Toward the establishment of new image recognition method based on machine learning by quantum computers.

Author: Keiichiro HAYASHI, Nada High School

Prepared as Final Research Report for the Advanced Stage in the GSC/ROOT Program.

Advisors: Satofumi SOUMA, Kobe University

Date: February 28, 2022

Abstract

Quantum image recognition machine learning was performed by assembling a quantum circuit that mimics a classical neural network. The features of this study include: minimizing the number of calculations by using the simultaneous perturbation method; improving learning accuracy and efficiency by finding the gradient vector to optimize the weight parameters; and designing an algorithm that can handle an increase in the number of qubits used and weight layers. 6 qubits were used for the weights. The number of layers was also verified by trying several different ways. Although the $C^{\otimes n}Z$ gate alone did not demonstrate the usefulness of this method, it did complete the general framework of a new quantum image recognition machine learning method. We expect that the $C^{\otimes n}RZ$ gate will improve the learning accuracy and learning efficiency by utilizing the high representativeness of the input values and the gradient vector.

Introduction

In this research, based on the concept of quantum machine learning, especially quantum neurons, which is proposed in the following paper as one of the application algorithms of quantum computers that have been attracting attention in recent years, we will tackle the unresolved issue of embodiment of the input and output parts for these quantum neurons and optimization of the weights by machine learning.

The principle of quantum machine learning has been proposed in previous research, but when applying it to image recognition as a practical example, it is difficult to determine how to use the input image as input to the quantum neuron, and how to judge that the image has been recognized based on the result of the quantum process. In addition, we have not sufficiently studied the "issues that emerge when trying to apply the quantum process to specific problems," such as how to determine whether an image has been recognized based on the results of the quantum process. By confronting these practical issues, we believe that we can reveal new similarities and differences with conventional methods. In addition, we attempted a more practical image recognition machine learning by preparing multiple weight layers in the process of image discrimination. In this research, we will ignore the effect of noise and tackle this problem using Qiskit, an open source for quantum programming, to build a foundation for practical applications.

Materials and Methods / Theory

In a neural network in classical machine learning, the process of mapping input information to nodes, multiplying them by 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 1). In this study the inspiration for how to map the input information to quantum states and how to multiply the weights was derived 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.

$$\vec{i} = \begin{pmatrix} i_0 \\ i_1 \\ \vdots \\ i_{m-1} \end{pmatrix}, \vec{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{m-1} \end{pmatrix} \tag{0.1}$$

The condition is $i_j, w_j \in \{-1,1\}$, which defines these quantum states $|\psi_i\rangle, |\psi_w\rangle$.

$$|\psi_i\rangle = \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} i_j |j\rangle, |\psi_w\rangle = \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} i_w |j\rangle \tag{0.2}$$

$|j\rangle \in \{|00...00\rangle, |00...01\rangle, ..., |11...11\rangle\}$, and interprets a sequence of numbers consisting only of zeros and ones to be the binary notation for j, a decimal number. The quantum bits are $|0\rangle, |1\rangle$, respectively, and the quantum state is $|0\rangle$ if all N quantum states are $|00...00\rangle \equiv |0\rangle^{\otimes N}$. $|\psi_i\rangle, |\psi_w\rangle$ are represented by the superposition of these multiple quantum states. Using this idea, let us consider the structure of a simple quantum circuit (Fig 2). Here, U_i is a unitary gate that creates the quantum state $|\psi_i\rangle$ from the initial state $|0\rangle^{\otimes N}$. U_w is a unitary gate that multiplies the weights expressed by $|\psi_w\rangle$. Finally, we assume that the firing is left to the inversion of the Ancilla qubit by the $C^{\otimes N}X$ gate.

$$U_i |0\rangle^{\otimes N} = |\psi_i\rangle, U_w |\psi_w\rangle = |1\rangle^{\otimes N} = |m-1\rangle \tag{0.3}$$

In this case, $m = 2^N$. When U_w is weighted $|\psi_w\rangle$, if the quantum state immediately before

U_w is applied is $|\psi_w\rangle$, then all N qubits after U_w are $|1\rangle$, the $C^{\otimes N}X$ gate inverts the ancilla qubit, and the measured value is 1. When U_w is applied to $|\psi_i\rangle$ corresponding to the input information,(1.4)

$$|\phi_{i,w}\rangle \equiv U_w|\psi_i\rangle = \sum_{j=0}^{m-1} c_j |j\rangle \tag{0.4}$$

Using the definitions in (Eq 1.3) and (Eq 1.4), the inner product of the two quantum states is(Eq 1.15).

$$\langle\psi_w|\psi_i\rangle = \langle\psi_w|U_w^{\dagger}U_w|\psi_i\rangle = \langle m-1|\phi_{i,w}\rangle = c_{m-1} \tag{0.5}$$

$|c_j|^2$ denotes the probability that the quantum state of N qubits is $|j\rangle$ just before the $C^{\otimes N}X$ gate is applied. Therefore, $|c_{m-1}|^2$ denotes the probability that the quantum state is $|11...11\rangle$, i.e., the probability that the ancilla qubit is inverted.

