Quantum Machine Learning with Phase Manipulation

－Toward the establishment of a new method for general-purpose quantum image processing－

Keiichiro Hayashi
Nada High School 8-5-1 Uozakikita-machi, Higashinada-ku, Kobe-city, Hyogo, 6580052 Japan
Email: keiichiroharry884@gmail.com

Abstract Inspired by classical convolutional neural networks, I designed a quantum circuit that performs image input and convolution using MCRz-gates to perform quantum image recognition machine learning. The following seven points are unique to this research.
(1) The point of dropping data as a phase into the coefficients that depend on the measurement probability of the superimposed quantum state.(2) The development of an algorithm that automatically performs input for arbitrary image sizes.(3) The fact that I overcame a barrier unique to my method at the time of convolution by arranging the padding method.(4) The number of classical calculations is minimized by using the simultaneous perturbation method.(5) The algorithm is a hybrid type algorithm in which the parameters are updated by classical computation.(6) Nonlinearity is ensured by inserting multiple observations.(7) The method can cope with an increase in the number of qubits and convolution layers. I designed a model to identify whether an image is 0 or 1 on the MNIST image set, and performed machine learning using Qiskit. The accuracy aspects were verified, leading to the completion of the general framework of the theory of a new quantum image processing method. The model is expected to improve in accuracy as it is refined, and I show its potential as a method for processing multidimensional information not only for quantum data but also for classical data in a general-purpose manner.
Keywords Quantum Computer, Image Recognition, Machine Learning, Neural Networks

1 Introduction

The motivation for this research can be explained by the author's interest in medical image processing. Suffering from a brain disease, the author dreamed of a system to analyze the human body in an integrated manner and make a comprehensive diagnosis as a future diagnostic technology. Although CT and MRI can be used to obtain image information of the inside of the body, the current situation is that from there it is left to the physician's judgment. The latest trend in clinical medicine is to add informatics-based diagnosis to the artificial judgment of physicians, as suggested in the lecture by Professor Motoyasu Honma of Tohoku University Graduate School of Biomedical Engineering at the 194th Annual Conference on Medical Imaging and Informatics. When he conceived of a system that could provide a comprehensive diagnosis, surpassing the accuracy of a physician's diagnosis, based on image information such as CT and MRI data or information before it is processed into images, he discovered the potential of algorithms using quantum computation. In order to realize an automatic diagnosis based on information processing that uses an excessive amount of information and processing processes that cannot be handled in batches at present, I believe that it is essential to establish a new information processing method that is supported by quantum transcendence and is outside the framework of classical computation.

In this research, based on the idea of quantum machine learning, especially quantum neurons, proposed in the following paper as one of the application algorithms of quantum computers that have been attracting attention in recent years, I will start with an approach to the unexplored issue of materializing the input-output part for these quantum neurons, and then develop a learning model that can demonstrate the advantages of quantum algorithms. I will work on the realization of a learning model that can demonstrate the advantages of quantum algorithms, and on the optimization of weights by hybrid machine learning.

The principle of quantum machine learning has been proposed in previous research, but when I try to apply it to image recognition as a practical example, I need to consider the "specifics" such as how to use the input image as input to the quantum neuron and how to judge that "the image was recognized" from the result of the quantum process. The "issues that will emerge when trying to apply this technology to specific problems" have not been sufficiently examined so far. By confronting such practical issues, it is thought that new similarities and differences with conventional methods can be made apparent.

I also attempted a more practical image recognition machine learning by designing a structure with multiple weight layers, while taking care of the shortcomings caused by linear methods in the process of image discrimination. In this study, I ignore the influence of noise and tackle this problem using Qiskit, an open source for quantum programming, to build a foundation for practical application. The general-purpose development environment provided by NICT was used in the experiments.

2 Materials and Method/Theory

The research was conducted using Qiskit, an open source framework for quantum circuit generation and quantum computation, coded in Python on a Jupyter notebook. The features of this research described in the "Abstract" and the accompanying methodologies are described for each research method, and additions are made as appropriate.

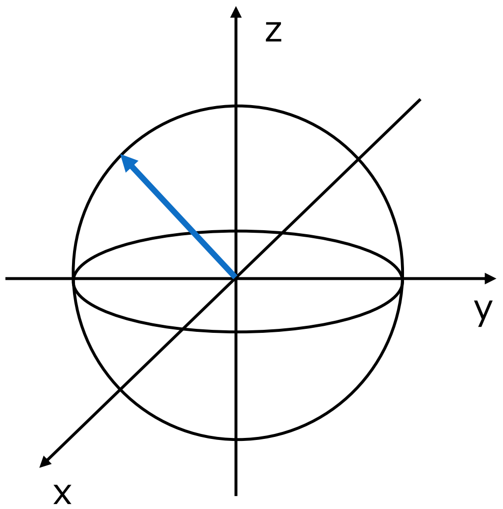
(1) The point where the data is dropped as a phase into the coefficients that depend on the measurement probability of the superimposed quantum state.

The superposition of qubits is represented by an Adamar gate as follows.

$$H|0> = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad (1.1)$$

When this is observed in the Z basis, the probability of |0> and |1> being observed is 1/2, respectively. In general, given $(\alpha, \beta)^T$ in standardized α and β, the probability of observing |0> is $|\alpha|^2$ and the probability of observing |1> is $|\beta|^2$. Inspired by this, equation (1.1) can be expressed as

$$H|x_k> = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ exp\left(\frac{2\pi i}{2}x_k\right) \end{pmatrix} \quad (1.2)$$



(Fig.1 Bloch sphere)

when generalized in view of the case with initial value |1>. Introducing the $RZ(\theta)$ gate represented by

$$Rz(\theta) = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix} \qquad (1.3)$$

Therefore, the state of equation (1.2) is represented by

$$Rz(\theta)\,|\psi> = \frac{1}{\sqrt{2}} \begin{pmatrix} exp\left(-i\frac{\theta}{2}\right) \\ exp\left(i\frac{2\pi x_k+\theta}{2}\right) \end{pmatrix} \qquad (1.4)$$

where $|\psi>$ is the state of equation (1.2). The input and processing of image information can be done by $C^{\otimes n}Rz$ (where $C^{\otimes n}$ is the number of n control bits) by manipulating the value of this θ. In detail See (2) and (3) for details.

