and amplitude embedding, 16 values can be embedded in a quantum circuit at a time. In this case, if the previous layer has 64 output neurons, a total of 4 circuits are needed. However, our implementation does not create 4 independent circuits, but actually has a structure in which the same circuit is repeated 4 times. Therefore, 4 qubits are used instead of 16 qubits, and a circuit using 4-qubits is executed 4 times, 64 weights can be processed. Based on these design principles, experiments were conducted on the number of quantum circuits, and we used four circuits in consideration of the training time and the number of available resources. Through this, 16 weights are embedded in each circuit, and the output of each circuit is 4 measured values for 4 qubits. After the circuit has been run 4 times, there are a total of 16 measured values, and the corresponding outputs are fed into the classical output layer. The last output layer predicts 10 key bits. And this neural network calculates the loss through the mean squared error (MSE) function. Afterward, the parameters of all classical layers and quantum circuits are updated.

In this way, classical layers and quantum circuits can be combined. And Pennylane provides a function to change a quantum circuit into a TensorFlow-Keras layer in order to properly train a hybrid neural network that combines two types of layers.

4 Evaluation

In this section, various experiments were conducted for cryptanalysis for `S-DES` using quantum neural networks, and the results were compared and analyzed. For three methods (classical neural network, quantum neural network using amplitude embedding and IQP embedding), the accuracy for each bit according to epoch was compared. Then, by analyzing the results, we present a direction for cryptanalysis using quantum neural networks.

4.1 Experiment Environment and Data Set

Experiment Environment 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.

In Table 7, each library was compared based on the contents mentioned in Section 2.3. Except for Pennylane, other libraries had minor errors, or circuit execution was unstable. However, Pennylane satisfies all the conditions and is easy to develop. Based on this comparison, we chose Pennylane. Also, we experimented with and without noise using simulators provided by Pennylane.

Table 7: Comparison of quantum computing frameworks.

	D-wave Leap	IBM Qiskit	AWS Braket	Microsoft Azure Quantum	Pennylane
Support quantum circuits	X	O	O	O	O
Rotation gate	X	O	O	O	O
The number of qubits for quantum neural network	X	27	20	Error	16
Related Libraries	X	O	X	O	O
Support hybrid method	X	Pytorch	Python	Python, C#	Tensorflow, Pytorch

Data Set Since the data for cryptanalysis is high-dimensional and the labels are 10 bits, it is unreasonable to train a lot of data using current quantum neural networks. Therefore, in this work, 35000 data were used considering the training time. It is divided into 19950 training data, 14000 validation data, and 1050 test data.

4.2 Analysis of Bit Accuracy by Epoch

We compared the results of classical and quantum neural networks to confirm the quantum advantage. To compare under the same conditions, the model structure of [15] was used, and the same data set as the quantum neural network was used. But epochs are different. The reason is that it takes a lot of time to train quantum neural networks, so it is practically difficult to repeatedly experiment up to 100 epochs. Therefore, the quantum neural network was trained for only 30 epochs. Here, the accuracy of each bit is called BAP (Bit Accuracy Probability). In addition, a BAP of 0.5 or more and less than 0.6 is a secure bit, 0.6 or more to less than 0.8 is a normal bit, and 0.8 or more is a vulnerable bit.

Table 8, Table 9, and Table 10 show the BAP analysis by epoch for three neural network types. For classical neural networks (See Table 8), the 4th, 5th and 7th bits in every epoch are secure bits. And the 6th and 9th are vulnerable bits. In addition, the 1st and 2nd bits were safe bits at 15 epochs, but after training progressed, the BAP improved and were excluded from the secure bits. As a result of training for 100 epochs, the 4th, 5th, and 7th are secure bits, and

Table 8: BAP by epoch (Classical neural network).

Epoch	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th	Avg.
15	59	59	65	53	54	80	54	56	80	72	63.2
20	60	60	65	51	54	81	51	56	82	72	63.2
25	62	60	65	53	55	81	54	56	83	70	63.9
30	61	60	67	56	54	80	52	59	83	74	64.6
100	60	66	69	57	56	81	53	60	86	80	66.8

Table 9: BAP by epoch (Quantum neural network (Amplitude)).

Epoch	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th	Avg.
20	53	52	65	52	54	80	53	56	83	71	61.9
25	61	61	65	55	52	80	54	56	82	74	64.0
30	64	71	71	56	51	80	56	60	85	80	67.4

the 8th is excluded from the secure bit. Also, in 100 epochs, the 10th bit along with the 6th and 9th bits were detected as vulnerable bits.