Consider the breakdown of U_i, U_w. For simplicity, consider the case of $N=2$. If each square in a black-and-white image of 4 squares is black, assign -1 or that is white, assign to i_n ($i_0, i_1, i_2, i_3, \in \{-1,1\}$), and denote $|\psi_i\rangle = \frac{1}{\sqrt{4}}\left(i_0|00\rangle + i_1|01\rangle + i_2|10\rangle + i_3|11\rangle\right)$. If each square is black, assign 0 to j_n, and white, assign 1 to j_n, and each image can be represented by $k_i \equiv j_3 j_2 j_1 j_{0(2)}$ (Fig 3). Immediately after applying $H^{\otimes 2}$ to the initial state $|0\rangle^{\otimes 2}$, the image is $j_0 = j_1 = j_2 = j_3 = 1$. Here, I introduce the $C^{\otimes n}Z$ gate.

$$C^{\otimes n}Z_{\vec{l}}|j\rangle = \begin{cases} -|j\rangle \ if \bigcap_{i=0}^{N-1} j_i \geq l_i \\ |j\rangle \ otherwise \end{cases} \tag{0.6}$$

$\vec{l}$ has N elements ($l_i \in \{0,1\}$). The Z-gate is an operation that rotates the Bloch sphere around the Z-axis by π. Essentially, a multi-control gate applies an operation to a target qubit when all specific qubits are $|1\rangle$. Applying the operation to the $|0\rangle$ quantum state causes no change,

while applying it to the $|1\rangle$ quantum state causes a phase inversion. (fig 4). Therefore, the multi-control Z-gate is phase-reversed when all quantum states of the "specific qubit" and the target qubit are $|1\rangle$ (fig 5). $\vec{l}$ is an array of "specific qubits" and the location of the target qubit is set to 1, and all other locations are set to 0. When $N=2$, the $C^{\otimes 0}Z_{\{1,0\}}$ gate (fig 6.1) inverts the sign of $|10\rangle,|11\rangle$, the $C^{\otimes 0}Z_{\{0,1\}}$ gate (fig 6.2) inverts the sign of $|01\rangle,|11\rangle$, and the $C^{\otimes 1}Z_{\{1,1\}}$ gate (fig 6.3) inverts the sign of $|11\rangle$. The following table shows the results. (Eq. 1.6) generalizes this. $n+1$ is the number of 1's in $\vec{l}$. However, since these gates cannot invert the sign of $|00\rangle$, the X gate is used to adjust by exchanging the coefficients of the bits. Based on this, the contents of $N=2$ in the case of U_i,U_w are written out (fig. 7).

Let us take the circuit of $k_i=k_w=2$ as an example (fig 8). In U_i, the quantum state after the $C^{\otimes 0}Z_{\{0,1\}}$ and $C^{\otimes 1}Z_{\{1,1\}}$ gates is $\frac{1}{\sqrt{4}}\left(-|00\rangle-|01\rangle+|10\rangle-|11\rangle\right)$ because after the $\frac{1}{\sqrt{4}}\left(|00\rangle+|01\rangle-|10\rangle+|11\rangle\right)$ and $X^{\otimes 2}$ gates are applied, it is replaced by $|00\rangle\leftrightarrow|11\rangle,|01\rangle\leftrightarrow|10\rangle$ ($0010_{(2)}=2$). Then $C^{\otimes 0}Z_{\{0,1\}}$ gate and $C^{\otimes 1}Z_{\{1,1\}}$ gate are applied in U_i, and it becomes $\frac{1}{\sqrt{4}}\left(|00\rangle+|01\rangle-|10\rangle+|11\rangle\right)$, and after $H^{\otimes 2}$ it is $|11\rangle$, and finally Ancilla qubit is inverted and observed 1. Only in $k_i=k_w$ the observed value is 1. Since the number of qubits is 2^{2^N}, it was necessary to make a function for when the number of qubits increases (Appendix).

The final observed value is always 0 or 1, and which one is observed depends on probability. The average of the observed values was taken and this was the expected value of the inner product.

$$f\left(k_i,k_w\right)\equiv E\left(\langle\psi_w|\psi_i\rangle\right)=\frac{1}{\kappa}\sum_{k=1}^{\kappa}\langle\psi_w|\psi_i\rangle \tag{0.7}$$

This time it was done as $\kappa=1024$.

We set up a problem to identify whether the image is ○ or × by learning features and optimizing weights from 100 images of ○ and ×'s in 8×8 cells (fig. 9). In order to achieve a model that outputs 1 for ○ and 0 for ×, the loss function was designed as follows.

$$L\left(k_w\right)=\frac{1}{2m}\sum_{j=1}^{m}\left[\left\{1-f\left(k_{i,j}^{circle},k_w\right)\right\}^2+\left\{f\left(k_{i,j}^{cross},k_w\right)\right\}^2\right] \tag{0.8}$$

k_i was assumed to be an 8×8-digit binary number created from the black and white of each square in the image and then foolishly converted to a decimal number. Initial values were given to $k_{w,0}$ as appropriate, and k_w was updated in the direction that minimized the loss function.

$$k_{w,t+1}=k_{w,t}-\eta\frac{dL}{dk_{w,t}} \tag{0.9}$$

$$L'\left(k_{w,t}\right)=\frac{L\left(k_{w,t}+dk_{w,t}\right)-L\left(k_{w,t}-dk_{w,t}\right)}{2dk_{w,t}} \tag{0.10}$$

The change in weights must be differentiated for each coefficient of each quantum state of $\left|00...00\right\rangle\sim\left|11..11\right\rangle$, which is regarded as a bit.

$$L'\left(k_{w,t}\right)=\left(\frac{\partial L\left(k_{w,t}\right)}{\partial i_{t,0}},\frac{\partial L\left(k_{w,t}\right)}{\partial i_{t,1}},\cdots,\frac{\partial L\left(k_{w,t}\right)}{\partial i_{t,2^N-1}}\right)^{\mathrm{T}} \tag{0.11}$$

