Quantum Machine Learning with Phase Manipulation

Keiichiro Hayashi

December 23, 2023

Abstract

Inspired by classical convolutional neural networks, I designed a quantum circuit that performs image input and convolution using $\hat{C}R_z$-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) 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) Non-linearity 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. I show its potential as a method for processing multidimensional information in a general-purpose manner, not only for quantum data but also for classical data.

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

1 Introduction

The motivation behind this research stems from the author's profound interest in medical image processing. Having personally experienced the challenges associated with a brain disease, I envisioned a system capable of integrated analysis of the human body, offering a comprehensive diagnosis as a futuristic medical technology. While technologies like CT and MRI provide valuable insights into the internal structures of the body, the current approach heavily relies on the subjective judgment of physicians. In contemporary clinical medicine, there is a growing recognition of the need to augment traditional diagnosis with informatics-based approaches, as advocated by some medical scientists. Envisioning a system that could outperform the accuracy of human diagnosis by providing a thorough analysis of image data, including CT and MRI, or even information before processing into images, the author recognized the potential of algorithms utilizing quantum computation. To achieve the goal of automatic diagnosis based on extensive information processing—beyond the capabilities of current batch processing—it becomes imperative to establish a novel information processing method. This method should transcend the limitations of classical computation and find support in the realm of quantum computation. Quantum transcendence is crucial for handling the overwhelming volume of information and processing tasks involved in medical image analysis, facilitating the realization of an advanced and efficient automatic diagnostic system.

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 optimizing 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 applying this technology to specific problems" have not been sufficiently examined. 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 usable 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.

2.1 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.

$$H\left|0\right\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1\\ 0\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\ 1\end{pmatrix} \tag{1}$$

When this is observed in the Z-basis, the probability of $|0\rangle$ and $|1\rangle$ being observed is 0.5 respectively. In general, given $\begin{pmatrix}\alpha\\ \beta\end{pmatrix}$ in standardized α and β, the probability of observing $|0\rangle$ is $|\alpha|^2$ and the probability of observing $|1\rangle$ is $|\beta|^2$. Inspired by this, Eq.1 can be expressed as

$$H\left|x_k\right\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\ e^{i\pi x_k}\end{pmatrix} \tag{2}$$

Introducing $R_z(\theta)$-gate represented by

$$R_z(\theta) \coloneqq e^{-i\theta Z/2} = \begin{pmatrix}e^{-i\frac{\theta}{2}} & 0\\ 0 & e^{i\frac{\theta}{2}}\end{pmatrix} \tag{3}$$

Therefore, the state of Eq.2 is represented by

$$R_z(\theta)\left|\psi\right\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}e^{-i\frac{\theta}{2}} & 0\\ 0 & e^{i(\pi x_k+\frac{\theta}{2})}\end{pmatrix} \tag{4}$$

$|\psi\rangle$ is the state of Eq.2. The input and processing of image information can be done by $C^{\otimes n}R_z$, where $C^{\otimes n}$ is the number of n control bits, by manipulating the value of this θ. For details, see Sections 2.2 and 2.3.

Even if I observe Eq.4 as it is, the observed probabilities of $|0\rangle$ and $|1\rangle$ are still 0.5, 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 $\begin{pmatrix}x_0 & x_1 & \dots & x_{N-1}\end{pmatrix}$ to vector $\begin{pmatrix}y_0 & y_1 & \dots & y_{N-1}\end{pmatrix}$ can be defined as

$$y_k = \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1} x_j w^{jk} \tag{5}$$

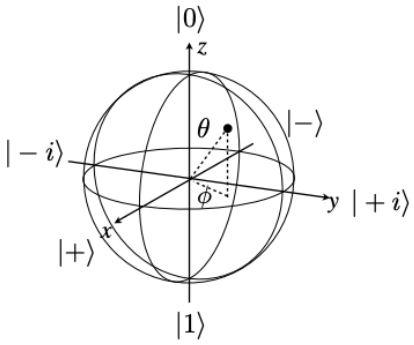


Figure 1: Bloch sphere

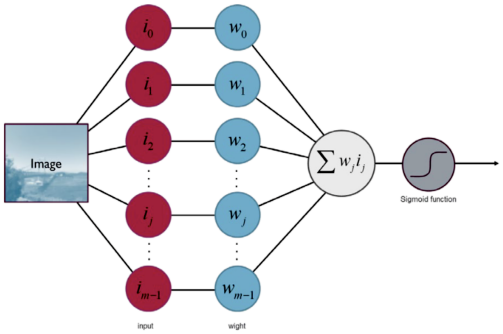


Figure 2: Schematic of a classical neural network.