Even if we observe equation (1.4) as it is, the observed probabilities of |0> and |1> are still 1/2, respectively. Therefore, I introduce the quantum Fourier transform (QFT). The quantum Fourier transform is a quantum algorithm of the discrete Fourier transform for the amplitude of a wave function. In the discrete Fourier transform, the mapping of vector $\left(x_0, x_1, \ldots, x_{N-1}\right)$ to vector $\left(y_0, y_1, \ldots, y_{N-1}\right)$ can be defined as

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j w_N^{jk} \qquad (1.5)$$

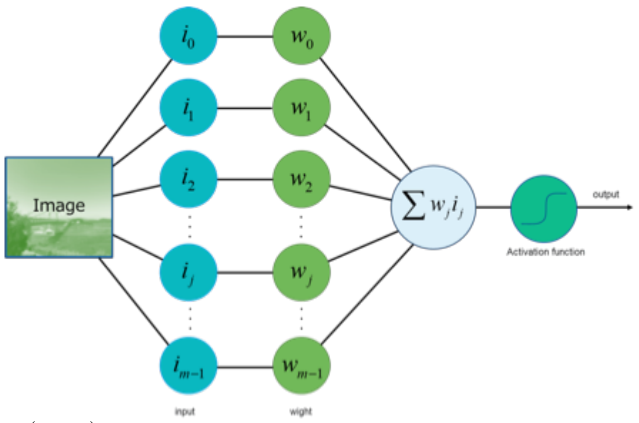
,where $w_N^{jk} = e^{2\pi i\frac{jk}{N}}$. On the other hand, the quantum Fourier transform maps quantum state $\sum_{i=0}^{N-1} x_i|i>$ to quantum state $\sum_{i=0}^{N-1} y_i|i>$ according to the same correspondence between x and y as in equation (1.5).

$$|x> \mapsto \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} w_N^{xy}|y> \qquad (1.6)$$

As a unitary matrix, it is expressed as

$$U_{QFT} = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} w_N^{xy}|y><x| \qquad (1.7)$$

The quantum Fourier transform is an operation that transforms two bases, the computational basis (Z basis) and the Fourier basis (which depends on the exponential part of e. It transforms the states |0> and |1> of the Z basis into the states |+> and |-> of the x basis. Similarly, every qubit in the computational basis has a state in the



(Fig.2) Schematic of a classical neural network corresponding Fourier basis. In the Bloch sphere of Fig.1, the observation probability is a quantity that depends on the square of the coordinates of a point when mapped to the z-axis. In this sense, it can be understood that the observation probability of |0> and |1> is half and half in Fig.1, even if the input and convolution operations are performed to move on the xy-plane by rotating the sphere around the z-axis. In contrast, the quantum Fourier transform gives unique observables by switching the basis to escape from the xy-plane. It is an intuitive example that the H-gate in Eq. (1.2) is a one-qubit quantum Fourier transform. In this research, the process of inputting discrete classical data and processing the image is performed with a phase specification, and the phase is extracted by quantum Fourier transform upon observation. This is shown below.

(2) <u>The development of an algorithm that automatically performs input for any image size.</u>

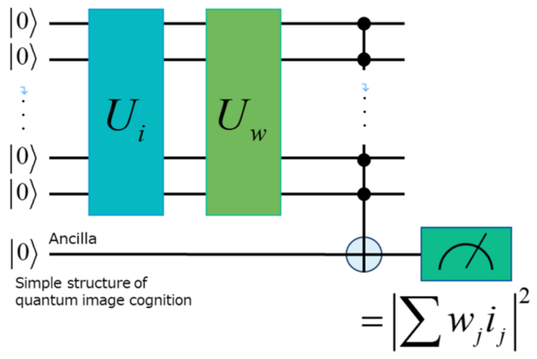
In a neural network in classical machine learning, the process of mapping input information to nodes, multiplying weights, and firing when they exceed a threshold value is performed many times to classify the original input information or predict values from the final output (Fig.2). The inspiration for how the input information is mapped to the quantum states in this study and how the weights are multiplied together came from previous research [1]. Let $\vec{i}$ be the vector to which the input information is mapped and $\vec{w}$ be the vector of weights to multiply it by.

$$\vec{i} = \begin{pmatrix} i_0 \\ i_1 \\ \vdots \\ i_{m-1} \end{pmatrix}, \vec{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{m-1} \end{pmatrix} \qquad (2.1)$$

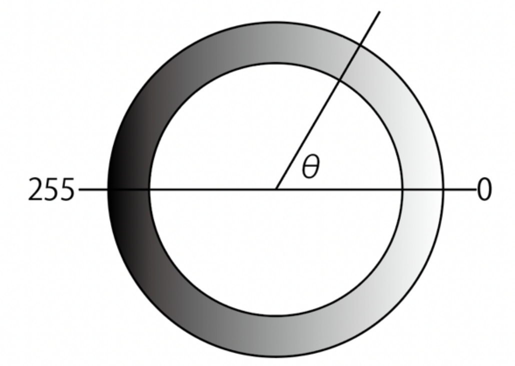
, where $i_j, w_j \in \{e^{i\theta} | \theta \in \mathbb{R}\}$ and the quantum state $|\psi_i>, |\psi_w>$ is represented by

$$|\psi_i> = \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} i_j |j>, |\psi_w> = \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} w_j |j> \qquad (2.2)$$

$|j> \in \{|00\ldots00>, |00\ldots01>, \ldots, |11\ldots11>\}$ at this time, and this 0 and The sequence of numbers with only 1 is interpreted to be the binary notation of j, a decimal number. Quantum bits are superpositions of states in which The qubits are superpositions of states in which |0> and |1> are probabilistically observed, respectively. and $|\psi_i>, |\psi_w>$ represents the state of these multiple qubits. Based on this idea, let me consider the structure of a simple quantum circuit (Fig.3). Here, U_i is a unitary gate that creates the quantum states of $|0>^{\otimes n}$ to $|\psi_i>$, the initial states. U_w is a unitary gate that multiplies the weights represented by $|\psi_w>$. Finally, the firing is left to the inversion of the Ancilla qubit by the $C^{\otimes N}X$ gate.