Let $i_{t,j}$ be the coefficient of $\left|j\right\rangle$ within the quantum state represented by $k_{w,t}$ after t updates. $\partial i_{t,j}$ be the change in each square, that is, the change of white to black and black to white. Consider a finite field with phase 2 for each digit in the binary notation of $i_{t,j}$.

$$\frac{\partial L\left(k_{w,t}\right)}{\partial i_{t,j}}=\frac{L\left(\left(k_{w,t(2)}+e_{t,j}\right)_{(10)}\right)-L\left(k_{w,t}\right)}{1}=L\left(\left(k_{w,t(2)}+e_{t,j}\right)_{(10)}\right)-L\left(k_{w,t}\right) \tag{0.12}$$

$e_{t,j}$ is a binary number with only the j th digit being 1 and the rest 0. The difference

approximation in (Eq. 1.10) is inconvenient in this case, so we adopt the form (Eq. 1.12): to obtain the gradient vector of $L\left(k_{w,t}\right)$ for a single update of k_w, the value of L must be obtained $\left(2^N+1\right)$ times, which requires a huge amount of computation. Therefore, a simultaneous perturbation optimization method is introduced to reduce the computational complexity.

$$k_{w,t+1} = k_{w,t} - \eta \Delta k_{w,t} \tag{0.13}$$

$$\Delta i_{t,j} = \frac{L\left(k_{w,t} + c_t\right) - L\left(k_{w,t} - c_t\right)}{2c_{t,j}} \tag{0.14}$$

$$\Delta k_{w,t} = \Delta i_{t,2^N-1} \Delta i_{t,2^N-2} \cdots \Delta i_{t,1} \Delta i_{t,0} \tag{0.15}$$

$\Delta k_{w,t}$ represents the estimated value of the partial derivative in $k_{w,t}$. $c_t, c_{t,j}$ represents the perturbation value and its j th digit in binary conversion, respectively. Convert to binary and process $k_{w,t} - \Delta k_{w,t}, k_{w,t} + c_t$ as a finite field of phase 2 for each digit. The perturbation c_t was generated using Bernoulli random numbers based on probability distributions. $c_{t,j}$ is i.i.d. (independent homodistribution) for t, j, its distribution is symmetric around 0 and its magnitude is uniformly finite. Since $L\left(k_{w,t} + c_t\right) - L\left(k_{w,t} - c_t\right)$ is constant for all j s in (Eq. 1.14), only two values of L are needed to obtain the gradient vector of $L\left(k_{w,t}\right)$ for a single update of k_w, thus reducing the computational complexity compared to (Eq. 1.12). We adopt (Eq 1.16) because $\Delta i_{t,j}, c_{t,j}$ are limited to 0 or 1 in this study, respectively.

$$\Delta i_{t,j} = \begin{cases} 1 \ \textit{if} \ \left(L\left(k_{w,t} + c_t\right) - L\left(k_{w,t}\right)\right) < 0 \cap \ c_{t,j} = 1 \\ 0 \ \textit{otherwise} \end{cases} \tag{0.16}$$

It is also generally suggested that increasing the number of intermediate layers in a neural

network improves learning accuracy and efficiency, and we verified this in the case of $k_w \in \Re^{\gamma}$, i.e., when the weights are multilayered into γ layers (fig. 10). In accordance with (Eq. 1.4), it is defined as follows.

$$\left|\phi'_{i,w}\right\rangle \equiv \prod_{s=1}^{\gamma} U_{w,s} \left|\psi_i\right\rangle \tag{0.17}$$

We designed a structure in which U_w is repeated γ times as it is; $\overrightarrow{k_w}$ is a 1-dimensional vector with γ elements, and the γ parameters are independent of each other.

$$g\left(k_i, \overrightarrow{k_w}\right) \equiv E\left(\left\langle \psi'_w \middle| \psi_i \right\rangle\right) = \frac{1}{\kappa} \sum_{k=1}^{\kappa} \left\langle \psi'_w \middle| \psi_i \right\rangle \tag{0.18}$$

For convenience, we denote it as (Eq. 1.18).

$$L\left(\overrightarrow{k_w}\right) = \frac{1}{2m} \sum_{j=1}^{m} \left[\left\{1 - g\left(k_{i,j}^{circle}, \overrightarrow{k_w}\right)\right\}^2 + \left\{g\left(k_{i,j}^{cross}, \overrightarrow{k_w}\right)\right\}^2 \right] \tag{0.19}$$

$$k_{w,s,t+1} = k_{w,s,t} - \eta \Delta k_{w,s,t} \tag{0.20}$$

$$\Delta i_{s,t,j} = \frac{L\left(\overrightarrow{k_{w,t}} + \overrightarrow{c_t}\right) - L\left(\overrightarrow{k_{w,t}} - \overrightarrow{c_t}\right)}{2c_{s,t,j}} \tag{0.21}$$

$$\Delta k_{w,s,t} = \Delta i_{s,t,2^N-1} \Delta i_{s,t,2^N-2} \cdots \Delta i_{s,t,1} \Delta i_{s,t,0} \tag{0.22}$$

In this case, the calculation of the loss function L itself would be longer with more layers, but to find the gradient vector of $L\left(\overrightarrow{k_{w,t}}\right)$ for a single $\overrightarrow{k_w}$ update, we only need to calculate the value of L twice. After all, since $\Delta i_{s,t,j}, c_{s,t,j}$ is limited to 0 or 1 in this study, respectively, (Eq. 1.24) is adopted.

$$\Delta i_{s,t,j} = \begin{cases} 1 \;\; if \;\; \left(L\left(\overrightarrow{k_{w,t}} + \overrightarrow{c_t}\right) - L\left(\overrightarrow{k_{w,t}}\right)\right) < 0 \cap \; c_{s,t,j} = 1 \\ 0 \;\; otherwise \end{cases} \tag{0.23}$$

We attempted image recognition machine learning while updating the weights 500 times each for $\gamma = 1, 3, 5$ as the initial value $k_{w,s,0} = 0$ of each parameter, and compared the results.

