一般社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report

MCRZ ゲートを用いたニューラルネットワークの実装
－ゲート式量子コンピュータによる画像認識機械学習の新手法の確立に向けて－

林 慶一郎†

†私立灘高等学校 〒658-0082 兵庫県神戸市東灘区魚崎北町 8 丁目 5-1
E-mail: †keiichiroharry884@gmail.com

あらまし ゲート式量子コンピュータでの任意の qubit 数で行う計算において、複数の *MCRZ* ゲートを組み合わせた量子回路を自動生成して古典的なニューラルネットワークを模した画像認識機械学習を試みた。 *MCRZ* ゲートではある一つの量子ビットに対して、他の量子ビットの状態を条件として Z 軸周りに任意の角度だけ回転させる操作を行える。本研究の特色としては、同時摂動法を用いて計算回数を極力減らしたこと、重みパラメータ最適化のための勾配ベクトルの設計部分、使用する量子ビットの数や重み層が増えた場合にも対応したアルゴリズムを設計したしたことだ。計算量の制約もあり当手法の有用性までは示せなかったが、新たな量子画像認識機械学習手法の大枠を完成させるに至った。

キーワード 量子コンピュータ 画像認識 機械学習 ニューラルネットワーク

Implementation of Neural Networks with *MCRZ* Gates
－Toward the Establishment of New Method for Image Recognition Machine Learning in gate-based Quantum Computers－

Keiichiro HAYASHI†

† Nada High School 8-5-1 Uozakikita-machi, Higashinada-ku, Kobe-city, Hyogo, 658-0082 Japan
E-mail: †keiichiroharry884@gmail.com

Abstract In a calculation performed with an arbitrary number of *MCRZ* qubits in a gate-based quantum computer, I attempted image recognition machine learning that mimics a classical neural network by automatically generating a quantum circuit that combines multiple *MCRZ* gates. In this gate, an operation to rotate one qubit by an arbitrary angle around the Z-axis can be performed on one qubit, subject to the states of the other qubits. The features of this study are that the number of calculations was minimized by using the simultaneous perturbation method, the design of the gradient vector for optimizing the weight parameters, and the design of an algorithm that can handle an increase in the number of qubits used and weight layers. Although I was unable to demonstrate the usefulness of our method due to computational limitations, I was able to complete the general framework of a new quantum image recognition machine learning method.

Keywords Quantum Computer, Image Recognition, Machine Learning, Neural Networks

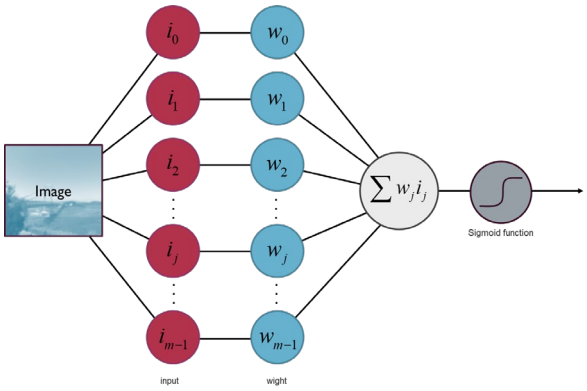
1. Introduction

In this research, based on the idea 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, I will tackle the unexplored issues related to the embodiment of the input-output part for these quantum neurons and the optimization of the weights by machine learning.

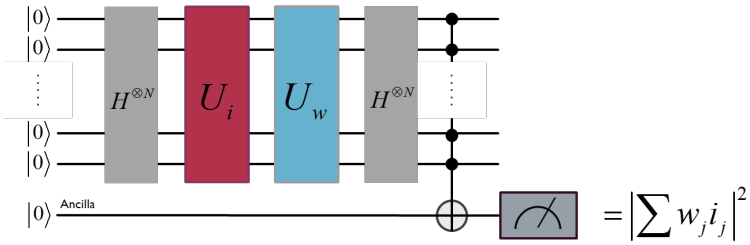
The principle of quantum machine learning has been proposed in previous research, but when it is applied to image recognition as a practical example, there are many "concrete issues" such as how to use the input image as input to quantum neurons, how to design perceptron, and how to determine that an image has been recognized from the result of quantum processes. How to design the perceptron, and how to determine that an image has been "recognized" from the results of the quantum process have not yet been sufficiently investigated. By confronting such practical issues, it is thought that new similarities and differences with conventional methods can be made apparent. I also attempted more practical image recognition machine learning by preparing multiple layers of weights in the image discrimination process. In this study, I ignore the influence of noise and address this issue using Qiskit, an open source for quantum programming, to build a foundation for practical application.