|0⟩
|0⟩
|0⟩
|0⟩
|0⟩
U_i
U_w
Ancilla
Simple structure of quantum image cognition
$= \left|\sum w_j i_j\right|^2$

(Fig.3 Schematic of a quantum neural network circuit)



(Fig.4 Correspondence between phase and grayscale)

$$U_i|0>^{\otimes N} = |\psi_i>,$$
$$U_w|\psi_w> = |1>^{\otimes N} = |m-1> \quad (2.3)$$

, where $m = 2^N$. When U_w applies the weight $|\psi_w>$, if the quantum state immediately before U_w is applied is $|\psi_w>$, then all N qubits after U_w is applied will be |1>. The $C^{\otimes N}X$ gate inverts the ancilla qubit and the measured value is 1. When U_w is applied to $|\psi_i>$ corresponding to the input information,

$$|\phi_{i,w}> \equiv U_w|\psi_i> = \sum_{j=0}^{m-1} c_j|j> \quad (2.4)$$

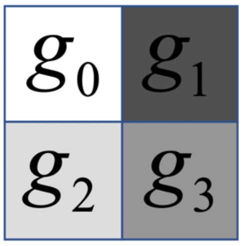
Using the definitions in equations (2.3) and (2.4), the inner product of the two quantum states is

$$<\psi_w|\psi_i> = <\psi_w|^t U_w{}^* U_i|\psi_i> \quad (2.5)$$
$$= <m-1|\phi_{i,w}> = c_{m-1}$$

$\left|c_j\right|^2$ represents the probability that the quantum state of the N qubits is |j> just before the $C^{\otimes N}X$ gate is applied. Therefore, $\left|c_{m-1}\right|^2$ indicates the probability that the quantum state is |11...11>, i.e., the probability that the ancilla qubit is inverted. The first step is to normalize the grayscale of each pixel of the input image. Here, in order to correspond the phase to the grayscale as shown in Fig.4, the phase is represented by $\theta = g\pi/256$ for the grayscale $g(0 \leq g < 256)$. Therefore, it is represented as $i_j \in \{e^{i\theta}|\theta \in \mathbb{R}\}$. Given an image with n × n pixels, the number of qubits to store this image is $\left[2\log_2 n\right] + 1$ (where [] is a Gaussian symbol). The normalized grayscale is stored in this n x n θ, and the image is input based on it. For padding, see (3). As an example, let me consider a 2 × 2 image as shown in Fig.5. This image can contain information in 2 qubits, and the method is described below. When the $C_{\vec{l}}{}^{\otimes n}Rz(\theta)$ gate is introduced, it is represented as

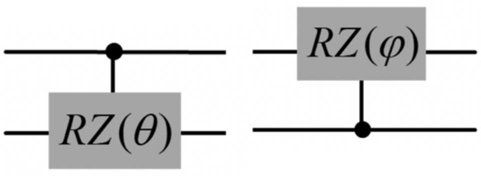
$$C_{\vec{l}}{}^{\otimes n}Rz_k(\theta)\,|j> = \begin{cases} e^{i\left(\frac{\pi}{2}-\frac{\theta}{2}\right)}|j> \ if \bigcap j_i \geq l_i \ \ and\ j_k = 0 \\ e^{i\left(\frac{\pi}{2}+\frac{\theta}{2}\right)}|j> \ if \bigcap j_i \geq l_i \ \ and\ j_k = 1 \\ |j> \quad otherwise \end{cases} \quad (2.6)$$

for the quantum state called |j>. Here, the m-th element of $\vec{l}$ stores 1 if the m-th qubit is a control bit and 0 otherwise, and k indicates that the kth qubit is the target bit. $Rz(\theta)$ is applied to the target bit when the control bits are all 1s, but the sign in the middle changes



$$0 \leq g_j < 256$$

(Fig.5 Example of a black-and-white image of 2 x 2 squares)



(Fig.6 CRZ gate)

depending on whether the kth qubit is 0 or 1 (see equation (1.3)). As long as this qubit is on the xy-plane, the probability that this kth qubit is |0>,|1> is 1/2 each. Here, using the two gates as shown in Fig. 6 for the two-qubit after applying an H-gate to each, we obtain

$$|\psi> = \frac{1}{2}\left(\begin{array}{l} |00> + e^{i\left(\frac{\pi}{2} - \frac{\varphi}{2}\right)}|01> \\ + e^{i\left(\frac{\pi}{2} - \frac{\theta}{2}\right)}|10> + e^{i\left(\pi + \frac{\varphi}{2} + \frac{\theta}{2}\right)}|11> \end{array} \right) \tag{2.7}$$

The parameters of the Rz gate are determined by designing a linear system of equations from the matrix and solving them so that the coefficient part of e is in phase $\vec{i}$ corresponding to the input. In this case, the example in equation (2.7) cannot solve the simultaneous equation because there are two parameters for three input values, but according to