Results

(Eq 1.19) is plotted on the horizontal axis with epoch as the result of the calculation Energy. The weights are updated by adopting the weights at the time when the previous loss function reached its minimum value. The variation of the minimum value of the loss function is also plotted as Minimum Energy (fig 11). The initial and final values are summarized (fig 12).

Discussion

From the validation results, the weights were updated, and the loss function was successfully reduced. In this verification, there was no improvement in learning accuracy by increasing the number of weight layers. Although it cannot be called a success of machine learning in terms of accuracy, it can be said to be a successful validation with a small amount of test data and a minimum number of parameters.

This study focused on establishing a framework for the author's own method of quantum image recognition machine learning. The advantage of using a quantum computer is that it can make full use of the computational power of quantum superposition states.

As expressed as $|j\rangle \in \{|00...00\rangle, |00...01\rangle, ..., |11...11\rangle\}$, N qubit combinations of 0s and 1s can be probabilistically superimposed by the H gate. The coefficients of these quantum states and the probability of observation can be manipulated by gate operations in quantum circuits. The $C^{\otimes n}Z$ gate used this time rotates the phase by π around the Z axis on the Bloch sphere when a particular qubit is 1, i.e., in short, it is an operation that switches the sign of the coefficients, so if there is no last $H^{\otimes N}$ in (fig. 10), the probability of 1 firing is $\frac{1}{2^N}$ no matter which timing $C^{\otimes n}X$ is applied and observed. It is expected that the use of $C^{\otimes n}RZ$ gates, which allow phase rotation to be set as a continuous value in the future, will increase expressiveness. In this case, where the coefficients are 1 or -1 (strictly speaking, $\pm 2^{-\frac{N}{2}}$), N qubits could be used to represent

a black-and-white image up to $2^{\frac{N}{2}} \times 2^{\frac{N}{2}}$ squares in a quantum state. In this case, k_w takes only integer values, and even if one tries to figure out the amount of change by forceful differentiation, the minimum value of the minute change is 1. Also, depending on the value of k_w, there may be too large a difference in the amount of change. Following a series of inner product calculation methods, we can imagine that a change of 1 in k_w is sometimes a change of only one square assigned as a coefficient of $|11...11\rangle$, and sometimes a change of several specific squares during binary carry-up. In the classical process, the partial differentiation is done for the weights of each node respectively to obtain the amount of micro change, but in this case, the coefficients of each quantum state of $|00...00\rangle \sim |11..11\rangle$, which are regarded as bits, only changed between -1 or 1, and were not differentiable in multiple ways, forcing the (Eq. 1.21) to (Eq 1.23). This study adopts a method that will live on even when $C^{\otimes n}RZ$ gates are introduced in the future.

The forced procedure of converting to binary numbers and using the finite body concept for each digit in the first place can also be solved by introducing the $C^{\otimes n}RZ$ gate. This study employs the seemingly incoherent process of differentiating each digit on a finite field of phase 2, because the coefficient i_j can be set to a continuous value using the $C^{\otimes n}RZ$ gate instead of the $C^{\otimes n}Z$ gate. After all, if we use the "phase" θ, we can go around in 2π, so there is no need to think in terms of a finite field. Although it has been a rough approach of updating parameters randomly and adopting them if there is a more appropriate one than before, we are convinced that enabling partial differentiation using continuous values will improve learning accuracy and efficiency. In that case, it makes no sense to make each digit a continuous value by performing a binary conversion this time, but this is an issue for future study.

The perturbation value was verified using Bernoulli random numbers with an expected value of 0.5, but there must be other optimal values. We also came up with the idea of incorporating the noise generated in the quantum computer into the perturbation. We also found a previous study in which the gain coefficient η was also varied, and further study is needed in the future.

Conclusions

Although we were not able to demonstrate the usefulness of the quantum image recognition machine learning method employed in this study, we were able to establish the general framework of our method. The strengths of this method are the use of simultaneous perturbation methods to reduce computational complexity and the use of multiple layers of weights. With the introduction of $C^{\otimes n}RZ$ gate, we expect to improve the learning accuracy and learning efficiency by making use of the representativeness of the input values and the gradient vector.

Acknowledgements

This work was supported by ROOT program. I am grateful to Prof. Satofumi Souma for his kind guidance and advice in this research.

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).