Array $\boldsymbol{x}$ is standardized as $\sum |x_j|^2 = 1$, and $w = e^{2i\pi/N}$,$N = 2^n$. As you substitute Eq.5, you can find that the quantum Fourier transform maps quantum state $|x\rangle \coloneqq \sum x_j|j\rangle$ to quantum state $|y\rangle \coloneqq \sum y_k|k\rangle$ according to the same correspondence between $|x\rangle$ and $|y\rangle$, where $|i\rangle$ is a shorthand for the quantum state $|i_1 \dots i_n\rangle$ corresponding to the binary representation $i_1 \dots i_n (im = 0, 1)$ of integer i; for example, $|2\rangle = |0 \dots 010\rangle, |7\rangle = |0 \dots 0111\rangle$.

$$|y\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \sum_{j=0}^{N-1} x_j w^{jk} |k\rangle = \sum_{j=0}^{N-1} x_j \left(\frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} w^{jk} |k\rangle \right) \tag{6}$$

$$|j\rangle \overset{QFT}{\longmapsto} \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} w^{jk} |k\rangle \tag{7}$$

Thus, for the quantum Fourier transform, we only need to find a quantum circuit U that performs Eq. 7. As a unitary matrix, it is expressed as

$$U_{QFT} = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} w^{jk} |k\rangle \langle j| \tag{8}$$

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 $|0\rangle$ and $|1\rangle$ of the Z-basis into the states $|+\rangle$ and $|-\rangle$ of the X-basis. Similarly, every qubit in the computational basis has a state in the 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 $|0\rangle$ and $|1\rangle$ 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.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.

2.2 The development of an algorithm that automatically performs input for any image size.

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[11][8][12]. Let $\boldsymbol{i}$ be the vector to which the input information is mapped and $\boldsymbol{w}$ be the vector of weights to multiply it by.

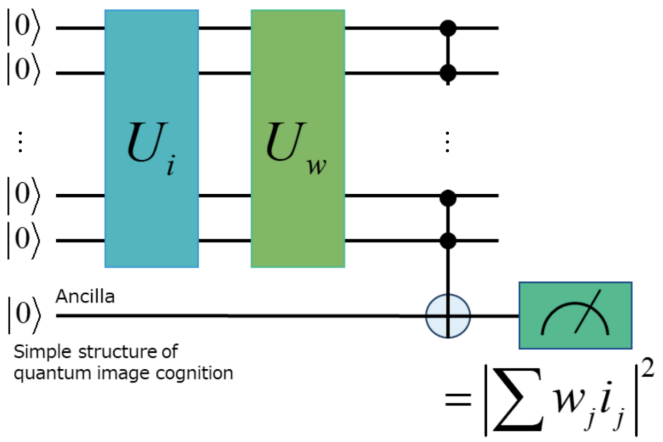


Figure 3: Schematic of a quantum neural network circuit.

$$\boldsymbol{i} = \begin{pmatrix} i_0 \\ i_1 \\ \vdots \\ i_{N-1} \end{pmatrix}, \boldsymbol{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{N-1} \end{pmatrix} \tag{9}$$

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

$$|\psi_i\rangle \coloneqq \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} i_j |j\rangle, |\psi_w\rangle \coloneqq \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} w_j |j\rangle \tag{10}$$

Quantum bits are superpositions of states in which qubits are superpositions of states in which $|0\rangle$ and $|1\rangle$ are probabilistically observed respectively, and $|\psi_i\rangle, |\psi_w\rangle$ represent 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 $|0\rangle^{\otimes n}$ to $|\psi_i\rangle$, the initial states. U_w is a unitary gate that multiplies the weights represented by $|\psi_w\rangle$. Finally, the firing is left to the inversion of the Ancilla qubit by the $C^{\otimes n}X$-gate.[11][9][10][7]

$$U_i |0\rangle^{\otimes n} = |\psi_i\rangle \tag{11}$$

$$U_w |\psi_w\rangle = |1\rangle^{\otimes n} = |N-1\rangle \tag{12}$$

When U_w applies the weight $|\psi_w\rangle$; if the quantum state immediately before U_w was $|\psi_w\rangle$, all n qubits after U_w will be $|1\rangle$. The $C^{\otimes n}X$ gate inverts the ancilla qubit, and the measured value will be 1. When U_w is applied to $|\psi_i\rangle$ corresponding to the input information,

$$|\phi_{i,w}\rangle \coloneqq U_w |\psi_i\rangle = \sum_{j=0}^{N-1} c_j |j\rangle \tag{13}$$

Using the definitions in Eq.12 and Eq.13, the inner product of the two quantum states is

$$\langle \psi_w | \psi_i \rangle = \langle \psi_w | U_w^\dagger U_w | \psi_i \rangle = \langle N-1 | \phi_{i,w} \rangle = c_{N-1} \tag{14}$$

$|c_j|^2$ represents the probability that the quantum state of the n qubits is $|j\rangle$ just before the $C^{\otimes n}X$-gate is applied. Therefore, $|c_{N-1}|^2$ indicates the probability that the quantum state is $|11\ldots11\rangle$, 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 $\theta = \pi \frac{g}{256}$ for the