$$\sum_{k=2}^{n} (nCk) \times k > 2^n \ \ (n > 2) \tag{2.8}$$

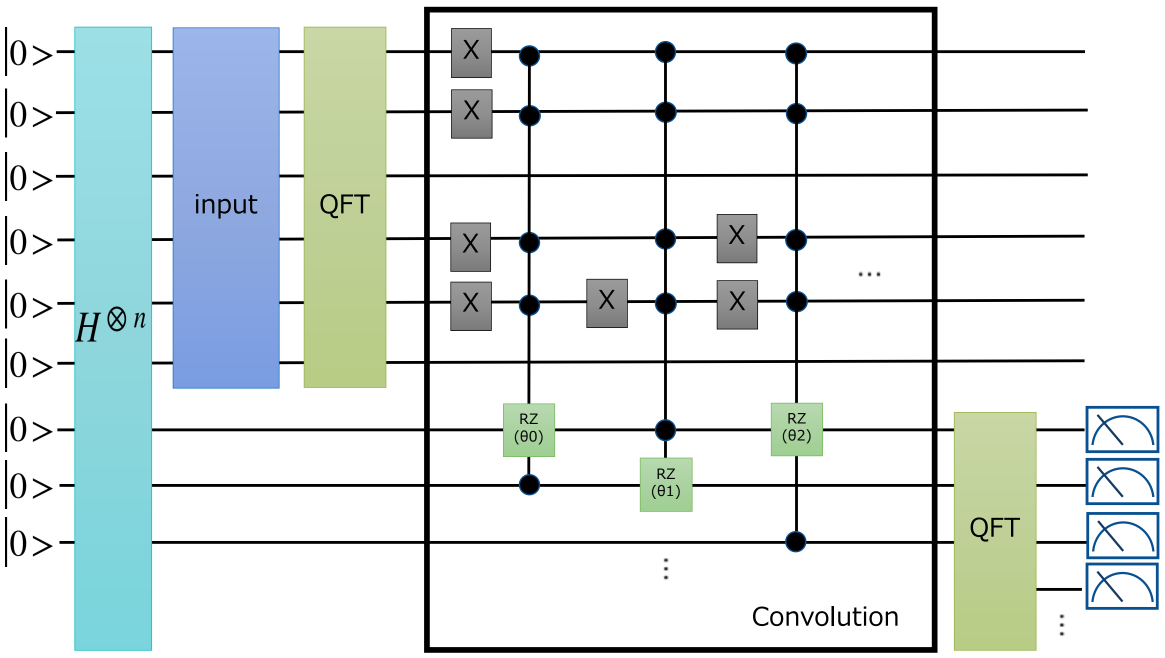
, the method can be used because the type of Rz gate is larger than the number of input values for three qubits or more. According to this, I have completed an algorithm to automatically design quantum circuits for the contents of U_i and U_w. The specific code is omitted, but for the two that were created separately, I was able to confirm that the circuits were assembled correctly, as they output 1 when U_w is applied to $|\psi_i>$ corresponding to the input information (2.5).

(3) <u>Barriers unique to my method at the time of convolution are overcome by arranging the padding method.</u>

The MNIST image was preprocessed into a 32 x 32 image by adding pixels around the image (padding) since the MNIST image is 28 x 28 in size. This reduces the number of qubits needed for the input to 10. Convolution is often explained as the repeated process of applying a filter to an image, multiplying the values of overlapping areas, and adding them together. In this research, the coefficients of the superposition are assigned to the image as shown in Fig.7 (for 6 qubits), and the proposed quantum circuit is in the form shown in Fig.8. Regarding Fig.7, for example, in the top left four squares, the 0th, 1st, 3rd, and 4th qubits from the top (starting counting from 0) are all 0 and common. If we apply the X-gate and the $C_{(110110)}{}^{\otimes 4}Rz(\theta)$ gate to these four qubits, the Rz-gate will act on the newly introduced target bit only when the

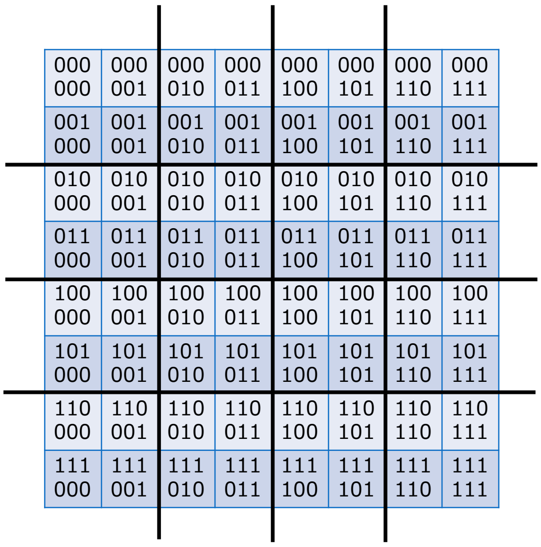
000 000	000 001	000 010	000 011	000 100	000 101	000 110	000 111
001 000	001 001	001 010	001 011	001 100	001 101	001 110	001 111
010 000	010 001	010 010	010 011	010 100	010 101	010 110	010 111
011 000	011 001	011 010	011 011	011 100	011 101	011 110	011 111
100 000	100 001	100 010	100 011	100 100	100 101	100 110	100 111
101 000	101 001	101 010	101 011	101 100	101 101	101 110	101 111
110 000	110 001	110 010	110 011	110 100	110 101	110 110	110 111
111 000	111 001	111 010	111 011	111 100	111 101	111 110	111 111

(Fig.7 Correspondence between quantum information and the image information to be accommodated in its coefficients)



(Fig.8 Flow of Input, Convolution, Pooling, and Observation)

0,1,3,4,th qubits are initially 0. In this case, some of the newly introduced qubits are also control bits. By using a different $C^{\otimes n}Rz(\theta)$ each time when ignoring the 10 qubits used for input, the information is mapped to a newly introduced group of qubits as independent information for each convolution process. The additional control and target bits in this case do not need to be mapped in a way that preserves the positional relationship of the input image. Even if a mapping is done in a way that preserves positional relationships if the phases of the coefficients can be controlled in the order of |00…00>,|00…01>,…,|11…11> from the upper left, there is no need to write code to design and solve simultaneous equations when image information is input (actually, this was the most difficult part). It is no exaggeration to say that the inability to manipulate one quantum state individually was the most difficult part of this research. The positional information is retained to a certain extent by the correlation of the control bits of the convolution, and there is little need to be concerned with positional information anymore, rather than to find a relationship between distant information in Hilbert space. In any case, in general, classical neural networks and in this method, the information of the location relation is lost because the all-joining layer finally joins all nodes together in a full exploratory manner.
When implementing convolution here, I faced a barrier unique to the quantum algorithm considered in this study. The filter cannot be applied to straddle the lattice as shown in Fig.9. According to this example, half of the quantum states in the lattice are matched, but the number of matches changes when crossing the lattice. If we forcefully set the control bits using the same technique, we can confirm that we can



