Chapter XIII

Qubit Neural Network: Its Performance and Applications

Nobuyuki Matsui
University of Hyogo, Japan

Haruhiko Nishimura
University of Hyogo, Japan

Teijiro Isokawa
University of Hyogo, Japan

ABSTRACT

Recently, quantum neural networks have been explored as one of the candidates for improving the computational efficiency of neural networks. In this chapter, after giving a brief review of quantum computing, the authors introduce our qubit neural network, which is a multi-layered neural network composed of quantum bit neurons. In this description, it is indispensable to use the complex-valued representation, which is based on the concept of quantum bit (qubit). By means of the simulations in solving the parity check problems as a benchmark examination, we show that the computational power of the qubit neural network is superior to that of the conventional complex-valued and real-valued neural networks. Furthermore, the authors explore its applications such as image processing and pattern recognition. Thus they clarify that this model outperforms the conventional neural networks.

INTRODUCTION

Since Shor (1994) proposed a way of factorizing large integers in polynomial time by using a quantum computing algorithm, the study of quantum information science, including quantum communication, quantum cryptography, quantum computer and so on, has been intensified (Nielsen & Chuang, 2000). Shor’s proposal has not only proved itself to be a milestone in quantum computing, but also created a novel research paradigm of neural computing, i.e., quantum neural computing (Kak, 1995). Since then, various quantum neural computing models have been proposed for the improvement of the computational ability of neural networks so as to expand their applications.
Qubit Neural Network

(Peruš, 1996, 2004; Behrman, Nash, Steck, Chandrashekar, & Skinner, 2000; Narayanan, & Menneer, 2000; Ezhov, Nifanova, & Ventura, 2000; Matsui, Takai, & Nishimura, 1998, 2000; Kouda, Matsui, & Nishimura, 2002, 2004; Kouda, Matsui, Nishimura, & Peper, 2005a, 2005b; Mori, Isokawa, Kouida, Matsui, & Nishimura, 2006; Rigui, Nan, & Quilin, 2006). In this chapter, we introduce a qubit neural network model that is a complex-valued multi-layered neural network composed of quantum bit neurons. We also clarify its learning performance numerically through the benchmark simulations by comparing it to that of the conventional neural networks. The quantum bit (hereafter qubit) neuron model was one that we proposed for the first time, inspired by quantum computation and quantum circuit (see the reference, Matsui, Takai, & Nisimura, 1998 in Japanese, 2000 in English) and we also proved that our qubit neural network model (hereafter Qubit NN) is more excellent in learning ability than the conventional real-valued neural network model through solving the image compression problem (Kouda, Matsui, & Nishimura, 2002) and the control problem of inverted pendulum (Kouda, Matsui, Nishimura, & Peper, 2005b). We indicated that these results could be ascribed to the effects of quantum superposition and probabilistic interpretation in the way of applying quantum computing to neural network, in addition to the complex number representation. In the formulation of our model, complex numbers play an essential role, as a qubit is based on the concept of quantum mechanics. Here, to clarify these quantum effects, we show the characteristic features of Qubit NN are superior to those of the old-fashioned conventional complex-valued and/or real-valued neural networks by means of the simulations in solving the parity check problems and the function identification problem as a benchmark examination (see, Kouda, Matsui, Nishimura, & Peper, 2005a). Lastly, we add to the results of the new application examples: the well-known iris data classification and the night vision image processing. Thus we conclude that Qubit NN model outperforms Classical NNs. Here, we call the conventional neural networks Classical NNs according to the traditional comparison: Classical physics versus Quantum physics.