Next, Table 9 shows the results of the quantum neural network method (Amplitude embedding) that embeds and trains data through entanglement and rotation. For all epochs, the 4th, 5th and 7th are secure bits, and the 6th and 9th are vulnerable bits. The 10th bit, which was a normal bit, was detected as a vulnerable bit in 30 epochs, and the 8th bit was excluded from the secure bit. his is a result that can be obtained by training 100 epochs when using a classical neural network. Looking at the average BAP (Avg.) over 30 epochs, this method using amplitude embedding achieved a 2.8% higher BAP than the classical neural network. Also, the result for 30 epochs using the amplitude method is 0.6% higher than the maximum performance (100 epochs) of the classical neural network.

In addition, from 20 epochs to 30 epochs, the BAP increase is larger than that of the classical neural network method. The amplitude method increased BAP by 2.1% (20 25 epochs) and by 3.4% (25 30 epochs), and the classical neural network increased BAP by 0.7% in both sections. Therefore, compared to the classical method, the amplitude method obtained the same number of vulnerable bits and secure bits in fewer epochs and was able to achieve a higher BAP.

Table 10: BAP by epoch (Quantum neural network (IQP)).

Epoch	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th	Avg.
20	53	63	55	56	48	79	54	52	83	54	59.7
25	52	55	59	53	51	79	51	52	82	57	59.1
30	58	65	60	55	54	79	51	57	84	75	63.8

Table 10 shows the analysis results of a quantum neural network using IQP embedding that encodes the data after creating a superposition using the Hadamard gate. For all epochs, the 1st, 4th, 5th, 7th, and 8th bits are the secure bits, and the 9th bits are the vulnerable bits. Compared to the two methods (classical and quantum amplitude), the number of secure bits is larger and the number of vulnerable bits is smaller. Since the number of features that can be reflected in training is smaller than the amplitude method (quantum), BAP is lower than other methods. In particular, the analysis for the 5th bit failed at 20 epochs. And, in 30 epochs, the average BAP was lower than the other methods, and the 6th bit was not detected as a vulnerable bit only in this method. This result means the training was not sufficient.

Experimental results for the above three methods show the following common results.

1. For all epochs, the common secure bits are the 4th and 7th, and the common vulnerable bit is the 9th. Thus, all results have a similar trend. In particular, the 9th bit that achieves a high BAP even in an IQP method with low performance due to limitations of quantum resources is a very vulnerable bit.
2. As the training progresses, the prediction probability of the neural network increases, so the BAP according to the epoch increases overall. Accordingly, it can be seen that the number of secure bits decreases and the number of vulnerable bits increases.
3. Analyzing the above results, the most suitable method for analyzing the `S-DES`cipher is the quantum neural network method (amplitude embedding) using rotation and entanglement. In this method, the number of secure bits is the smallest and the number of vulnerable bits is the highest in the same 30 epochs compared to other methods. In addition, a higher BAP could be obtained in fewer epochs compared to the classical neural network.

4.3 Results of Key Recovery Attack for `S-DES` using Quantum Neural Network

Table 11: Comparison of BAP, epoch, number of parameters (C: classical neural network, Q_{Amp}: Quantum neural network with amplitude embedding, Q_{IQP}: Quantum neural network with IQP embedding, Params: the number of parameters).

Method	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th	Avg.	Epoch	Params
C	60	66	69	57	56	81	53	60	86	80	66.8	100	55092
Q_{Amp}	64	71	71	56	51	80	56	60	85	80	67.4	30	44276
Q_{IQP}	58	65	60	55	54	79	51	57	84	75	63.8	30	38084

Table 11 shows the comparison of BAP, epoch, and number of parameters. The classical neural network method has the second highest average BAP (Avg.), but requires more epochs and parameters than quantum neural networks. The quantum neural network method (Q_{Amp}) using amplitude embedding has the highest average BAP and requires fewer epochs and fewer parameters than classical neural networks. That is, there is an advantage that the average BAP is higher than that of the classical method despite fewer epochs and fewer parameters required to detect the same number of vulnerable bits and secure bits. The quantum neural network method (Q_{IQP}) using IQP embedding has a small number of epochs and parameters, but this is due to the limitation of quantum resources. Also, since the average BAP is the lowest, the number of secure bits is the highest and the number of vulnerable bits is the smallest. This is an unsuitable approach for current quantum computers, as they are difficult to train sufficiently. Q_{Amp} also has a small epoch due to the limitation of quantum resources, but it is an advantage because the BAP is high. Therefore, it is the best method for cryptanalysis for S-DES. Based on the above experimental results, we can get the following conclusions.