000 000	000 001	000 010	000 011	000 100	000 101	000 110	000 111
001 000	001 001	001 010	001 011	001 100	001 101	001 110	001 111
010 000	010 001	010 010	010 011	010 100	010 101	010 110	010 111
011 000	011 001	011 010	011 011	011 100	011 101	011 110	011 111
100 000	100 001	100 010	100 011	100 100	100 101	100 110	100 111
101 000	101 001	101 010	101 011	101 100	101 101	101 110	101 111
110 000	110 001	110 010	110 011	110 100	110 101	110 110	110 111
111 000	111 001	111 010	111 011	111 100	111 101	111 110	111 111

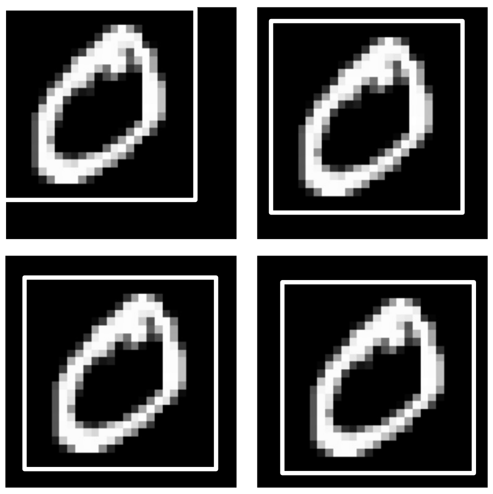
(Fig.9 Convolutional filter cannot cross the grating)

find other combinations of four squares that hit the same conditions. This does not allow us to obtain information on the filtered areas independently. It was also thought that the relationship between pixels separated by a grid must be completely ignored, but as a small technique to overcome this problem, I focused on padding, which is done as a process before the image is input. It is usual to process a 28 x 28 MNIST image into a 32 x 32 image by adding two black cells around the perimeter of the original 28 x 28 MNIST image, but several different images can be created by varying the cells to be added to the top, bottom, left, and right (Fig.10). If the image is a square with m sides, there are $\left(2^{[\log_2 m]+1} - m\right)^2$ numbers of degrees of freedom, and by preparing as many of these as the number of filter cells, it is possible to perform convolution between pixels that straddle the grid shown in Fig.9. Here, the control and target bits of the $C^{\otimes n}Rz(\theta)$ gate are chosen so that the information of the same m-th image in each processing pattern is close in distance in the Hilbert space.

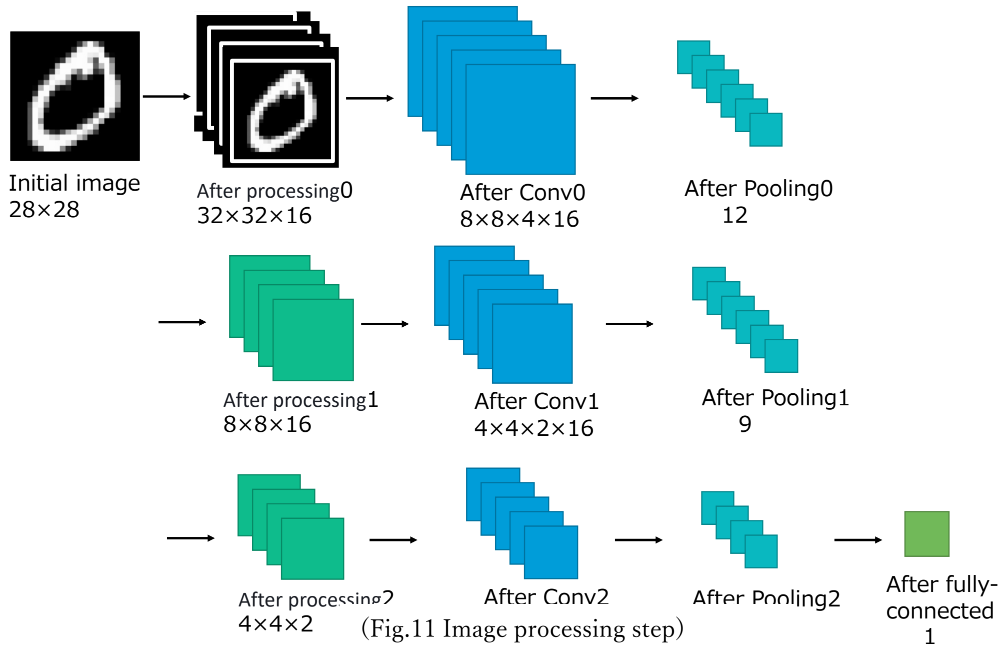
Based on these considerations, the image processing steps are shown in Fig.11, where the Pooling layer refers to the last quantum Fourier transform operation in Fig.8, which transforms the phase basis information into the X basis to obtain classical observed values while reducing the data size. To the existing method of repeating the Convolution and Pooling layers, I added a step where I put in an observation, a nonlinear operation, and input it back into the quantum state. See also (6) for more on this, where image data consisting only of 0s and 1s are learned, and the probability that an image is a 1 is finally output via all the Convolution layers.

(4) The number of classical calculations is reduced as much as possible by using the simultaneous perturbation method.