BACKGROUND

Before entering into the discussion of Qubit NN, we introduce its study background. Neural computing and quantum computing, both considered as promising innovative computation models, have attracted much interest from researchers. Neural computing is based on the intellectual and soft information processing of the brain that fuses and harmonizes the analog computation with the digital computation. It has been studied not only in modeling brain functions but also in solving various practical problems in industry such as data classification, pattern recognition, motion control, image processing and so on. Recently, in order to make it a reliable technology of information processing, as well as to expand its applications, Complex-valued, Quaternion-valued and Clifford-algebra neural networks have been explored (Hirose, 2003). We hypothesize that their computational ability can become higher than Real-valued one. This is not a clear problem, but Siegelmann (1999) shows that the computational ability of a neural network depends on the type of numbers utilized as its weight parameters: Integer-valued, Rational-valued and Real-valued type networks are computationally equivalent to finite automata, Turing machine and non-uniform computational models, respectively. Then, we speculate that hyper complex-valued NNs are beyond the computational ability that Turing model has achieved. It is also widely known that the computation based on quantum mechanics has higher computational ability than the Turing model. As for quantum computing, Feynman (1982) raised a question about the feasibility of the computing architectures based on quantum mechanics, and Deutsch (1985) started the study of quantum computing by proposing a computer model that operates according to principles of quantum mechanics—namely, the universal quantum Turing machine was proposed. Deutsch and Jozsa (1992) put forward a prototypical quantum algorithm for the first time, and proved their algorithm was able to speed up the computational processing ability by means of the quantum parallelism. Consequently, the study of quantum computer began to be more accelerated aiming at making clear its computational properties. Bernstein and Vazirani (1993, 1997) showed the existence of universal quantum Turing machines capable of simulating other quantum Turing machines in polynomial time, and Yao (1993) proved that quantum Turing machines are computationally equivalent to quantum circuits. Then, Shor (1994) showed how a solution of a large integer factorization problem in polynomial time is possible in principle by utilizing an algorithm operating on a quantum computer. His algorithm has attracted widespread interest because of the security of modern cryptography. Next, Grover (1996, 1997) discovered a fast quantum algorithm for database search. His
Qubit Neural Network algorithm was a significant breakthrough for wide applications. At the present, no quantum algorithms have been discovered beyond the capacity of these three quantum algorithms. On the other hand, neural computing, though successful in modeling information processing in the brain above mentioned, has faced problems in practice, since the massively parallel characteristics of most models in this framework are not suitable for the simulation in a reasonable time on classical computers. It is indeed a problem that today’s computer algorithms based on classical physics fail to fully describe the basic nature of neural networks, which encompasses the unification of distributed neuron operation and integrated coherent behavior. To accomplish such characteristics in a computing system, therefore, we need a new computational principle.

Kak (1995) and Peruš (1996) drew attention to the similarity between the description of neuron states and that of quantum mechanical states and discussed a notion of quantum neural computing that is in accordance with quantum theory. Similar early works have been explored by Menneer et.al. (1995), Behrman et.al. (1996), and Ventura et.al. (1997). In these approaches, they discussed the possibilities of a quantum neural network inspired by the many-worlds interpretation (Menneer et.al.), a quantum dot neural network (Behrman et.al.) and a neuron model with its weight vectors as the quantum wave functions (Ventura et.al.). These models were theoretically interesting and stimulating. However, to make progress with regard to practical enhancement of neural networks’ computational power, we have to construct a neural network capable of industrial applications. We, therefore, have attempted to establish a correspondence between neural networks and quantum circuits by proposing a qubit neuron model (Matsui et.al., 1998). In our neuron model, called Qubit neuron model, the states of neurons and their interactions with other neurons are based on the laws of quantum physics.

Now we aim to develop practical quantum neural networks and expect to establish these schemes as a step for a novel quantum algorithm.

QUANTUM COMPUTING AND NEURON MODEL

Qubit neuron and its neural network are neural models based on the concept of ‘quantum bit’ called qubit as the counterpart of the classical concept of ‘bit’. We start this section with a brief review of quantum computing to explain the neural network inspired by the quantum concept.

Qubit and Its Representation

The qubit is a two-state quantum system. It is typically realized by an atom with its ground state and one of its excited states, an electronic spin with its up state and down one, or a photon with its two polarization states. These two states of a qubit are represented by the computational basis vectors $|0\rangle$ and $|1\rangle$ in a two-dimensional Hilbert space. An arbitrary qubit state $|\phi\rangle$ maintains a coherent superposition of the basis states $|0\rangle$ and $|1\rangle$ according to the expression:

$$|\phi\rangle = c_0 |0\rangle + c_1 |1\rangle; \quad |c_0|^2 + |c_1|^2 = 1,$$

where $c_0$ and $c_1$ are complex numbers called the probability amplitudes. When one observes the $|\phi\rangle$, this qubit state $|\phi\rangle$ collapses into either the $|0\rangle$ state with the probability $|c_0|^2$, or the $|1\rangle$ state with the probability $|c_1|^2$.