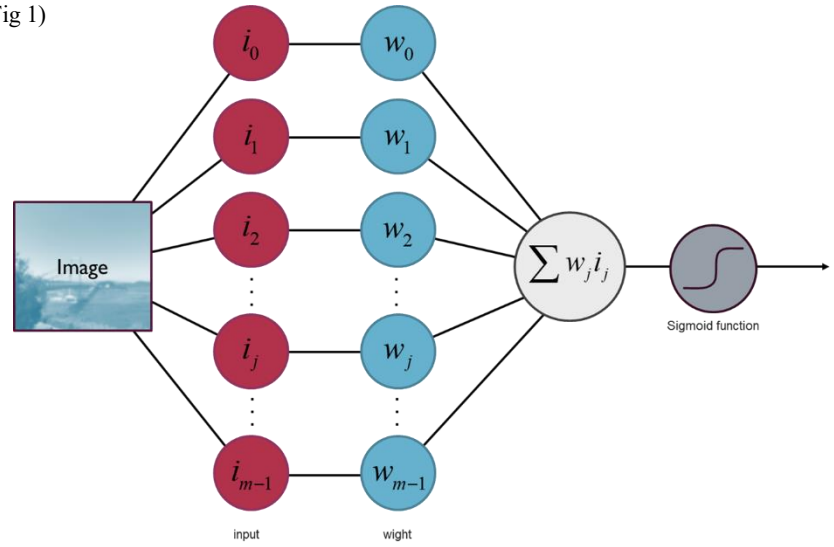
Appendix

The algorithm for automatic generation of quantum circuits is shown. The function itself returns the inner product value.

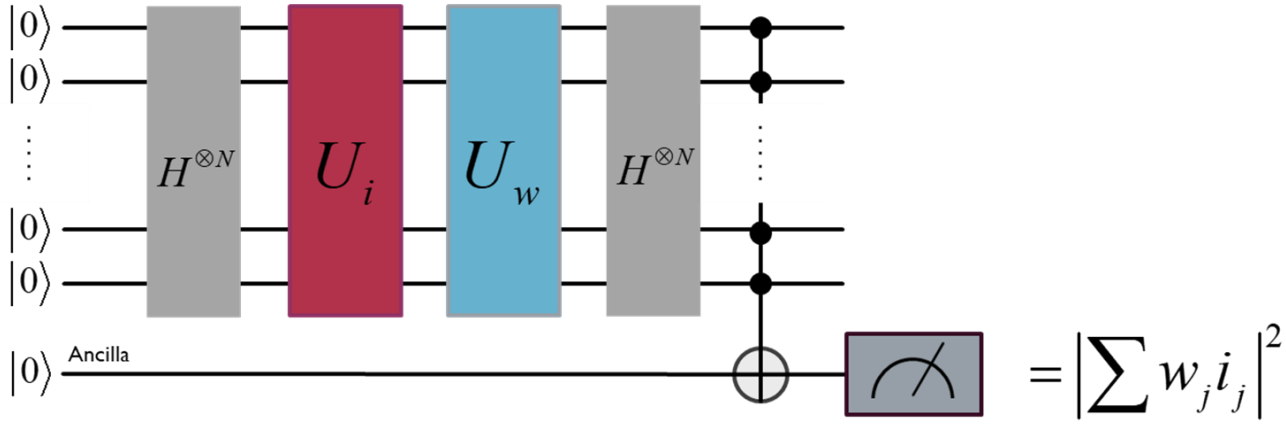
```
def function_n(n,ki,kw,kw_n):
    #n は使用する qubit 数,ki は入力値,kw は長さγの 1 次元ベクトル,kw_n は層数γ
  #入力値、重みパラメータにはそれぞれ0 ~ 2^2^N −1の数値を与えられる。
    #bit は要素数n×2^n の配列
    #gate は要素数n×2^n の配列
    #A は要素数2^n×2^n の配列
    #B は要素数2^n の配列
    #C は要素数2^n の配列
    #D は要素数2^n×kw_n の配列
    #E は要素数2^n×kw_n の配列
    #FLAG は要素数2^n の配列
    #X は要素数n の配列
    for i in range(0,pow(2,n)):
        for j in range(0,n):
            bit[i][n-1-j]=bin4(i,n,j)
    #bit 配列は 0-pow(2,n)の 2 進数表記
      for i in range(0,pow(2,n)):
        for j in range(0,n):
            bit[i][n-1-j]=bin4(i,n,j)

    a=1
    #a は正方行列 A の横の番号.予め全て初期値 1 を入れてある.0 行目は全て初期値の 1.
    for j in range(1,n+1):
        for i in range(0,pow(2,n)):
            if sum(bit[pow(2,n)-1-i])==j:
    #bit 配列を下から、和が 1 からだんだん増やして見ていく.これによってゲートの順になる.
                gate[a]=bit[pow(2,n)-1-i]
    #ついでに gate 配列作成、gate[0]は意味ない
                for k in range(0,pow(2,n)):
                    for l in range (0,n):
                        if((bit[k][l]-bit[pow(2,n)-1-i][l])<0):
    #任意の bit 配列が上の if 文を満たす配列を包含しないなら
                            A[k][a]=0
    #包含しない bit 配列はそのゲートで反転できない
                            break
                a+=1
    #まず縦に|00...0>-|11...1>順番に、横にゲートを順に考えて、それがそのビットを反転させるなら 1。一番左は便宜上で無視。

    for i in range(pow(2,n)):
        C[pow(2,n)-1-i]=1-bin4(ki,pow(2,n),i)
    #反転するなら 1,反転しないなら 0。H ゲート適用後は全ビットが 1 で、0 にするべきビットは 1、そのまま 1 のビットは 0。
    if C[0]==1:
    #C[0](0 だけのビット)は(C^(0-(n-1)))Z ゲートでは反転できないから、他のビットと入れ替えるしかない。
        for i in range (1,n+1): #bit 配列の和が 1 から n の間で
            for j in range(pow(2,n)):
```

```
                    if sum(bit[pow(2,n)-1-j])==i and C[pow(2,n)-1-j]==0:
    #下から順に考えて bit 配列の和が i と一致してそのビットを反転させなく良いとき
                        for k in range(pow(2,n)):
                            if FLAG[decimal(n,bit[k])]==0:
    #まだ交換がなされてないとき
                                C[decimal(n,bit[k])],C[decimal(n,(bit[pow(2,n)-1-j]-bit[k])%2)]=C[decimal(n,(bit[pow(2,n)-
1-j]-bit[k])%2)],C[decimal(n,bit[k])]
    #和が(差でもいいけど)bit[pow(2,n)-1-j]になる組み合わせで交換し合う
                                FLAG[decimal(n,bit[k])],FLAG[decimal(n,(bit[pow(2,n)-1-j]-bit[k])%2)]=1,1
    #交換済みのマーカー
                        for k in range(n):
                            X[k]=bit[pow(2,n)-1-j][k]
                        break
                    else:
                        continue
                if C[0]==0:
    #C の入れ替えが済んだなら終了
                    break
    B=solve(A,C)
    #連立方程式で解を求める。
    for i in range(pow(2,n)):
        B[i]=round(B[i])%2
    #位相 2 の有限体
    for h in range(kw_n):
        for i in range(pow(2,n)):
            D[h][i]=(sum(bit[i])-(1-bin4(kw[h],pow(2,n),pow(2,n)-i-1)))%2
    #反転するなら 1,反転しないなら 0
        if(D[h][0]==1):
            for i in range(pow(2,n)):
                D[h][i]=1-D[h][i]
    #|00...0>が反転すべきなら、するしないを入れ替える
        E[h]=np.dot(np.linalg.inv(A),D[h])
        for i in range(pow(2,n)):
            E[h][i]=round(E[h][i])%2

    q = QuantumRegister(n+1, 'q')
    c = ClassicalRegister(1, 'c')
    qc = QuantumCircuit(q, c)

    for i in range(n):
        qc.h(i)
        qc.barrier(i)
    if B[0]==1:
        qc.z(0)
        qc.x(0)
        qc.z(0)
    else:
        for i in range(1,pow(2,n)):
            if B[i]==1:
                MCZ(gate[i],sum(gate[i]),n,qc,q)
    for i in range(n):
        if X[i]==1:
            qc.x(i)
    for h in range(kw_n):
        for i in range(1,pow(2,n)):
            if E[h][i]==1:
                MCZ(gate[i],sum(gate[i]),n,qc,q)
    for i in range(n):
        qc.h(i)
    MCX(n+1,qc)
    qc.measure(q[n],c[0])

    shots=1024
    simulator = Aer.get_backend('qasm_simulator')
    result = execute(qc, backend = simulator, shots=shots).result()
    counts = result.get_counts(qc)
    num_ones = counts.get('1', 0)/shots

    qc.draw('mpl')
    return num_ones

    #MCX(m,qc)で m 番目の qubit に X ゲートを適用する関数を用意.
    #MCZ(bit,m,n,qc,q)で bit[n]の 1 に当たる m 個の qubit 間で CZ ゲートを適用する関数を用意.
```