2. Materials and Method/Theory

In a neural network in classical machine learning, the process of mapping input information to nodes, multiplying



(Fig.1 a simple structure of classical image recognition)



(Fig.2 a simple structure of quantum image recognition)

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.

$$\vec{i}=\begin{pmatrix} i_0 \\ i_1 \\ \vdots \\ i_{m-1} \end{pmatrix}, \vec{w}=\begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{m-1} \end{pmatrix} \tag{1.1}$$

The condition is $i_j, w_j \in \{e^{i\theta} \mid \theta \in \mathbb{R}\}$, which defines these quantum states $|\psi_i\rangle, |\psi_w\rangle$.

$$|\psi_i\rangle=\frac{1}{\sqrt{m}}\sum_{j=0}^{m-1} i_j |j\rangle,\ |\psi_w\rangle=\frac{1}{\sqrt{m}}\sum_{j=0}^{m-1} i_w |j\rangle \tag{1.2}$$

$|j\rangle \in \{|00...00\rangle, |00...01\rangle, ..., |11...11\rangle\}$, 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, I assume that the firing is left to the inversion of the Ancilla qubit by the $C^{\otimes N}X$ gate.

$$U_i|0\rangle^{\otimes N}=|\psi_i\rangle, U_w|\psi_w\rangle=|1\rangle^{\otimes N}=|m-1\rangle \tag{1.3}$$

$$|\phi_{i,w}\rangle \equiv U_w|\psi_i\rangle=\sum_{j=0}^{m-1} c_j |j\rangle \tag{1.4}$$

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

$$\begin{aligned}\langle\psi_w|\psi_i\rangle &= \langle\psi_w|U_w^{\dagger}U_w|\psi_i\rangle \\ &= \langle m-1|\phi_{i,w}\rangle = c_{m-1}\end{aligned} \tag{1.5}$$

$|c_j|^2$ 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.

The grayscale of a square is normalized by multiplying it by $\frac{\pi}{256}$ and assigning it to θ of $i_j \in \{e^{i\theta} \mid \theta \in \mathbb{R}\}$, of which there are 2^n assigned to each square. Now that the image information is stored in the array of $i_j \in \{e^{i\theta} \mid \theta \in \mathbb{R}\}$, the image is represented using gates based on this information. If each square is white, assign 1

to j_n, and black, assign -1 to j_n.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 $C^{\otimes n}RZ$ gate.

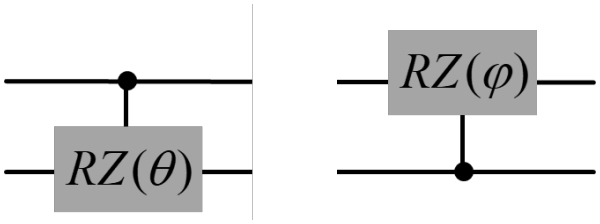
$$C^{\otimes n}RZ_{\vec{l},k}(\theta)|j\rangle = \begin{cases} e^{i(\pi/2-\theta/2)}|j\rangle \\ e^{i(\pi/2+\theta/2)}|j\rangle \end{cases} if \bigcap_{i=0}^{N-1} j_i \geq l_i \; and \begin{cases} j_k = 0 \\ j_k = 1 \end{cases} \\ = |j\rangle \qquad otherwise \tag{1.6}$$

$\vec{l}$ has N elements ($l_i \in \{0,1\}$). $RZ(\theta)$ gate is an operation that rotates the Bloch sphere around the Z-axis by θ. It is generally expressed in matrix form as follows.

$$RZ(\theta) = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix} \tag{1.7}$$

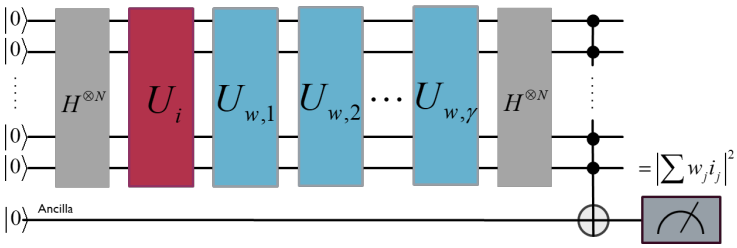
Essentially, a multi-control gate applies an operation to a target qubit when all specific qubits are $|1\rangle$. If the operation is applied to state $|0\rangle$, $e^{i(\pi/2-\theta/2)}$ is multiplied; if applied to state $|1\rangle$, $e^{i(\pi/2+\theta/2)}$ is multiplied. $\vec{l}$ is an array that shows where "specific qubits" is as 1, and all other locations are set to 0. k also shows the location of target qubit. When $N = 2$, the $C^{\otimes 1}RZ(\theta)_{\{1,0\},1}$ gate (Fig.3.1) multiplies $e^{i(\pi/2-\theta/2)}$ and $e^{i(\pi/2+\theta/2)}$ by the coefficients of $|10\rangle$ and $|11\rangle$, respectively, and the $C^{\otimes 1}RZ(\varphi)_{\{0,1\},0}$ gate (Fig.3.2) multiplies $e^{i(\pi/2-\varphi/2)}$ and $e^{i(\pi/2+\varphi/2)}$ by the coefficients of $|01\rangle$ and $|11\rangle$. With $\frac{1}{2}(|00\rangle+|01\rangle+|10\rangle+|11\rangle)$ as the initial state, the application of these two eventually results in $\frac{1}{2}\left(|00\rangle + e^{i\left(\frac{\pi}{2}-\frac{\varphi}{2}\right)}|01\rangle + e^{i\left(\frac{\pi}{2}-\frac{\theta}{2}\right)}|10\rangle + e^{i\left(\pi+\frac{\theta}{2}+\frac{\varphi}{2}\right)}|11\rangle\right)$.