These complex-valued probability amplitudes have four real numbers; one of these is fixed by the normalization condition. Then, the qubit state (1) can be written by

$$|\phi\rangle = e^{i\lambda} (\cos \theta |0\rangle + e^{i\chi} \sin \theta |1\rangle),$$

where $\lambda$, $\chi$, and $\theta$ are real-valued parameters. The global phase parameter $\lambda$ usually lacks its importance and consequently the state of a qubit can be determined by the two phase parameters $\chi$ and $\theta$ without loss of generality.
That is

\[ |\phi\rangle = \cos \theta |0\rangle + e^{ix} \cdot \sin \theta |1\rangle \]  

(3)

Thus, the qubit can store the value 0 and 1 in parallel so that it carries much richer information than the classical bit.

Quantum Gates

To create a novel neuron model inspired by the concept of qubit, we need to incorporate the concept of quantum logical gates into our neuron model. In quantum computing, the logical operations are realized by reversible, unitary transformations on qubit states (Nielsen & Chuang, 2000). Here, in order to explain the idea of our qubit neuron model, we introduce the symbols for the logical universal operations, i.e., the single-qubit rotation gate \( U_\theta \) shown in Figure 1 and the two-qubit controlled NOT gate \( U_{CNOT} \) shown in Figure 2.

First we sketch the single-qubit rotation gate \( U_\theta \). We can represent the computational basis vectors \(|0\rangle\) and \(|1\rangle\) as vectors in a two-dimensional Hilbert space as follows:

\[
|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.
\]  

(4)

In such a case we have the representation of \(|\phi\rangle = \cos \theta |0\rangle + \sin \theta |1\rangle\):

\[
|\phi\rangle = \begin{pmatrix} \cos \theta \\ \sin \theta \end{pmatrix}.
\]

and the matrix representation of \( U_\theta \) operation can be described by Eq.(5).

\[
U_\theta = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}.
\]  

(5)

This gate varies the phase of the probability amplitudes from \( \theta \) into \( \theta + \theta \) as follows:

\[
|\phi\rangle = U_\theta |\phi\rangle = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} \cos \theta \\ \sin \theta \end{pmatrix} = \begin{pmatrix} \cos \theta \cdot \cos \theta - \sin \theta \cdot \sin \theta \\ \sin \theta \cdot \cos \theta + \cos \theta \cdot \sin \theta \end{pmatrix} = \begin{pmatrix} \cos(\theta, + \theta) \\ \sin(\theta, + \theta) \end{pmatrix}.
\]  

(6)

Next we outline the two-qubit controlled NOT gate \( U_{CNOT} \). From Figure 2 we see the \( U_{CNOT} \) gate operates on two-qubit states. These are states of the form \( |a\rangle \otimes |b\rangle \), which can be written more simply as \( |ab\rangle \), as a tensor product of two vectors \( |a\rangle \) and \( |b\rangle \). It is usual to represent these states as follows:

\[
|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad |01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad |10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad |11\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.
\]  

(7)
This standard representation is one of several important bases in quantum computing. When the $U_{\text{CNOT}}$ gate works on these two-qubit states as vectors (7) in a four-dimensional Hilbert space, the matrix representation of the $U_{\text{CNOT}}$ operation can be described by

$$
U_{\text{CNOT}} = \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0
\end{pmatrix}.
$$

This controlled NOT gate has a resemblance to a XOR logic gate that has $|a\rangle$ and $|b\rangle$ inputs. As shown in Figure 2, this gate operation regards the $|a\rangle$ as the control and the $|b\rangle$ as the target. If the control qubit is $|0\rangle$, then nothing happens to the target one. If the control qubit is $|1\rangle$, then the NOT matrix is applied to the target one. That is, $|a, b\rangle \rightarrow |a, b \oplus a\rangle$. The symbol $\oplus$ indicates the XOR operation.

An arbitrary quantum logical gate or quantum circuit is able to be constructed by these universal gates. For example, the three-bit quantum circuit, which is a minimum logical operation circuit constructed by four rotation gates and three controlled NOT gates, is shown in Figure 3. Its operator matrix $U_c$ is as follows:

$$
U_c = \begin{pmatrix}
U_1 & U_2 & 0 \\
0 & U_3 & U_4
\end{pmatrix},
$$

$$
U_i = \begin{pmatrix}
\begin{pmatrix}
\cos \Theta_i \\
\sin \Theta_i
\end{pmatrix} & -\begin{pmatrix}
\sin \Theta_i \\
\cos \Theta_i
\end{pmatrix}
\end{pmatrix},
\Theta_1 = \theta_1 + \theta_2 + \theta_3 + \theta_4,
\Theta_3 = \theta_1 - \theta_2 - \theta_3 + \theta_4,
\Theta_2 = \frac{\pi}{2} - (\theta_1 + \theta_2 - \theta_3 - \theta_4),
\Theta_4 = \frac{\pi}{2} - (\theta_1 - \theta_2 + \theta_3 - \theta_4),
\end{array}
\right.
$$

$i \in \{1, 2, 3, 4\}$.  