1. In the cryptanalysis for S-DES, the quantum neural network achieved better performance than the classical neural network. Qubits can represent values on the Bloch sphere as well as 0 and 1. That is, since it has a wider expression range for data, it is possible to express the weights more elaborately, and thus, it is thought that better performance can be obtained. In other words, the quantum advantage obtained through quantum neural networks is attributed to the entanglement of qubits and their rich expressive range.
2. Comparing the two types of quantum neural networks, it is difficult to obtain good performance in the method of embedding one data feature in one qubit due to the limitations of quantum resources. In the future, if available qubits increase and acceleration of circuit execution become possible, it is thought that the characteristics of quantum neural networks can be sufficiently used.
3. Currently, in our implementation, the quantum neural network is configured in such a way that the 4 qubits quantum circuit is performed 4 times to process 64 weights, and then the outputs are all summed. That is, the weights of the previous layer are not processed at once, but the weights are partially divided and trained. However, in the case of cryptographic data, all bits are correlated by confusion and diffusion. Therefore, if it is possible to operate 16 qubits at once within a realistic time, it is expected that higher performance can be achieved because all weights can be entangled. In other words, if a quantum computer and quantum neural network technology that can use more resources is developed, a higher BAP will be achieved.

4.4 Results of Key Recovery Attack with Quantum Noise

All of the previous experiments were performed in a noise-free environment of the quantum computer, and Table 12 shows the results of analyzing S-DESusing a simulator with added noise of the quantum computer. In order to consider

Table 12: Comparison of BAP considering the noise of quantum computers.

Method	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th	Avg.	Epoch
C	59	59	65	53	54	80	54	56	80	72	63.2	15
Q_{Amp}	59	54	55	57	51	81	52	53	82	66	61.0	15

the noise of a quantum computer, a real quantum computer must be used. For this, an attempt was made to access the actual quantum hardware through the IBM Qiskit plug-in, but a backend error occurred due to the library update. Therefore, to consider the noise, we added a noisy quantum channel to the noise simulator and used it. We compared a classical neural network and a quantum neural network (Q_{Amp}) trained with a noise simulator. We trained for 15 epochs for both methods, and we can see that the BAP of the classical neural network is higher. The number of vulnerable bits is the same in both methods, but the number of safe bits is greater in the noise simulator-based quantum neural network. This means that an error occurred due to the occurrence of quantum noise and training was not performed properly. From this, we can draw the following conclusions.

1. For now, cryptanalysis using real quantum computers has difficulties due to quantum noise.
2. In the future, if the available qubits of quantum computers increase and error correction becomes possible, the loss of accuracy due to noise will decrease.

5 Conclusion

In this work, cryptanalysis based on quantum neural networks was performed by using Pennylane, a quantum hybrid neural network library, together with Tensorflow-Keras for classical neural networks. Analysis of `S-DES`using quantum neural networks was successful, yielding a slight quantum advantage (higher BAP with fewer epochs and fewer parameters), confirming its potential. At the same time, it was concluded that analysis of more complex ciphers would be difficult in practice. Therefore, it is expected that the cryptanalysis technique proposed in this work will be better utilized if stable quantum computers on large scale are developed.

In the future, we plan to implement hybrid quantum neural networks capable of more accurate cryptanalysis using more qubits and a variety of embedding and parameterized circuits.

References

1. F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell, *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019. 1