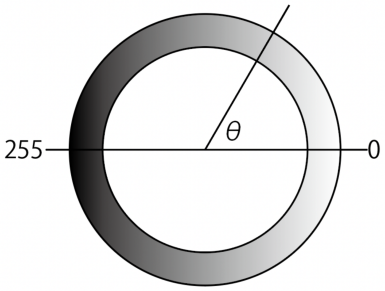


Figure 4: Correspondence between phase and gray-scale

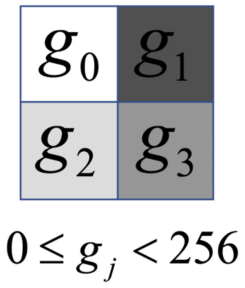


Figure 5: image of 2×2 squares

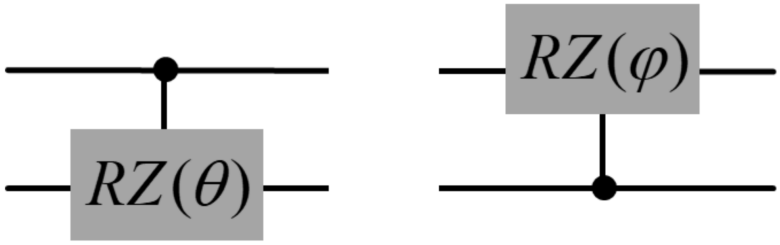


Figure 6: $\hat{C}R_z$-gates

grayscale $g(0 < g \leq 256)$. Given an image with $n \times n$ pixels, the number of qubits to store this image is $\lceil 2 \log_2 n \rceil$, where $\lceil x \rceil = \min\{n \in \mathbb{Z} \mid x \leq n\}$. The normalized grayscale is stored in n squared θ, and the image is input based on them. For padding, see Section 2.3. 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 gate $\hat{C}R_z$ is introduced, it performs as

$$\hat{C}_l R_{zk}(\theta) |j\rangle = \begin{cases} e^{i(\frac{\pi}{2}-\frac{\theta}{2})} |j\rangle & (\bigcap_1^n j_i \geq l_i \cap j_k = 0) \\ e^{i(\frac{\pi}{2}+\frac{\theta}{2})} |j\rangle & (\bigcap_1^n j_i \geq l_i \cap j_k = 1) \\ |j\rangle & otherwise \end{cases} \tag{15}$$

Here, if l is expressed as an n-digit binary number and the m-th qubit is 1, then the m-th qubit is a control bit. Also, the k-th qubit is the target bit. $R_z(\theta)$ is applied to the target bit when the control bits are all $|1\rangle$, but the sign in the middle changes depending on whether the k-th qubit is 0 or 1(Eq.3). As long as this qubit is on the xy-plane, the probability that this k-th qubit is $|0\rangle$,$|1\rangle$ is 0.5 each. Here, using the two gates as shown in Fig.6 for the two-qubit after applying an H-gate to each, we obtain

$$|\psi\rangle = \frac{1}{2}\left(|00\rangle + e^{i(\frac{\pi}{2}-\frac{\phi}{2})} |01\rangle + e^{i(\frac{\pi}{2}-\frac{\theta}{2})} |10\rangle + e^{i(\frac{\pi}{2}+\frac{\phi}{2}+\frac{\theta}{2})} |11\rangle\right) \tag{16}$$

The parameters of the R_z-gate are determined by designing a linear system of equations from the matrix and solving them so that the exponent part of e is in phase $\boldsymbol{i}$ corresponding to the input. In this case, the example in Eq.16 cannot solve the simultaneous equation because there are two parameters for three input values. Still, according to Eq.17, the method can be used because the type of $\hat{C}R_z$-gate is larger than the number of input values for three qubits or more.

$$\sum_{k=2}^{n} \binom{n}{k} k > 2^n \quad (n \geq 3) \tag{17}$$

The left-hand side is the sum of $k-1$ control bits and a target bit choices with respect to $2 \leq k \leq n$. See Appendix A for the proof of this formula.

I have completed an algorithm to automatically design quantum circuits for the contents of U_i and U_w, solving the simultaneous equation.

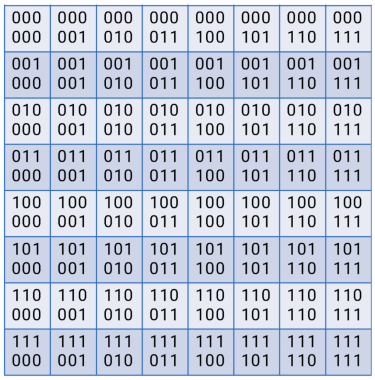


Figure 7: Correspondence between qubit status and the image information.