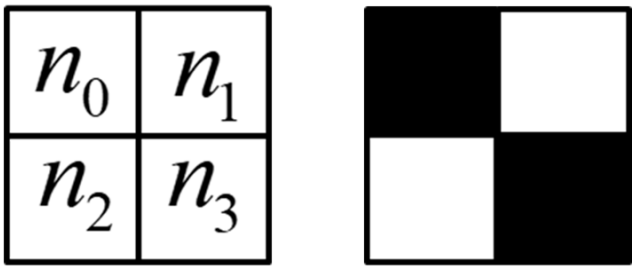
(Fig 1)



(Fig 2)



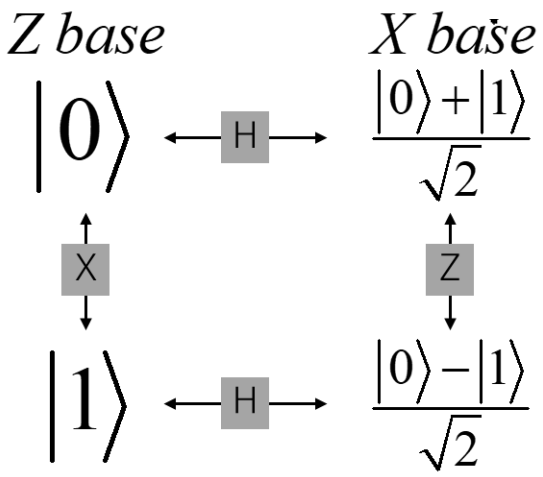
(Fig 3)



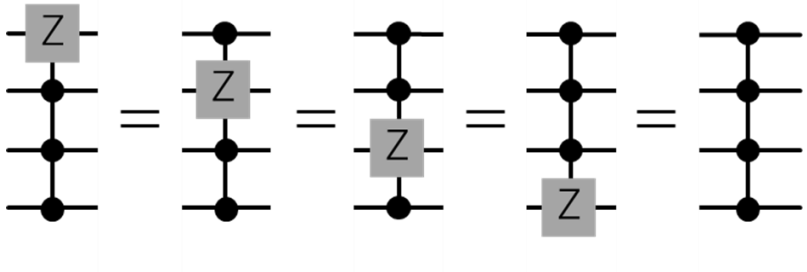
$k = n_3 n_2 n_1 n_0 \quad ex) k = 6$

$black : 0, white : 1$

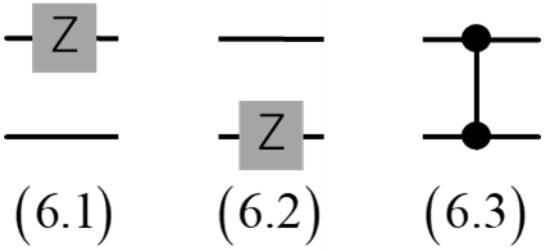
(Fig 4)