In the image processing step of this study shown in Fig.11, the number of parameters to be updated by the gradient descent method of machine learning is 1292. Let be defined as the



(Fig.10 Processing the same image with different paddings)



(Fig.11 Image processing step)

$$E(i,\vec{k}) = \frac{1}{\kappa}\sum_{m=0}^{\kappa-1} E_m{}'(i,\vec{k}) \qquad (4.1)$$

expected value $E(i,\vec{k})$ of the output result obtained from some training data i and a set of parameters $\vec{k}$. where κ is the number of shots, and in this study, κ = 1024. Also, for each m 0 and 1 training data, the loss function L is designed as

$$L(\vec{k}) = \frac{1}{2m}\sum_{j=0}^{m-1}\left[\left\{E\left(i_{zero,j},\vec{k}\right)\right\}^2 + \left\{1-E\left(i_{one,j},\vec{k}\right)\right\} \qquad (4.2)$$

In this study, the number of each training data was set at 100. Each parameter is updated in the direction of minimizing the value of this loss function; let me denote the parameter group $\vec{k}$ of the t-th training as $\vec{k}_t$,

$$\overrightarrow{k_{t+1}} = \vec{k}_t - \eta\overrightarrow{\Delta k_t} \qquad (4.3)$$

$$\Delta k_{t,j} = \frac{L\left(\vec{k}_t + c\vec{s}_t\right) - L\left(\vec{k}_t - c\vec{s}_t\right)}{2cs_{t,j}} \qquad (4.4)$$

$$= \frac{L\left(\vec{k}_t + c\vec{s}_t\right) - L\left(\vec{k}_t - c\vec{s}_t\right)}{2c} s_{t,j}$$

Here, the simultaneous perturbation method is used so that the loss function is calculated only twice when the parameters are updated once. $s_{t,j}$ is 1 or -1, and each element is determined independently and randomly based on a Bernoulli distribution. c(>0) represents the magnitude of the perturbation common to all elements. Foolishly updating the parameters would result in the calculation of

$$L'\left(\vec{k}_t\right) = \left(\frac{\partial L\left(k_{t,0}\right)}{\partial k_{t,0}}, \frac{\partial L\left(k_{t,1}\right)}{\partial k_{t,1}}, \ldots, \frac{\partial L\left(k_{t,1451}\right)}{\partial k_{t,1451}}\right)^T \qquad (4.5)$$

Using the simultaneous perturbation method greatly reduces the number of times the loss function is calculated.

(5) <u>It is a hybrid type of algorithm in which the parameters are updated by classical computation.</u>

As shown in (4), the parameters in the quantum circuit are updated to minimize the value of the loss function by the gradient descent method. The process is based on classical computation with reduced computational complexity and can be said to be a hybrid type of classical and quantum algorithm.

(6) Non-linearity is ensured by sandwiching multiple observations.

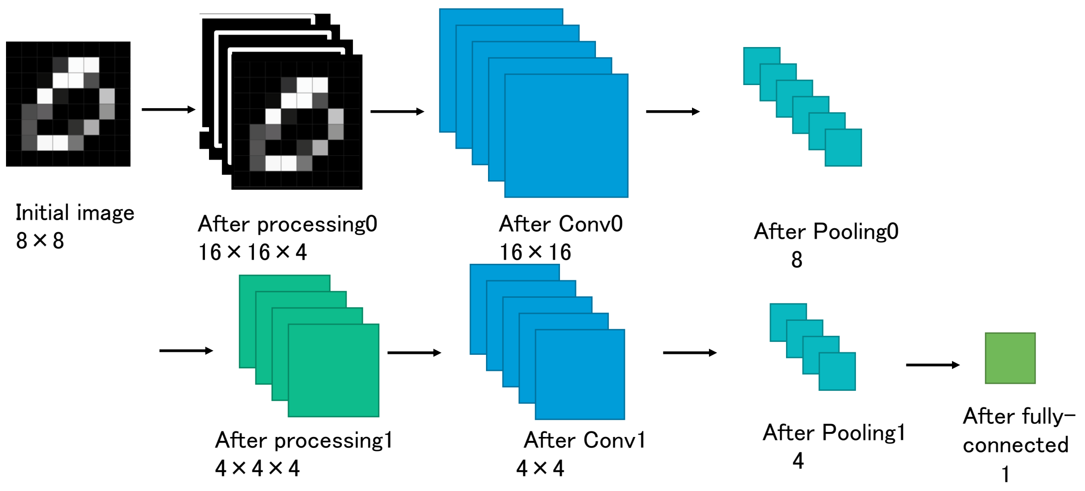
In the learning step in Fig.11, linearity is avoided by performing an observation after the Convolution and Pooling layers and re-inputting their values. The reason for this is that all quantum gates are linear operations because they apply unitary transformations. Therefore, even if we use many quantum gates to deepen the layers, they can be regarded as a single unitary gate, and the learning accuracy may reach a ceiling. As a simple example, a Rubik's cube can be mixed a lot from a state where all sides are aligned, but no matter how much it is mixed, it is the same as making that state in 10 moves from a state where all sides are aligned.

(7) The method can handle an increase in the number of qubits and convolution layers used.

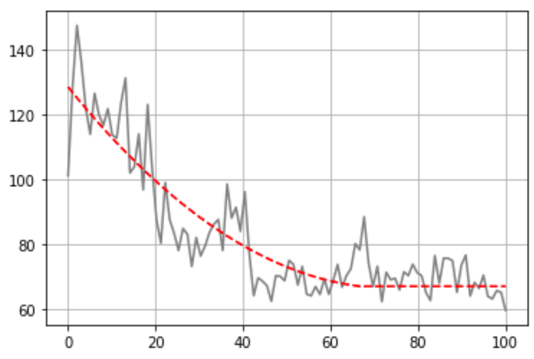
In considering my research methodology, I tried to make sure that it is a general-purpose quantum machine learning method that can be applied not only to MNIST, but also to images of arbitrary size, or data other than images.