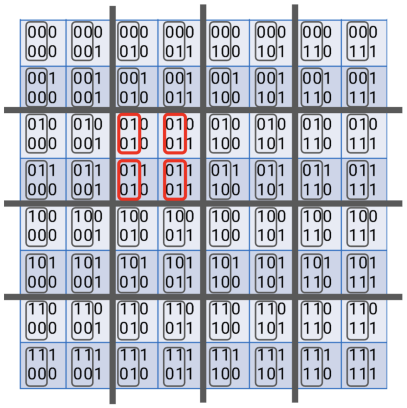
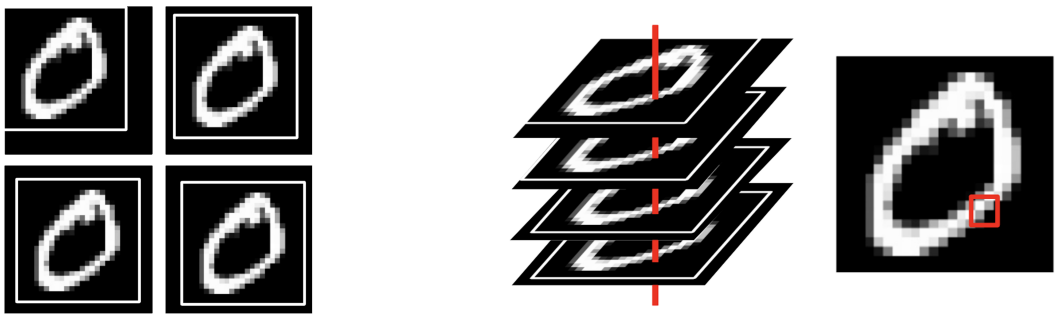


Figure 8: Convolutional filters cannot cross the grating.

2.3 Barriers unique to my method at the time of convolution are overcome by arranging the padding method.

MNIST images were preprocessed into 32×32 images by adding pixels around the image (padding) since MNIST images are 28×28 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 six qubits). The proposed quantum circuit is in the form shown in Fig.10. 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. Suppose we apply the X-gate and the $\hat{C}_{54}R_z(\theta)$-gate to these four qubits, the R_z-gate will act on the newly introduced target bit only when the 0th, 1st, 3rd, 4th qubits are initially 0; $54_{(10)} = 110110_{(2)}$. In this case, some newly introduced qubits are also control bits. By using a different $\hat{C}R_z(\theta)$ each time when ignoring the ten 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 $|00\ldots00\rangle, |00\ldots01\rangle, \ldots, |11\ldots11\rangle$ 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 correlation of the convolution's control bits retains the positional information to a certain extent, 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, in classical neural



(a) the same image with different paddings

(b) Selection of overlapping areas is possible.

Figure 9: How to choose a square at will

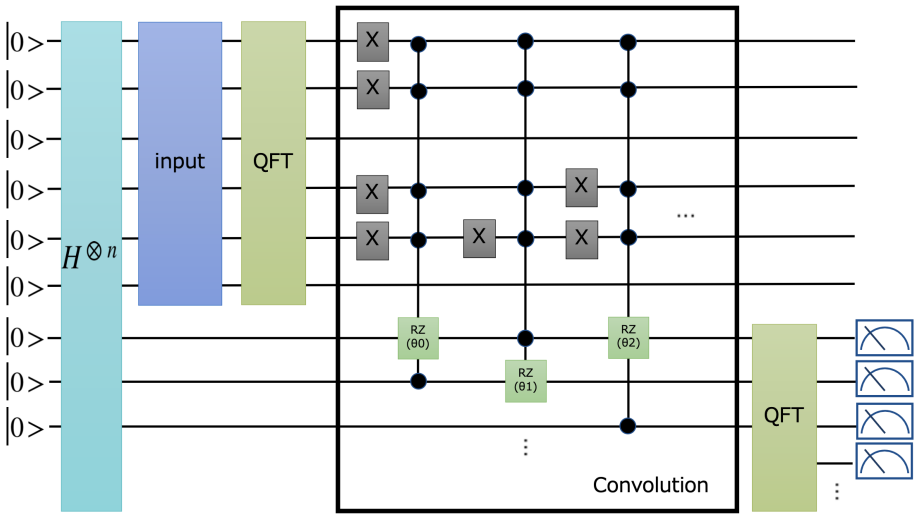


Figure 10: Flow of Input, Convolution, Pooling, and Observation.

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. In convolution, operations are performed by selecting squares with a common quantum state. For example, as shown in Fig.8, if we choose squares with four quantum states in common out of 6, we can select four squares in turn. However, this does not allow us to choose the four squares that straddle the grid. It was thought that the relationship between pixels separated by a grid must be completely ignored. Still, 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 28×28 MNIST image into a 32×32 image by adding two black cells around the perimeter of the original 28×28 MNIST image, but several different images can be created by varying the cells to be added to the top, bottom, left, and right (Fig.9(a)). When images with different padding are overlaid, as shown in Fig.9(b), squares in the same position viewed from above have a common quantum state. This allows us to select multiple squares more freely. If the image is a square with m×m, there are $\left(2^{\lceil \log_2 m \rceil} - m\right)^2$ 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.8. Here, the control and target bits of the $\hat{C}R_z(\theta)$-gates 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.10, 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 Section 2.6 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.

2.4 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's define the expected value $E(i, \boldsymbol{k})$ of the output result obtained from some training data i and a set of parameters $\boldsymbol{k}$.