(Eq.1.6) generalizes these operations. n is the number of 1's in $\vec{l}$. Using multiple $C^{\otimes n}RZ$ gate, this represents the grayscale of the pixel with each coefficient. Based on this, the contents of U_i, U_w are assembled.



(Fig.3.1/2 CRZ gates on a quantum computer)

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. It is also generally suggested that increasing the number of intermediate layers in a neural network improves learning accuracy and efficiency, and I verified this in the case of $k_w \in \Re^{\gamma}$, i.e., when the weights are multilayered into γ layers (Fig.4).



(Fig.4 quantum circuits including γ layers)

In accordance with (Eq.1.4), it is defined as follows.

$$|\phi'_{i,w}\rangle \equiv \prod_{s=1}^{\gamma} U_{w,s}|\psi_i\rangle \tag{1.8}$$

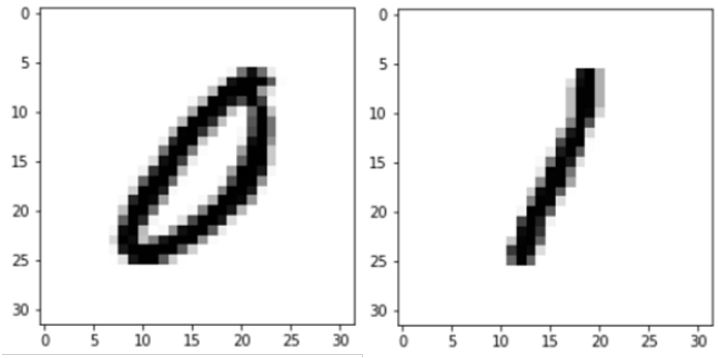
I designed a structure in which U_w is repeated γ times as it is.

$$f\left(k_i, \overrightarrow{k_w}\right) \equiv E\left(\langle\psi'_w|\psi_i\rangle\right) = \frac{1}{\kappa}\sum_{k=1}^{\kappa}\langle\psi'_w|\psi_i\rangle \tag{1.9}$$

$k_i \in \mathbb{R}^2$ is a grayscale array of pixels in the input image, from which the image data is represented in

quantum state; $k_w \in \mathbb{R}^2$ is an array of weight values to be multiplied by the input values. $\overrightarrow{k_w}$ is a 1-dimensional vector with γ elements, which each of it include information as same size as input images, and the γ parameters are independent of each other. This time it was done as $\kappa = 1024$.

In this case, image data with 0 and 1 labels were prepared from the MNIST data set. Originally, the data size of MNIST was 28×28, but I added two margins around it to make it 32×32(Fig.5).



(Fig.5) Examples of processed MNIST image

I designed the loss function as below.

$$L\left(\overrightarrow{k_w}\right) = \frac{1}{2m}\sum_{j=1}^{m}\left[\begin{array}{l}\left\{1 - f\left(k_{i,j}^{ones}, \overrightarrow{k_w}\right)\right\}^2 \\ \quad + \left\{f\left(k_{i,j}^{zeros}, \overrightarrow{k_w}\right)\right\}^2\end{array}\right] \quad (1.10)$$

Initial values were given to $\overrightarrow{k_{w,0}}$ as appropriate, and $\overrightarrow{k_{w,t}}$ was updated in the direction that minimized the loss function.