329
This quantum circuit can realize several two-input logic functions by setting the $\theta_1$ to $\theta_4$ into the appropriate values. In the case of $\theta_1 = \theta_2 = \pi/8$, $\theta_3 = \theta_4 = -\pi/8$ for example, it becomes AND or NAND logic gate depending on the state of $|c\rangle$. When the $|c\rangle$ is $|0\rangle$, the output $|c'\rangle$ in this quantum circuit results in the output for AND gate with two inputs $|a\rangle$ and $|b\rangle$. The circuit works as the NAND gate when the $|c\rangle$ is $|1\rangle$. The NAND gate can construct any logical functions by itself on conventional classical computing. Thus, this quantum circuit can construct any logical functions.

**Figure 3. Three-bit quantum circuit**

\[
\begin{align*}
|a\rangle & \quad \bullet \\
|b\rangle & \quad \bullet \\
|c\rangle & \quad \theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4 \\
|c'\rangle &
\end{align*}
\]

Complex-Valued Description of Qubit Neuron State

Now we proceed to describe our qubit neuron model. Our qubit neuron model is a neuron model inspired by the quantum logic gate functions: its neuron states are connected to qubit states, and its transitions between neuron states are based on the operations derived from the two quantum logic gates. To make the connection between the neuron states and the qubit states, we assume that the state of a firing neuron is defined as a qubit basis state $|1\rangle$, the state of a non-firing neuron is defined as a qubit basis state $|0\rangle$, and the state of an arbitrary qubit neuron is the coherent superposition of the two:

\[
|\psi\rangle = \alpha |0\rangle + \beta |1\rangle, \quad |\alpha|^2 + |\beta|^2 = 1
\]  

(10)

corresponding to Eq.(1). In this qubit-like description, the ratio of firing and non-firing states is represented by the probability amplitudes $\alpha$ and $\beta$. These amplitudes are generally complex-valued. We, however, consider the following state as shown in Figure 4, which is a special case of Eq. (3) with $\chi = 0$:

\[
|\psi\rangle = \cos \theta |0\rangle + \sin \theta |1\rangle
\]  

(11)

as a qubit neuron state in order to give the complex-valued representation of the functions of the single-qubit rotation gate $U_{\theta}$ and the two-qubit controlled NOT gate $U_{\text{CNOT}}$. To this end, in addition, we introduce the following expression instead of Eq.(11):

\[
f(\theta) = \cos \theta + i \sin \theta = e^{i\theta},
\]  

(12)

where $i$ is the imaginary unit $\sqrt{-1}$ and $\theta$ is defined as the quantum phase. The complex-valued description (12) can express the corresponding functions to the operations of the rotation gate and the controlled NOT gate.

a) Phase rotation operation as a counterpart of $U_{\theta}$

As shown in Figure 5, the rotation gate is a phase shifting gate that transforms the phase of qubit neuron state. Since the qubit neuron state is represented by Eq. (12), the following relation holds:
Qubit Neural Network

Figure 4. Qubit-like description of neuron state

\[
|1\rangle: \text{Firing} \\
|0\rangle: \text{Non-Firing}
\]

Figure 5. Phase rotation operation in a qubit neuron state as a counterpart of \(U_{\theta}\)

\[f(\theta_1 + \theta_2) = f(\theta_1) \cdot f(\theta_2),\]  \hspace{1cm} (13)

b) Phase reverse operation as a counterpart of \(U_{\text{CNOT}}\)
This operation is defined with respect to the controlled input parameter \(\gamma\) corresponding to the control qubit as follows:

\[f\left(\frac{\pi}{2} \gamma - \theta\right) = \begin{cases} 
\cos \theta - i \sin \theta & (\gamma = 0) \\
\sin \theta + i \cos \theta & (\gamma = 1)
\end{cases},\]  \hspace{1cm} (14)

where \(\gamma = 1\) corresponds to the control qubit \(-|1\rangle\), i.e., the role reversal-rotation of basic states, and \(\gamma = 0\) corresponds to the control qubit \(-|0\rangle\), i.e., the role conservable-rotation of basic states. In the case of \(\gamma = 0\), the phase of the probability amplitude of quantum state \(|1\rangle\) is reversed as shown in Figure 6. However, its observed probability is invariant so that we are able to regard this case as the role conservable-rotation of basic states.