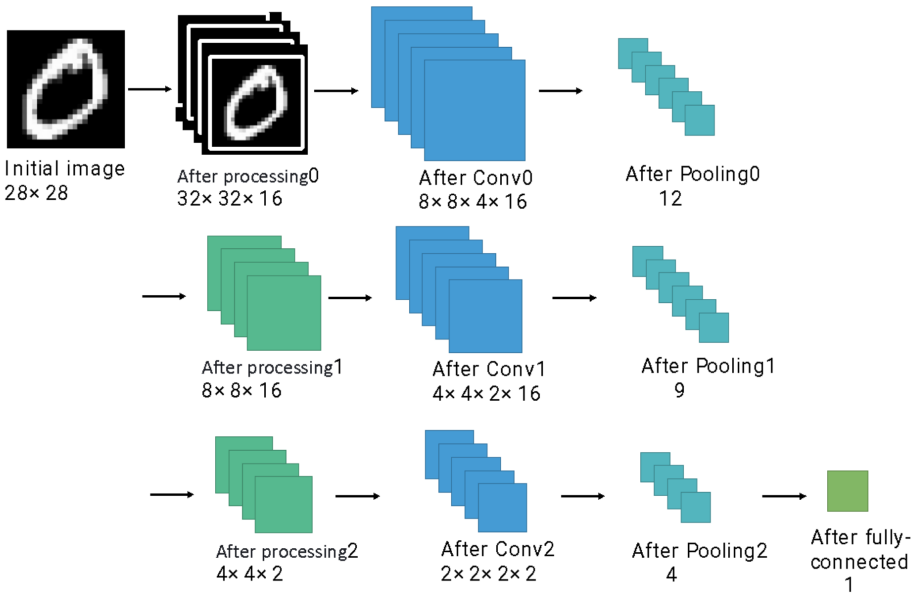


Figure 11: Circuit of 1st try

$$E(i, \boldsymbol{k}) = \frac{1}{\kappa} \sum_{m=0}^{\kappa} E_m(i, \boldsymbol{k}) \tag{18}$$

κ is the number of shots, and in this study, $\kappa = 1024$. Also, for each m images of 0 or 1, the loss function L is designed as

$$L(\boldsymbol{k}) = \frac{1}{2m} \sum_{j=0}^{m-1} \left[\{E\left(i_{0,j}, \boldsymbol{k}\right)\}^2 + \{1 - E\left(i_{1,j}, \boldsymbol{k}\right)\}^2 \right] \tag{19}$$

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

$$\boldsymbol{k}_{t+1} = \boldsymbol{k}_t - \eta \Delta \boldsymbol{k}_t \tag{20}$$

$$\Delta k_{t,j} = \frac{L\left(\boldsymbol{k}_t + c\boldsymbol{s}_t\right)}{2cs_{t,j}} = \frac{L\left(\boldsymbol{k}_t + c\boldsymbol{s}_t\right)}{2c} s_{t,j} \tag{21}$$

Here, the simultaneous perturbation method is used to calculate the loss function 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[6][5]. $c(> 0)$ represents the magnitude of the perturbation common to all elements. Simply updating the parameters would result in the calculation of

$$\nabla L(\boldsymbol{k}_t) = \left(\frac{\partial L(\boldsymbol{k}_t)}{\partial k_{t,0}}, \frac{\partial L(\boldsymbol{k}_t)}{\partial k_{t,1}}, \ldots, \frac{\partial L(\boldsymbol{k}_t)}{\partial k_{t,1291}} \right)^\top \tag{22}$$

Using the simultaneous perturbation method dramatically reduces the number of times the loss function is calculated. See Appendix B for proof that the expected value of a derivative by the simultaneous perturbation method is the value of the partial derivative in each variable.

2.5 It is a hybrid type of algorithm in which the parameters are updated by classical computation.

As shown in Section 2.4, 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

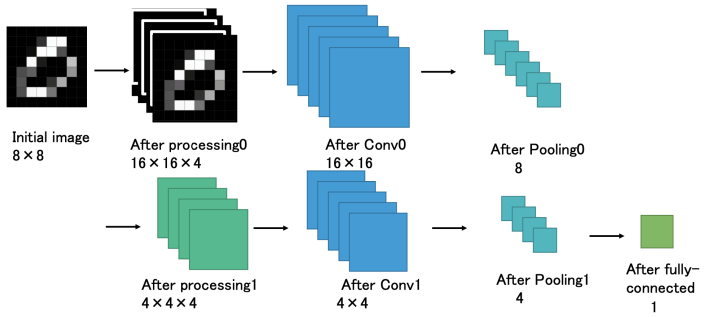


Figure 12: Circuit of 2nd try

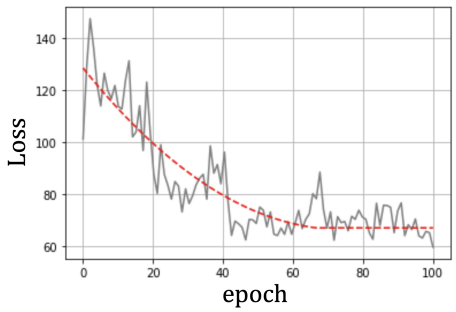


Figure 13: 2nd try loss function

with reduced computational complexity and can be said to be a hybrid type of classical and quantum algorithm.