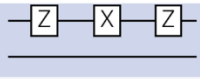
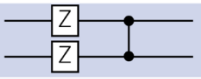
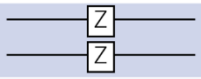
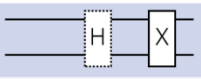
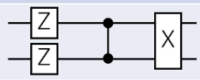
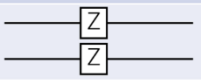
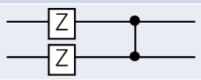
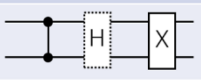
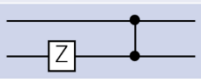
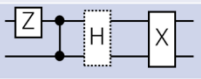
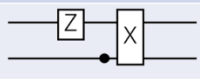
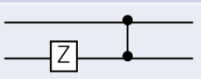
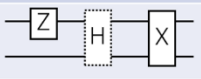
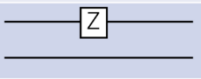
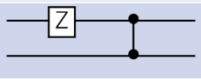
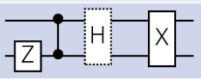
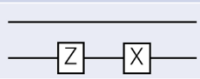
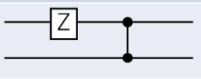
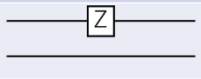
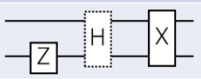
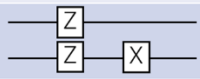
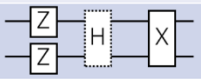
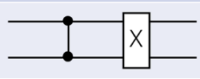
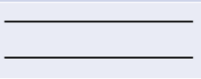
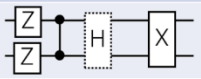
(Fig 5)



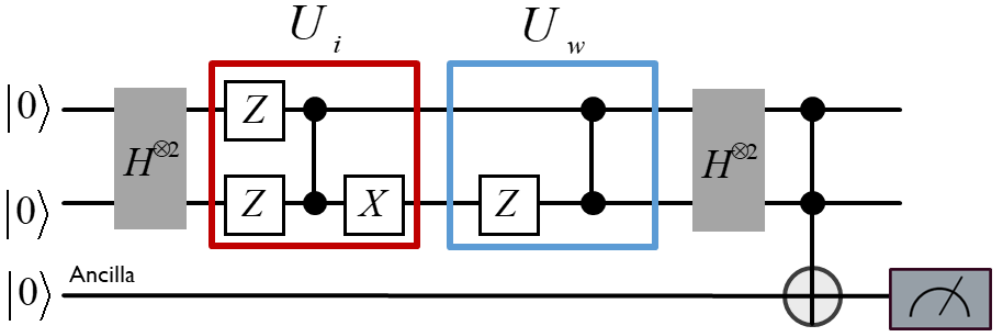
(Fig 6)



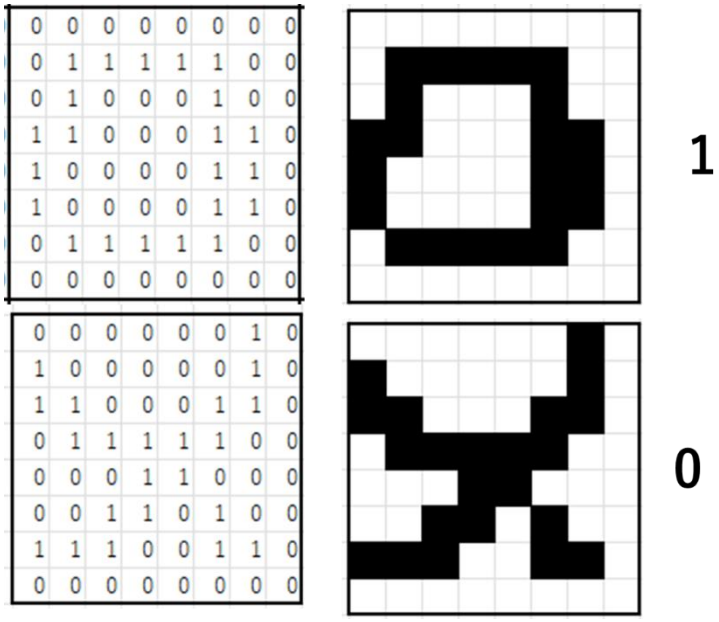
(Fig 7)

k_i	U_i	k_i	U_i	k_w	U_w	(U_w)
0		8		0,15		
1		9		1,14		
2		10		2,13		
3		11		3,12		
4		12		4,11		
5		13		5,10		
6		14		6,9		
7		15		7,8		