$$\overrightarrow{k_{w,t+1}} = \overrightarrow{k_{w,t}} - \eta\overrightarrow{\Delta k_{w,t}} \quad (1.11)$$

$$\begin{aligned}&\Delta k_{w,s,t,j} \\ &= \frac{L\left(\overrightarrow{k_{w,t}} + c\overrightarrow{s_t}\right) - L\left(\overrightarrow{k_{w,t}} - c\overrightarrow{s_t}\right)}{2cs_{s,t,j}} \\ &= \frac{L\left(\overrightarrow{k_{w,t}} + c\overrightarrow{s_t}\right) - L\left(\overrightarrow{k_{w,s,t}} - c\overrightarrow{s_t}\right)}{2c}s_{s,t,j}\end{aligned} \quad (1.12)$$

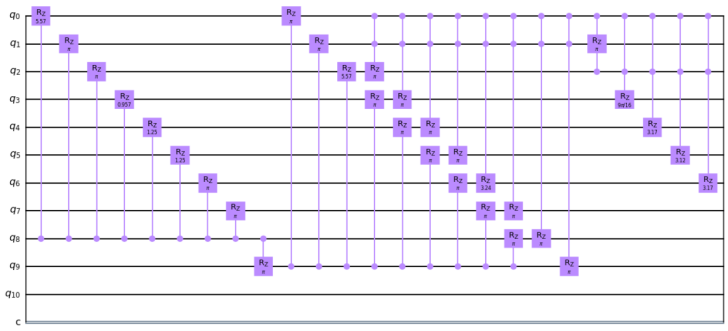
The smallest unit to be updated is each element in the two-dimensional $k_w\ (j = \mathbb{R}_{\leq 2^n - 1})$.

$\Delta k_{w,t}$ represents the partial derivative in $k_{w,t}$. $\overrightarrow{s_t}$ and $s_{t,j}$ represent the sign vector and its jth element. This element can be $+1$ or -1. $c(>0)$ represents the magnitude of the perturbation common to all elements. $\overrightarrow{s_t}$ was generated using Bernoulli random numbers based on probability distributions. $s_{t,j}$ is i.i.d. (independent homodistribution) for t, j, its distribution is symmetric around 0 and its magnitude is uniformly finite. Since $L\left(\overrightarrow{k_{w,t}} + c\overrightarrow{s_t}\right) - L\left(\overrightarrow{k_{w,t}} - c\overrightarrow{s_t}\right)$ is constant for all js in (Eq.1.12), 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 making foolish partial differentiations.

The expected value of the loss function L varies with the amount of data used; it should be changed each time so that the $\eta\overrightarrow{\Delta k_{w,t}}$ of (Eq.1.11) and the $c\overrightarrow{s_t}$ of (Eq.1.12) are both about $\frac{\pi}{256}$. 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, I only need to calculate the value of L twice.

3. Results

Although this research was realized in theory and source code (Fig.6), the environment was inadequate for machine learning using 10 qubits, and there were problems with computer processing power to obtain even the data that could be shown as a result.



(Fig.6 Just a few of the automatically generated quantum circuits)

4. Discussion

The reason I normalized the grayscale values and applied them to the exponential function in increments of $\frac{\pi}{256}$ is that there was no other way to clearly distinguish black and white, and I could have applied $\frac{\pi}{128}$ to the mapping between $0 \sim 2\pi$, but then the colors shown by phases 0 (pure white) and $\frac{255\pi}{128}$ (pure black) would be completely different, but when assigned to the exponential function, the values would be similar.

Under normal circumstances, partial differentiation should be done as in the following equation.

$$L'\left(k_{w,s,t}\right) = \left(\frac{\partial L\left(k_{w,s,t}\right)}{\partial i_{s,t,0}}, \frac{\partial L\left(k_{w,s,t}\right)}{\partial i_{s,t,1}}, \cdots, \frac{\partial L\left(k_{w,s,t}\right)}{\partial i_{s,t,2^N-1}}\right)^{\mathrm{T}} \quad (1.13)$$

But with this method, function L must be performed 2^n+1 times per $\overrightarrow{k_{w,t}}$. Performing function L takes time, and the exponentially increasing number of calculations must be avoided. If the number of weight layers and size and number of images are increased, the number of operations in the cost function increases accordingly, and since this is machine learning, this quantum circuit is run hundreds, thousands, or tens of thousands of times. Therefore, I adopted (Eq.1.12). This way, we only need to calculate the loss function twice to turn the quantum circuit once.

Continuing from the previous paper, I aimed to develop an algorithm that maximizes the potential of the quantum computer, but at this point, it became difficult to simulate on a classical computer, and I ended up talking about the theory and source code.

After all, the observation probability of the qubits entangled in the H-gate is freely manipulated by multiple $MCRZ$ gates, and although it may be possible to reduce the number of gates by using $MCRY$ gates, the theoretical accuracy will not change because the input, weighting, and firing will be the same as this.

5. Acknowledgements

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

6. 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).