3 Result

I implemented it in Python based on the method shown in the research method above, but it became a long code that handled 26 qubits, and the kernel crashed due to insufficient memory even using a 32-core NICT general development environment as well as the computer at hand. Therefore, I tried again with the learning step shown in Figure 12, compressing the 28 x 28 MNIST image to 8 x 8, padding to create 4 types of 16 x 16 images per original image, mapping to 10 qubits, using 18 qubits in the Convolution layer stage, and finally 8 qubits in the observations and output. These are



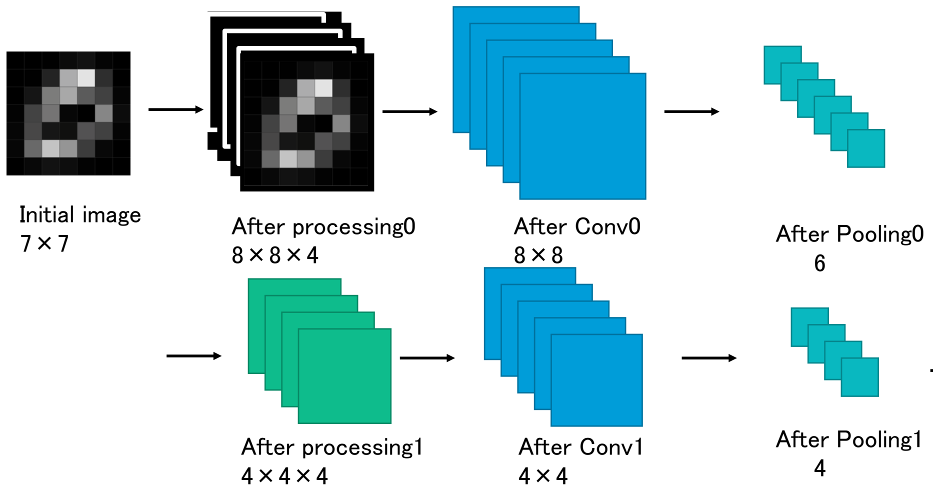
(Fig.12 Reduced data size and number of layers)



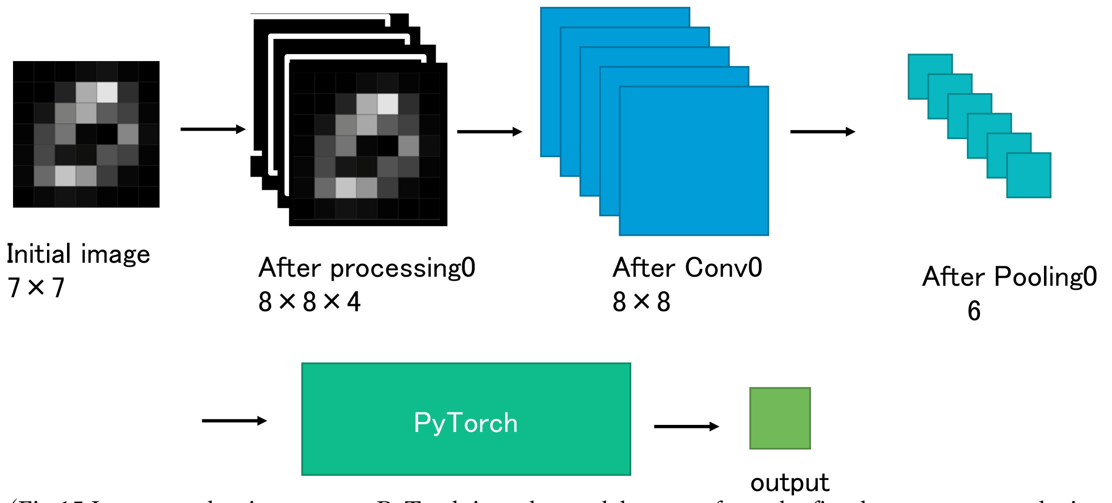
(Fig.13 The model in Figure 12 reduced the loss function but did not improve accuracy.)

then formatted as 4x4 images, and further formatted into 8x8 images with padding. This flow was processed in parallel on 32 cores and updated 100 times at a rate of 2 hours per training. The loss function is shown in Figure 13. The numbers converged smoothly, but the learning accuracy was 53%. The reason for this was thought to be that 3/4 of the 16 x 16 data were "blanks" that were covered with padding. Therefore, I further tried the learning step shown in Figure 14: compressing a 28 x 28 MNIST image to 7 x 7, padding to create four 8 x 8 images per original image, mapping to 8 qubits, using 14 qubits in the Convolution layer, and finally producing six outputs. Even so, the training accuracy was 55%." It seems that I was right to focus on the problem of many "blank" areas, but the shaping of the data into 4x4 data before entering the second round may not have been appropriate. Therefore, I attempted the learning step shown in Figure 15, in which the neural network for the 2-classification problem gradually makes the features more salient at each layer, and if the operation by the quantum algorithm in the first round is correct, then the classification should be possible when it is handed over to the classical algorithm afterward. Here, since the classical algorithm can calculate the probability of being 0 or 1 in each test case, I aggregated the square of the error with the correct answer and used it as the loss function. Here, PyTorch was employed as the classical algorithm.

As a result, what was initially 50% accuracy was improved to 73%.



(Fig.14 I further reduced the number of data, lightened the weight, and cut down on padding shortcomings, but did not improve accuracy.)

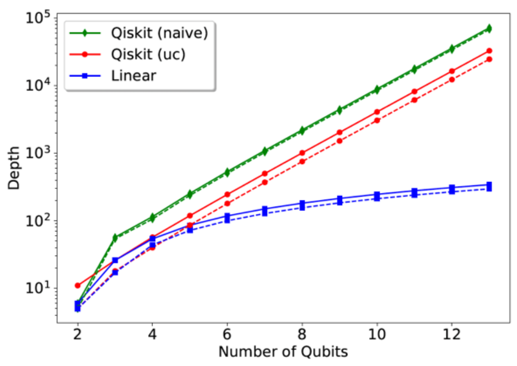


(Fig.15 I attempted to incorporate PyTorch into the model to transform the first layer output results into classifiable data.)

4 Discussion

Before discussing our method and its results, it is necessary to introduce the concept of "depth" of a quantum gate. Each quantum gate that performs a unitary transformation can be decomposed into several two-qubit combinations. I define "depth" as the number of 2 qubits required for decomposition. The depth of each quantum gate differs depending on the background decomposition method used by each of the several open sources. In Qiskit, as shown in Figure 16, if the number of qubits involved in qubits is n, the depth of the qubits is of the order of $O(k^n)$ $(2<n<3)$. On the other hand, the Linear method shown in Figure 16 allows decomposition on the order of $O(n^2)$.