2.6 Non-linearity is ensured by sandwiching multiple observations.

In the learning step in Fig.11, linearity is avoided by observing after the Convolution and Pooling layers and re-inputting their values. This is because all quantum gates are linear operations. After all, 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.[1][2][4]

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

In considering my research methodology, I made 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 an extended 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 Fig.12, compressing the 28×28 MNIST image to 8×8,

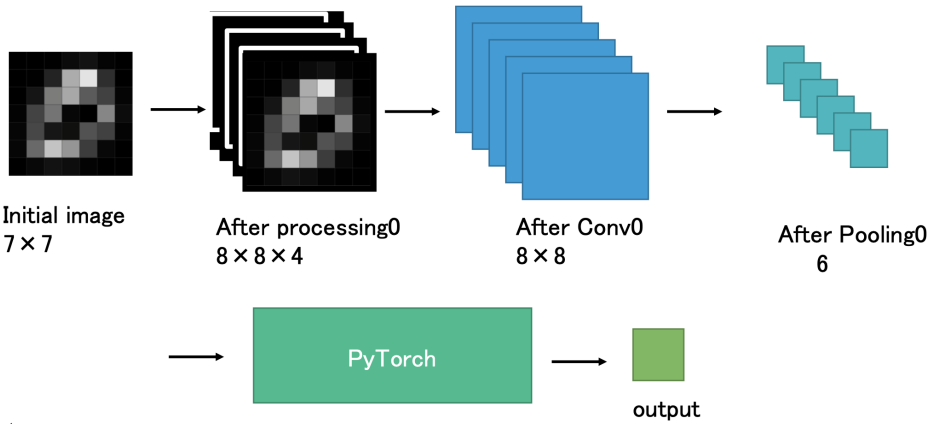


Figure 14: Final circuit

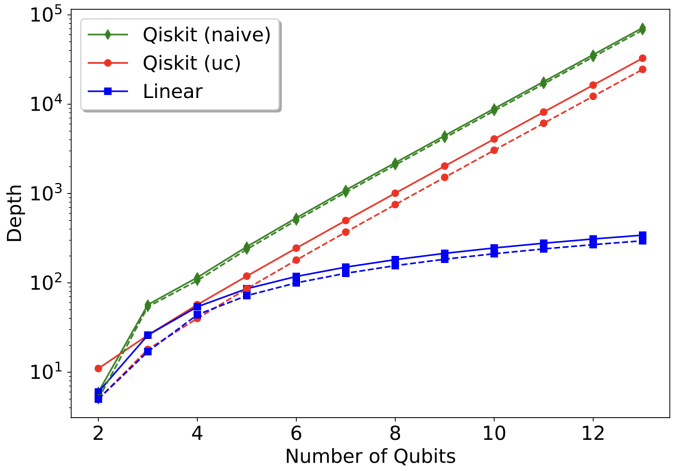


Figure 15: Depth employed in Qiskit and when using the Linear method (Da silva, A.j. & Park, D.K. (2022). Linear- Depth Quantum Circuits for Multi-Qubit Controlled Gates. cited from Phys. Rev. A 106, 042602[3])

padding to create four types of 16×16 images per original image, mapping to 10 qubits, using 18 qubits in the Convolution layer stage, and finally eight qubits in the observations and output. These are then formatted as 4×4 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 Fig.13. The numbers converged smoothly, but the learning accuracy was 53%. This was thought to be because 3/4 of the 16×16 data were "blanks" that were covered with padding. Therefore, I further tried the learning step compressing a 28×28 MNIST image to 7×7, padding to create four eight ×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 I was right to focus on the problem of many "blank" areas, but shaping the data into 4x4 data before entering the second round may not have been appropriate. Therefore, I attempted the learning step shown in Fig.14, in which the neural network for the 2-classification problem gradually makes the features more salient at each layer. If the operation by the quantum algorithm in the first round is correct, then the classification should be possible when 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%.

4 Discussion

Before discussing my 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 Fig.15, if the number of qubits involved in qubits is n, the depth of the qubits is of the order of $O(k^n)(2 < k < 3)$. On the other hand, the Linear method shown in Fig.15 allows decomposition on the order of $O(n^2)$[3]. One measure in evaluating many quantum algorithms, including my 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 $\hat{C}R_z$-gates used, as discussed above. In my method, one $\hat{C}R_z$-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, the classical operation is performed approximately 2^{n+m} times while the quantum operation is performed 2^n times. Also, under these conditions, the depth of a single quantum gate is 2^n. Therefore, the computational cost, which is the product of these two, is shown in Table

1 if the cost of the minimum gate operation is q for the quantum operation and c for the classical operation. Let m increase in proportion to n, and let $m = \gamma n$, using a constant γ of roughly 0.5 or less.

	Quantum	Classical
Depth	n^2	-
Calculation	2^n	$2^{n(1+\gamma)}$
Min Cost	q	c
Gross	$2^n n^2 q$	$2^{n(1+\gamma)} c$

Table 1: Calculation cost per filter

This is the cost required in the process of generating one new channel with one filter. The superiority of quantum computation can be confirmed by determining the ratio of this computation to the given amount of data (Fig.16). Quantum operations can be reduced to a ratio of $(\log N)^2$ when the amount of data is $N(= 2^n)$. I am currently developing a unitary computation method that can perform countless $\hat{C}R_z$ processes in a batch. In this case, the amount of computation "cost" can be reduced to $O((\log N)^4)(= O(n^4))$.