Qubit Neuron Model vs. Classical Neuron Model

Now we are ready to formulate the qubit neuron model by using the above complex-valued description of qubit neuron state. Before we discuss our qubit neuron model, we summarize the well-known classical neuron models to clarify the differences between the qubit and the classical ones.
Qubit Neural Network

Classical Neuron Models

The well known real-valued conventional neuron model is expressed by the equations:

\[ u = \sum_{m=1}^{M} w_m \cdot x_m - v \]  \hspace{1cm} (15)

\[ y = (1 + e^{-u})^{-1}. \]  \hspace{1cm} (16)

Here, \( u \) is the internal state of a neuron \( y \), \( x_m \) is the neuron state of the \( m \)-th neuron as one of \( M \) inputs to \( y \), \( w_m \) and \( v \) are the weight connection between \( x_m \) to \( y \) and the threshold value, respectively. These neuron parameters are real numbers.

The complex-valued neuron model is like a real-valued one except that the neuron parameters are extended to the complex numbers \( W \) as the weight, \( X \) and \( Y \) as the neuron state, \( V \) as the threshold and so on (Nitta, 1997) giving rise to the following equations that correspond to Eqs. (15) and (16), respectively:

\[ U = \sum_{l=1}^{L} W_l \cdot X_l - V \]  \hspace{1cm} (17)

\[ Y = (1 + e^{-\text{Re}(U)})^{-1} + i \cdot (1 + e^{-\text{Im}(U)})^{-1}. \]  \hspace{1cm} (18)

Qubit Neuron Model

We have to observe the transition of the state of the qubit neuron in terms of the unitary transformation as the qubit concept is used for the description of the neuron state. A certain unitary transformation can be realized by the combination of the single-qubit rotation gate \( U_\theta \) and the two-qubit controlled NOT gate \( U_{\text{CNOT}} \). It is natural, therefore, to construct a computing model whose transition of the neuron state is performed according to these two quantum logic gates. We try to incorporate their concept to the framework of the above-mentioned classical neuron model Eqs. (15) and (16) (or Eqs. (17) and (18)) using the descriptions of Eqs. (13) and (14) that correspond to \( U_\theta \) and \( U_{\text{CNOT}} \) operations respectively. In this case, the output state of qubit neuron has to be also described by Eq.(12). To implement this scheme, we assume the following: we replace the classical neuron weight parameter \( w_i \) (or \( W_i \)) with the phase rotation operation \( f(\theta_i) \) as a counterpart of \( U_\theta \) and install the phase reverse operation as a counterpart of \( U_{\text{CNOT}} \) instead of using the non-linear function in Eq.(16) (or Eq.(18)), and then we consider the following equations:
Qubit Neural Network

\[ u = \sum_l f(\theta_l) \cdot x_l - f(\lambda) = \sum_l f(\theta_l) \cdot f(y_l) - f(\lambda), \]  
(19)

\[ y = \frac{\pi}{2} g(\delta) - \arg(u), \]  
(20)

\[ z = f(y). \]  
(21)

Here, \( u \) is the internal state of a qubit neuron \( z \). \( x_l \) is the qubit neuron state of the \( l \)-th neuron as one of inputs from \( L \) other qubit neurons to \( z \). \( \theta_l \) and \( \lambda \) are the phases regarded as the weight connecting \( x_l \) to \( z \) and the threshold value, respectively. \( y \) and \( y_l \) are the quantum phases of \( z \) and \( x_l \), respectively. \( f \) is the same function as defined in Eq. (12) and \( g \) is the sigmoid function with the range (0,1) given by

\[ g(\delta) = \frac{1}{1 + e^{\delta}}. \]  
(22)