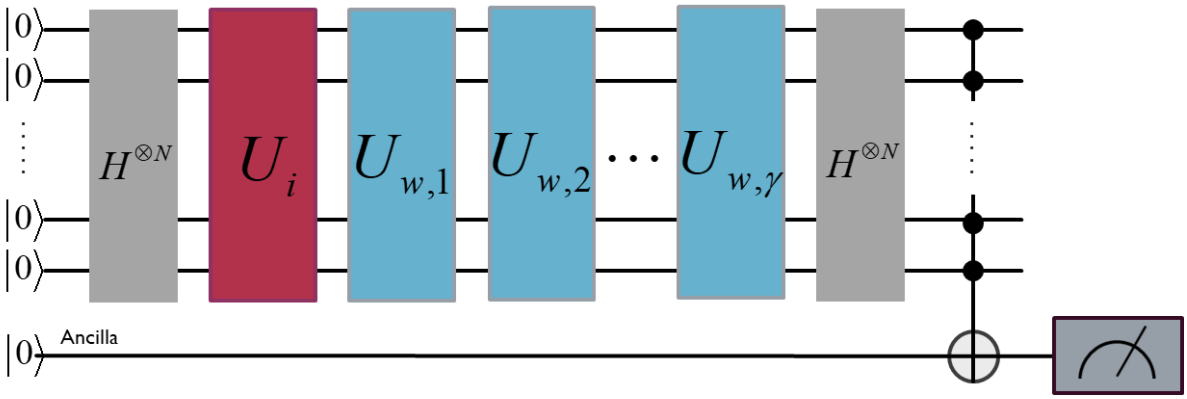
(Fig 8)



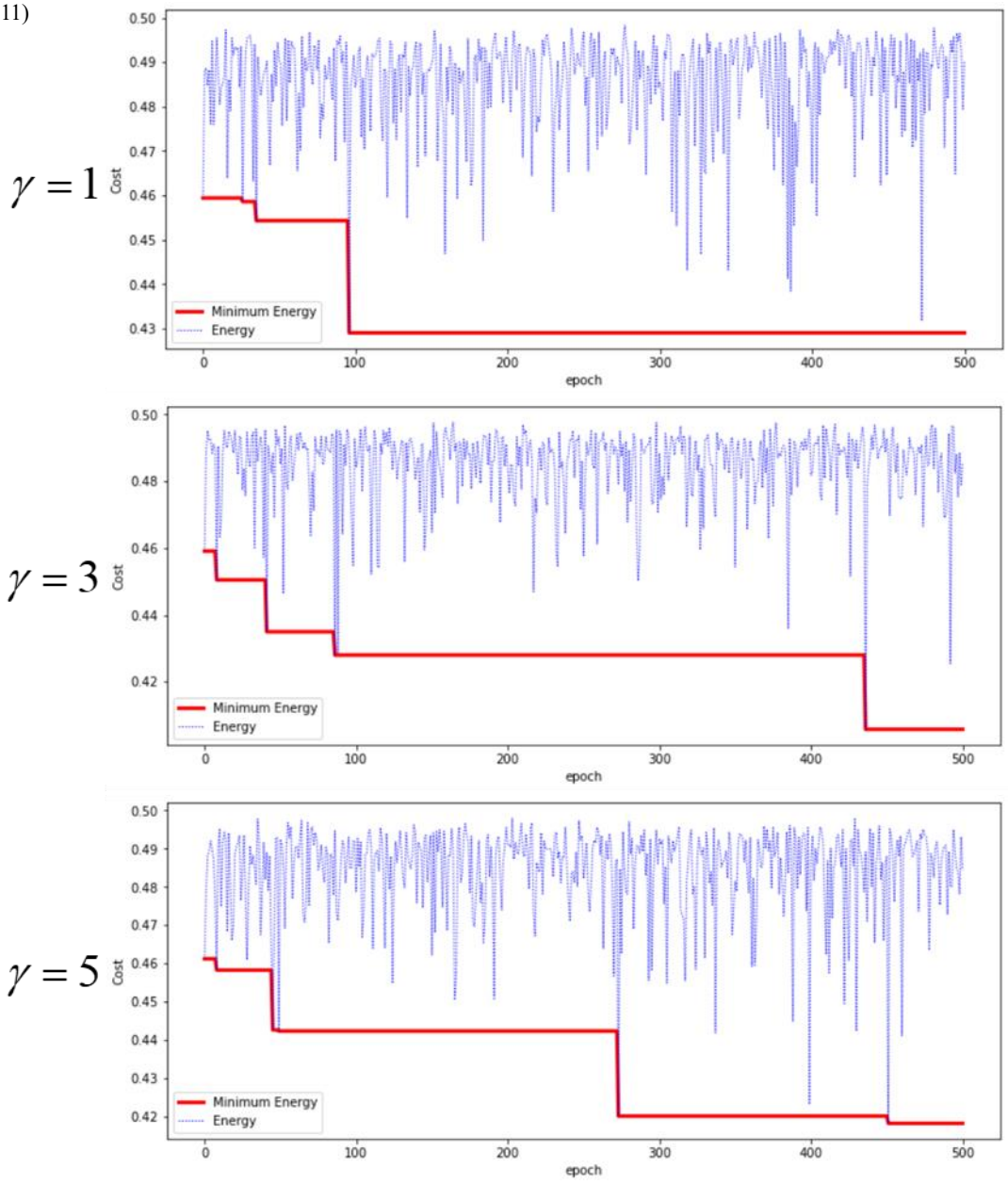
(Fig 9)



(Fig 10)


$|0\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$
Ancilla
$H^{\otimes N}$
U_i
$U_{w,1}$
$U_{w,2}$
$U_{w,\gamma}$
$H^{\otimes N}$

(Fig 11)


$\gamma = 1$
$\gamma = 3$
$\gamma = 5$
Cost
epoch
Minimum Energy
Energy

(Fig 12)

	$\gamma = 1$	$\gamma = 3$	$\gamma = 5$
initial value	0.4594	0.4591	0.4612
final value	0.4290	0.4057	0.4182