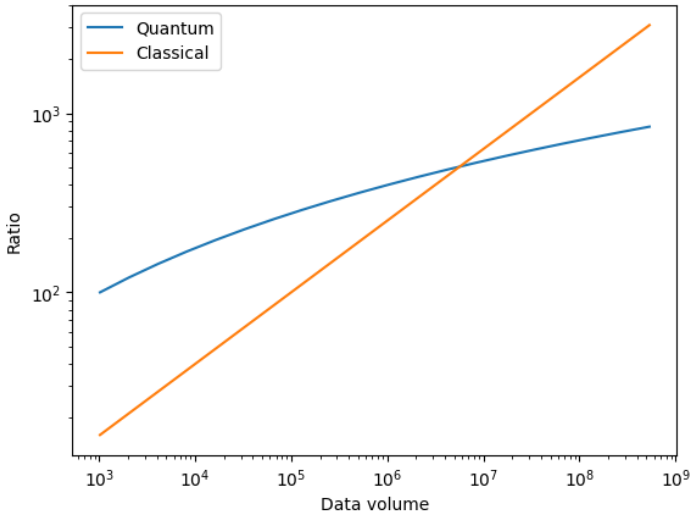


Figure 16: Ratio of computation cost to data volume. Calculated with $\gamma = 0.4$ and $q = c = 1$.

The number of fine qubits used simultaneously to solve MNIST was 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 my 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 was able to reduce the number of qubits to 14. 28×28 MNIST images were compressed to 7×7 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 an outstanding 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. 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 is the one that can make the best use of my method. The policy of hybrid quantum machine learning should be to reduce the huge amount of information that cannot be processed by conventional machine learning to a size that can be learned by conventional machine learning, while retaining the minimum number of features necessary for learning.

5 Conclusion

In this study, machine learning using a one-layer quantum algorithm increased 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 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.

The way things are going at the moment, it is necessary to perform the convolution process as many times as the amount of input data. I mentioned a prior work that reduces the depth of the $\hat{C}R_z$-gates handled in that convolution. In my role, I would like to further reduce the cost by devising a unitary operation that summarizes the entire convolution process.

6 Acknowledgments

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

References

[1] Marcello Benedetti. *Quantum-classical generative models for machine learning*. PhD thesis, UCL (University College London), 2019.

[2] Avinash Chalumuri, Raghavendra Kune, and BS Manoj. Training an artificial neural network using qubits as artificial neurons: a quantum computing approach. *Procedia Computer Science*, 171:568–575, 2020.

[3] Adenilton J Da Silva and Daniel K Park. Linear-depth quantum circuits for multiqubit controlled gates. *Physical Review A*, 106(4):042602, 2022.

[4] YaoChong Li, Ri-Gui Zhou, RuQing Xu, Jia Luo, and WenWen Hu. A quantum deep convolutional neural network for image recognition. *Quantum Science and Technology*, 5(4):044003, 2020.

[5] Hiroshi Maeda. What is simultaneous perturbation optimization? (ai qa classroom). *Systems/Control/Information*, 51(12):563–564, 2007.

[6] Hiroshi Maeda. Simultaneous perturbation optimization method and its applications. *Systems/Control/Information*, 52(2):47–53, 2008.

[7] Stefano Mangini, Francesco Tacchino, Dario Gerace, Daniele Bajoni, and Chiara Macchiavello. Quantum computing models for artificial neural networks. *Europhysics Letters*, 134(1):10002, 2021.

[8] Stefano Mangini, Francesco Tacchino, Dario Gerace, Chiara Macchiavello, and Daniele Bajoni. Quantum computing model of an artificial neuron with continuously valued input data. *Machine Learning: Science and Technology*, 1(4):045008, 2020.

[9] F. Tacchino, P. Kl. Barkoutsos, C. Macchiavello, D. Gerace, I. Tavernelli, and D. Bajoni. Title of the paper. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 130–136. IEEE, 2020.

[10] Francesco Tacchino, Panagiotis Barkoutsos, Chiara Macchiavello, Ivano Tavernelli, Dario Gerace, and Daniele Bajoni. Quantum implementation of an artificial feed-forward neural network. *Quantum Science and Technology*, 5(4):044010, 2020.

[11] Francesco Tacchino, Chiara Macchiavello, Dario Gerace, and Daniele Bajoni. An artificial neuron implemented on an actual quantum processor. *npj Quantum Information*, 5(1):26, 2019.

[12] Francesco Tacchino, Stefano Mangini, Panagiotis Kl Barkoutsos, Chiara Macchiavello, Dario Gerace, Ivano Tavernelli, and Daniele Bajoni. Variational learning for quantum artificial neural networks. *IEEE Transactions on Quantum Engineering*, 2:1–10, 2021.

A Proof that the number of CRz-gate types is greater than the number of input parameters

Theorem: For all natural numbers n greater than 2, the inequality $\sum_{k=2}^{n} \binom{n}{k} k > 2^n$ holds.