Two kinds of parameters exist in this neuron model: phase parameters in the form of weight connection \( \theta_l \) and threshold \( \lambda \), and the reversal parameter \( \delta \) in Eq.(22). The phase parameters correspond to the phase of the rotation gate, and the reversal parameter to the controlled NOT gate. By substituting \( y = g(\delta) \) in Eq.(14), we obtain a generalized approximate reversal representation operating as the controlled NOT gate, the basic logic gate in quantum computing. We assume that Eq.(19) expresses the state \( u \) of a neuron in the usual way, i.e., as the weighted sum of the states of the inputs minus a threshold. Eq.(20) adjusts the output quantum phase \( y \) more roughly than Eq.(19). In Eq.(20), \( \arg(u) \) means the argument of complex number \( u \), and is implemented by \( \arctan(\text{Im}(u)/\text{Re}(u)) \). By doing so, we can calculate the transition of these phases. We, thus, describe the transition of the qubit neuron state according to the phase transition, while a typical classical model does not use these phases for describing the state transition. That is, the multiplication of weights \( f(\theta_l) \) to the inputs \( x_l = f(y_l) \) results in the rotation of the neuron state based on the rotation gate. This we call is Qubit neuron model. A theoretical ground for adopting the classical model has not been sought for; we just adopt its formalism capable of applying to our model. Therefore the qubit neuron is not a reversible model on precise bases of quantum computing; this is an analogical model of quantum state.

Figs.7 and 8 give the qubit neuron model and its quantum gate diagram respectively. These diagrams may be useful for readers to understand our qubit neuron model.

Figure 7. Qubit neuron model
QUBIT NEURAL NETWORK

Now we proceed to construct the three-layered neural network employing qubit neurons called “qubit neural network: Qubit NN”.

Network Structure

As shown in Figure 9, Qubit NN has the three sets of neuron elements: \( \{ I_l \} (l=1,2,\ldots,L) \), \( \{ H_m \} (m=1,2,\ldots,M) \) and \( \{ O_n \} (n=1,2,\ldots,N) \), whereby the variables \( I, H \) and \( O \) indicate the Input, Hidden, and Output layers, and \( L, M \) and \( N \) are the numbers of neurons in the input, hidden and output layers, respectively. We denote this structure of the three-layered NN by the numbers of \( L-M-N \).

When input data (denoted by \( \mathit{input} \)) is fed into the network, the input layer consisting of the neurons in \( \{ I_l \} \) converts input values in the range \([0, 1]\) into quantum states with phase values in the range \([0, \pi/2]\). The output \( z^{I_l} \) of input neuron \( I_l \) becomes the input to the hidden layer:

\[
z^{I_l} = f(\frac{\pi}{2} \cdot \mathit{input}_l).
\]

The hidden and output layers with neurons from the sets \( \{ H_m \} \) and \( \{ O_n \} \), respectively, obey Eqs.(19), (20) and (21). We obtain the output to the network, denoted by \( \mathit{output}_o \), by calculating the probability for the basic state \( |1\rangle \) in the \( n \)-th neuron state \( z^{O_n} \) in the output layer:

\[
\mathit{output}_o = |\text{Im}(z^{O_n})|^2
\]

This \( \mathit{output} \) definition is based on the probabilistic interpretation in the way of applying quantum computing to neural network.

Quantum Modified Back Propagation Learning

Next, we define a quantum version of the well-known Back Propagation algorithm (BP algorithm) in order to incorporate learning process in Qubit NN. The gradient-descent method, often used in the BP algorithm, is employed as the learning rule. This rule is expressed by the following equations:
Learning Performances: Qubit NN vs. Classical NNs

We now investigate the information processing ability of Qubit NN using benchmark learning problems for training our network and compare its performance with that of classical neural networks, i.e., Real-valued NN and Complex-valued NN. Here, Real-valued NN is a network constructed with the well-known real-valued conventional neuron model as shown in the above-mentioned Eqs. (15) and (16). Similarly, Complex-valued NN is a network with the complex-valued neuron model described by Eqs. (17) and (18).

Both the classical NNs use the BP learning algorithm. The neuron parameters $w_{m}, \theta$ of Real-valued NN and $W_{r}, V$ of Complex-valued NN are updated in the same way as in Eqs. (25), (26) and (27). As a footnote, in Complex-valued NN, the real parts, $\text{Re}(W_{r})$, $\text{Re}(V)$ and the imaginary parts, $\text{Im}(W_{r})$, $\text{Im}(V)$ are updated independently.
Learning Conditions

In all simulations, the sizes of the above NNs are chosen such that all networks have almost the same number of neural parameters, thus ensuring that all networks have nearly the same degrees of freedom, which is an important factor determining the learning ability. The numbers of neural parameters $NUM$ of Qubit NN and classical NNs can be determined as follows.