2. S. Gupta and R. Zia, "Quantum neural networks," *Journal of Computer and System Sciences*, vol. 63, no. 3, pp. 355–383, 2001. 2
3. E. Moreno-Pineda, C. Godfrin, F. Balestro, W. Wernsdorfer, and M. Ruben, "Molecular spin qudits for quantum algorithms," *Chemical Society Reviews*, vol. 47, no. 2, pp. 501–513, 2018. 4
4. P. Rebentrost, M. Mohseni, and S. Lloyd, "Quantum support vector machine for big data classification," *Physical review letters*, vol. 113, no. 13, p. 130503, 2014. 4
5. W. S. Noble, "What is a support vector machine?," *Nature biotechnology*, vol. 24, no. 12, pp. 1565–1567, 2006. 5
6. E. Farhi and H. Neven, "Classification with quantum neural networks on near term processors," *arXiv preprint arXiv:1802.06002*, 2018. 5
7. V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, M. S. Alam, S. Ahmed, J. M. Arrazola, C. Blank, A. Delgado, S. Jahangiri, *et al.*, "Pennylane: Automatic differentiation of hybrid quantum-classical computations," *arXiv preprint arXiv:1811.04968*, 2018. 5
8. H. Suryotrisongko and Y. Musashi, "Evaluating hybrid quantum-classical deep learning for cybersecurity botnet dga detection," *Procedia Computer Science*, vol. 197, pp. 223–229, 2022. 6
9. E. Houssein, Z. Abohashima, M. Elhoseny, and W. Mohamed, "Hybrid quantum convolutional neural networks model for covid-19 prediction using chest x-ray images. 2021." 6
10. E. Grant, M. Benedetti, S. Cao, A. Hallam, J. Lockhart, V. Stojevic, A. G. Green, and S. Severini, "Hierarchical quantum classifiers," *npj Quantum Information*, vol. 4, no. 1, pp. 1–8, 2018. 6
11. H.-L. Huang, Y. Du, M. Gong, Y. Zhao, Y. Wu, C. Wang, S. Li, F. Liang, J. Lin, Y. Xu, *et al.*, "Experimental quantum generative adversarial networks for image generation," *Physical Review Applied*, vol. 16, no. 2, p. 024051, 2021. 6
12. C.-H. H. Yang, J. Qi, S. Y.-C. Chen, P.-Y. Chen, S. M. Siniscalchi, X. Ma, and C.-H. Lee, "Decentralizing feature extraction with quantum convolutional neural network for automatic speech recognition," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6523–6527, IEEE, 2021. 6
13. J. Bausch, "Recurrent quantum neural networks," *Advances in neural information processing systems*, vol. 33, pp. 1368–1379, 2020. 6
14. J. So, "Deep learning-based cryptanalysis of lightweight block ciphers," *Security and Communication Networks*, vol. 2020, 2020. 12, 13
15. H. Kim, S. Lim, Y. Kang, W. Kim, and H. Seo, "Deep learning based cryptanalysis of lightweight block ciphers, revisited," *Cryptology ePrint Archive*, 2022. 12, 13, 19
16. D. Jankovikj, H. Mihajloska Trpceska, and V. Dimitrova, "Cryptanalysis of round-reduced ascon powered by ml," 2022. 12, 13
17. 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. 13

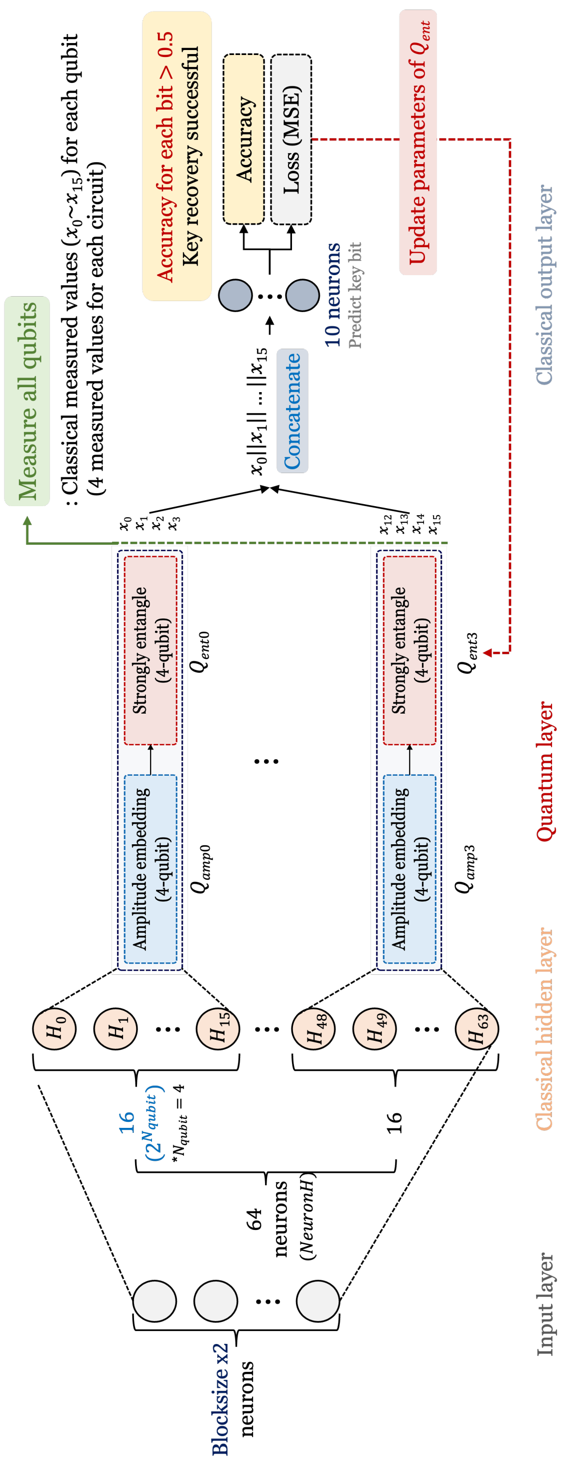


Fig. 7: Detailed diagram of a quantum-classical hybrid neural network.