Proof:

1. **Base case**: $n = 3$: For $\sum_{k=2}^{3} \binom{3}{k} k = 9$, $2^3 = 8$, and it is trivially true that $9 > 8$. Therefore, the inequality holds for $n = 3$.

2. **Inductive hypothesis**: Assume that for some natural number m, $\sum_{k=2}^{m} \binom{m}{k} k > 2^m$ holds.

3. **Inductive steps**:

$$(a+b)^m = \sum_{k=0}^{m} \binom{m}{k} a^k b^{m-k} \tag{23}$$

Substitute a=1 and b=1.

$$(1+1)^m = 2^m = \sum_{k=0}^{m} \binom{m}{k} = \sum_{k=0}^{m} \frac{m!}{(m-k)!k!} \tag{24}$$

Next, take the difference on the left side when $n = m+1$ and $n = m$.

$$\begin{aligned}
\sum_{k=2}^{m+1} \binom{m+1}{k} k - \sum_{k=2}^{m} \binom{m}{k} k &= (m+1)\binom{m+1}{m+1} + \sum_{k=2}^{m} \left(\binom{m+1}{k} - \binom{m}{k} \right) k \\
&= (m+1) + \sum_{k=2}^{m} \left(\frac{(m+1)!}{(m+1-k)!k!} - \frac{m!}{(m-k)!k!} \right) k \\
&= (m+1) + \sum_{k=2}^{m} \frac{m!}{(m-k)!(k-1)!} \left(\frac{m+1}{m+1-k} - 1 \right) \\
&= (m+1) + \sum_{k=2}^{m} \frac{m!}{(m+1-k)!(k-1)!} k \\
&= (m+1) + \sum_{k=1}^{m-1} \frac{m!}{(m-k)!k!} (k+1) \\
&> 2 + \sum_{k=1}^{m-1} \frac{m!}{(m-k)!k!} = \sum_{k=0}^{m} \frac{m!}{(m-k)!k!} = 2^m
\end{aligned} \tag{25}$$

Considering this and $\sum_{k=2}^{m} \binom{m}{k} k > 2^m$,

$$\sum_{k=2}^{m+1} \binom{m+1}{k} k > 2^m + \sum_{k=2}^{m} \binom{m}{k} k > 2^m + 2^m = 2^{m+1} \tag{26}$$

Therefore, $\sum_{k=2}^{m+1} \binom{m+1}{k} k > 2^{m+1}$ holds.

By the principle of mathematical induction, the inequality $\sum_{k=2}^{n} \binom{n}{k} k > 2^n$ is proven to be true for all natural numbers n greater than 2. The number of $\hat{C}R_z$-gate types is greater than the number of input parameters

B Proof that the expected value of the derivative using the simultaneous perturbation method is the value of the partial derivative in each variable

Expand $f(\boldsymbol{x}) + c\boldsymbol{s}$ by Taylor expansion.

$$f(\boldsymbol{x} + c\boldsymbol{s}) = f(\boldsymbol{x}) + c\sum_{i=0}^{n-1} \frac{\partial f(\boldsymbol{x})}{\partial x_i} s_i + \frac{c^2}{2} \sum_{i,j=0}^{n-1} \frac{\partial^2 f(\boldsymbol{x})}{\partial x_i \partial x_j} s_i s_j + \dots \tag{27}$$

Next, define the nabla and Hesse matrices.

$$\nabla = \begin{pmatrix} \frac{\partial}{\partial x_0} \\ \frac{\partial}{\partial x_1} \\ \vdots \\ \frac{\partial}{\partial x_{n-1}} \end{pmatrix} \tag{28}$$

$$H = \nabla^2 f(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial^2 f(\boldsymbol{x})}{\partial x_0 \partial x_0} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_0 \partial x_1} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_0 \partial x_{n-1}} \\ \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_0} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_{n-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\boldsymbol{x})}{\partial x_{n-1} \partial x_0} & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_{n-1} \partial x_1} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_{n-1} \partial x_{n-1}} \end{pmatrix} \tag{29}$$

Approximate the Taylor expansion up to the second-order terms.

$$f(\boldsymbol{x} + c\boldsymbol{s}) \simeq f(\boldsymbol{x}) + c(\nabla f, \boldsymbol{s}) + \frac{c^2}{2}(\boldsymbol{s}, H\boldsymbol{s}) \tag{30}$$

Using this in the expected value formula,

$$\frac{f(\boldsymbol{x} + c\boldsymbol{s}) - f(\boldsymbol{x})}{cs_i} \simeq (\nabla f(\boldsymbol{x}), \boldsymbol{s}) s_i + \frac{c}{2}(\boldsymbol{s}, H\boldsymbol{s}) s_i \tag{31}$$

$$\therefore E\left(\frac{f(\boldsymbol{x} + c\boldsymbol{s}) - f(\boldsymbol{x})}{cs_i}\right) \simeq \frac{\partial f(\boldsymbol{x})}{\partial x_i} \tag{32}$$

Using this value, the parameters can be updated as follows.

$$x_i' = x_i - \eta \frac{\partial f(\boldsymbol{x})}{\partial x_i} \tag{33}$$