- Qubit NN: $NUM_{qubit} = LM + MN + 2M + 2N$ \hspace{1cm} (29)
- Real-valued NN: $NUM_{real} = LM + MN + M + N$ \hspace{1cm} (30)
- Complex-valued NN: $NUM_{complex} = 2(LM + MN + M + N)$ \hspace{1cm} (31)

The subscripts, $qubit$, $real$ and $complex$ mean Qubit NN, Real-valued NN and Complex-valued NN, respectively. In these simulations, we consider that a network succeeds in training when the value of the squared error function $E_{\text{total}}$ decreases to less than a certain bound $E_{\text{lower}}$ and it fails in training when it does not converge within a certain number of learning iterations, denoted by $L_{\text{upper}}$. In both cases the network finishes the training. Here, one learning iteration encompasses the process of updating the learning parameters as the result of feeding all input patterns into the network exactly once.

The simulations are conducted in a number of epochs and the results are averaged, resulting in the average number of learning iterations required for a NN to learn a certain training set. We define the success rate as the percentage of simulation sessions in which the NN succeeded in its training, i.e., in which the NN required less than $L_{\text{upper}}$ iterations. The success rates and the average numbers of learning iterations are indicators of the processing efficiencies, by which the performances of the different models can be compared.

The simulations in accordance to the above conditions are carried out on the so-called 4, 6-bit parity check problems and the general function identification problem, described in the following sections.

Parity Check Problems

First, we investigate the performances of Qubit NN and Classical NNs in learning the 4 bit parity check problem in detail. For this simulation, we use a 4-6-1 Qubit NN, i.e., a three layered Qubit NN that has 4 neurons in the input layer, 6 neurons in the hidden layer and 1 neuron in the output layer mentioned previously. According to Eq.(29), the 4-6-1 Qubit NN has 44 neural parameters to be trained. A 4-8-1 Real-valued NN with 49 neural parameters and a 2-6-1 Complex-valued NN with 50 neural parameters are used for comparing the efficiencies.

These simulation results show in Figs.10 a), b) and c). In these figures, we denote the network structure and neural parameters by $L-M-N: NUM$.

As in the case of the 4-bit parity check problem, for the simulations with Real-valued NN and Qubit NN we use 16 input patterns, i.e., all the 4-tuple patterns in the range \{0,0,0,0\} \sim \{1,1,1,1\}, and we use as target signals the scalars 0 and 1 obtained from XORing all elements of the 4-tuples inputs. In case of Complex-valued NN, the input patterns are the 2-tuple patterns in the range \{0+i0, 0+i0\} \sim \{1+i1, 1+i1\} and the target signals are again the scalars 0 and 1 obtained from XORing all elements of the inputs. It is important for the performance comparisons to determine the optimal learning rate of each NN, but this is a difficult problem when doing simulations on NN. To cope with this we define a finite set of possible learning rates and try all values in the simulations. This procedure is followed for the learning rates $\eta_q$, $\eta_r$, $\eta_c$ of Qubit NN, Real-valued NN and Complex-valued NN, respectively.

The number of simulation sessions is set to 100 epochs for each learning rate value tested. In each epoch the values of the network parameters are initialized to random values. For each learning rate value the simulation results are averaged, giving rise to one data point, plotted in a figure that shows the learning abilities of the neural networks in terms of the average numbers of required learning iterations on the vertical axis with respect to the success rate on the horizontal axis. From this axis definition, the nearer a dot in the figure is to the corner right below, the more efficient the corresponding network is. The initial values of the parameters are in the ranges $[-\pi,$
**Qubit Neural Network**

Figure 10. Learning performance in the 4-bit parity check problem

![Plot](image)

(a) Average number of learning iterations vs. success rate (Condition for success: $E_{\text{lower}} = 0.005$, $L_{\text{upper}} = 3000$)

(b) Dependence of optimal average number of learning iterations on the number of neural parameters in network

(c) Dependence of success rate, by which learning is accomplished within the optimal average number of iterations, on the number of neural parameters in network.