Now, one measure in the evaluation of many quantum algorithms, including our method, is the computational cost. Since the computational cost can be thought of as the sum of the depth of the quantum gates used, it is necessary to try to reduce the depth and the number of MCRZ gates used, as discussed above. In our method, one MCRZ gate is used per filter superposition in the classical method. When the amount of original data is 2^n and the size of the filter is 2^m, this operation is performed 2^{n-m} times. Under the same conditions, the classical method can be approximated as 2^n times when m is sufficiently small relative to n. Also under these conditions, the depth of a single quantum gate is 2^n, the same as the



(Fig.16 Depth employed in Qiskit and when using the Linear method (Da silva, A.j., &Park, D.K. (2022). Linear-Depth Quantum Circuits for Multi-Qubit Controlled Gates. cited from Phys. Rev. A 106, 042602[10])) number of multiplications in the classical method. Therefore, the product of these two, the computational cost, can be considered as $2^{-m} \times$ T times the cost of the minimum gate operation compared to the classical method, if the cost of the minimum gate operation is T times the cost of multiplication in the classical calculation.

The number of fine qubits used simultaneously to solve MNIST is 26 in the case of the original image in Figure 11. A quantum circuit of this size is by no means impossible to construct, but our method is too heavy to be implemented in a classical computer simulation. After much trial and error, I arrived at the model shown in Figure 15 and were able to reduce the number of qubits to 14. 28 x 28 MNIST images were compressed to 7 x 7 and subjected to machine learning, and the fact that the classical machine learning method was able to classify with an accuracy of 88%, while the quantum computation method was able to classify with 73% is a great achievement.

If the advantage of processing classical data by quantum machine learning is that it can process a large amount of input data, then in the model shown in Figure 15, the output of quantum machine learning in the first layer decreases exponentially with the size of the input data, so there is no need to use a quantum algorithm in the second layer. This may be the case. Although I considered a model using a quantum algorithm for only the first layer to verify the usefulness of the method in this study, this model may coincidentally be the one that can make the best use of our method.

5 Conclusion

In this study, machine learning using a one-layer quantum algorithm was able to increase the accuracy from 50% to 73% by updating only the parameters used in the quantum circuit. This method is effective when the amount of data increases or when dealing with multidimensional data and is expected to clearly reduce the amount of computation compared to classical methods. The evaluation of the accuracy when the number of qubits is increased is a future issue.

6 Acknowledgments

I would like to express my deepest gratitude to Kobe University's ROOT Program, especially Professor Satofumi Soma, and NICT SecHack365, especially to the trainees and assistants of the Research Driven Course, for their great support and advice for this research.

7 References/Bibliography

[1] F. Tacchino, C. Macchiavello, D. Gerace, D. Bajoni, “An artificial neuron implemented on an actual quantum processor,” npj Quantum Information 5, 1-8 (2019).
[2] S. Mangini, F. Tacchino, D. Gerace, C. Macchiavello, D. Bajoni, “Quantum

computing model of an artificial neuron with continuously valued input data,"Mach. Learn.: Sci. Technol. 1 045008 (2020).
[3] F. Tacchino, C. Macchiavello, D. Gerace, D. Bajoni, "Variational learning for quantum artificial neural networks"
[4] F. Tacchino, P. Kl. Barkoutsos, C. Macchiavello, D. Gerace, I. Tavernelli, D. Bajoni, 2020 IEEE International Conference on Quantum Computing and Engineering (QCE), 130-136 (2020).
[5] F Tacchino, P. Barkoutsos, C. Macchiavello, I. Tavernelli, D. Gerace, "Quantum implementation of an artificial feed-forward neural network," Quantum Sci. Technol. 5 044010 (2020).
[6] S. Mangini, F. Tacchino, D. Gerace, D. Bajoni, C. Macchiavello, "Quantum computing models for artificial neural networks" EPL (Europhysics Letters) 134, 10002 (2021).
[7] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, "Parameterized quantum circuits as machine learning models," Quantum Sci. Technol. 4 043001 (2019).
[8] A. Chalumuri, R. Kune, B. S. Manoj, "Training an Artificial Neural Network Using Qubits as Artificial Neurons: A Quantum Computing Approach," Procedia Computer Science, 171, 568-575 (2020).
[9] Y. Li, R-G. Zhou, R. Xu, J. Luo, and W. Hu, "A quantum deep convolutional neural network for image recognition" Quantum Sci. Technol. 5 044003 (2020).
[10] Da silva, A.j., &Park, D.K. (2022). Linear-Depth Quantum Circuits for Multi-Qubit Controlled Gates. Phys. Rev. A 106, 042602

8 Appendix

The expected value of the derivative using the simultaneous perturbation method is the value of the partial derivative in each variable.

$$f(\vec{x}+c\vec{s}) = f(\vec{x}) + c\sum_{i=1}^{n} \frac{\partial f(\vec{x})}{\partial x_i} s_i + \frac{c^2}{2}\sum_{i,j=1}^{n} \frac{\partial^2 f(\vec{x})}{\partial x_i \partial x_j} s_i s_j + \cdots,$$

$$\bar{f} = f(\vec{x}),\ \nabla = \begin{pmatrix} \frac{\partial}{\partial x_1} \\ \vdots \\ \frac{\partial}{\partial x_n} \end{pmatrix},\ H = \nabla^2 \bar{f} = \begin{pmatrix} \frac{\partial^2 \bar{f}}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 \bar{f}}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 \bar{f}}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 \bar{f}}{\partial x_n \partial x_n} \end{pmatrix},$$

$$f(\vec{x}+c\vec{s}) \simeq \bar{f} + c(\nabla f, \vec{s}) + \frac{c^2}{2}(\vec{s}, H\vec{s}),$$

$$\frac{f(\vec{x}+c\vec{s}) - f(\vec{x})}{cs_i} \simeq (\nabla \bar{f}, \vec{s})\, s_i + \frac{c}{2}(\vec{s}, H\vec{s})\, s_i,$$

$$E\left(\frac{f(\vec{x}+c\vec{s}) - f(\vec{x})}{cs_i}\right) = \frac{\partial f(\vec{x})}{\partial x_i},\ x_i' = x_i - \eta \frac{\partial f(\vec{x})}{\partial x_i}$$