$p$] for Qubit NN, [-1, 1] for Real-valued NN and [-1+i, 1+i] for Complex-valued NN. $L_{\text{upper}}$ is set to 3000 and $E_{\text{lower}}$ is set to 0.005 in the 4-bit parity check problem. We observe from Figure 10(a) that the optimal average numbers of required iterations of the respective models are about 800 iterations at $\eta_q = 0.1$, about 1900 iterations at $\eta_r = 1.3$ and about 1380 iterations at $\eta_c = 1.3$. Of all the models, the dot of Qubit NN is the nearest to the corner right below. Figs. 10 b) and c) show the influence of the number of hidden layer neurons (or neural parameters) on the optimal average number of learning iterations and on the success rate, respectively. Learning is considered successful if $E_{\text{lower}} = 0.005$, $L_{\text{upper}} = 3000$. From these figures, we observe only Qubit NN accomplishes training with an average number of required iterations lower than 500. We also see that Qubit NN excels over Classical NNs in all cases with respect to the average number of required iterations. Furthermore, when the number of hidden layer neurons increases, the average number of required iterations of Qubit NN decreases more than that of Classical models.

Next, in Figure 11, we show the result for the 6-bit parity check problem, which is a more complicated problem. The input patterns for the 6-bit problem are established by following the same principles as in the 4-bit parity check problem. $L_{\text{upper}}$ is set to 10000 and $E_{\text{lower}}$ is set to 0.006. The results for Qubit NN are the nearest to the corner
Qubit Neural Network

Figure 11. Learning performance in the 6-bit parity check problem

\[ P(x) = \frac{\sin \pi x + \sin 2\pi x + 2.0}{4.0} \quad (0 \leq x < 2) \]
Qubit Neural Network

We use a 2-14-1 Qubit NN which has 72 neural parameters, a 2-18-1 Real-valued NN with 73 parameters and a 1-12-1 Complex-valued NN with 74 parameters. Similar to the previous parity check problem section, the networks have almost the same degrees of freedom. The input is set to \( \{ \text{input}_1, \text{input}_2 \} = \{ x, 0.0 \} \) in the range \( 0.0 \leq x \leq 1.0 \) and \( \{ \text{input}_1, \text{input}_2 \} = \{ 0.0, x-1.0 \} \) in the range \( 1.0 < x < 2.0 \) and the training target is \( P(x) \). In this simulation we adopt a step size of 0.1, giving rise to 21 data points. So the input patterns become \( \{ 0.0, 0.0 \} \sim \{ 1.0, 0.0 \} \) in \( 0.0 \leq x \leq 1.0 \) and \( \{ 0.0, 0.1 \} \sim \{ 0.0, 0.9 \} \) in \( 1.0 < x < 2.0 \).

The initial values of the neural parameters and the learning rates are in the same respective ranges as in the previous section. The value of \( E_{lower} \) is 0.01, and the value of \( L_{upper} \) is 10000.

The learning abilities of the respective networks are shown in Figure 12. From the figure it can be seen that the Qubit NN requires 2000 iterations to achieve a 100% success rate, while the Classical NN takes 4500 iterations and the complex NN takes 3100 iterations to achieve this.

Finally we check how well Qubit NN can make approximations on data that have not been used for training. The output of the network is plotted against the actual graph of \( P(x) \) in Figure 13. The squared mean error is 0.011, which is close to \( E_{lower} \), even though the data points were not part of the training data.

Quantum Effects in Learning Performances

In this section, we explore the reason why the proposed Qubit NN is more efficient than Classical NNs, by analyzing the results of the simulations on the 4-bit and 6-bit parity check problems. We clarify the following three points:

a. Does the superiority of Qubit NN consist in its being complex valued?
b. Does the superiority of Qubit NN consist in its state of quantum superposition?
c. Does the superiority of Qubit NN consist in its probabilistic interpretation?

First of all, we consider the point (a).

Figure 12. Learning performance in the identification of an arbitrary function

![Figure 12](image-url)
Qubit NN restricts its use of complex numbers in its states to those with polar radius “1” due to the state descriptions based on quantum superposition, whereas Complex-valued NN uses complex numbers in its states without this restriction. To check whether the better performance of Qubit NN was not due to its states being restricted to polar radius 1, we carried out simulations on Complex-valued NN whose neuron parameters also had restrictions with regard to the polar radius: in this case we used polar radii of the values 1, 5, 8, and 10. The best results of these simulations were obtained when the radius was 8 (see Figure 14). The number of neurons in the hidden layers of the radius-restricted complex-valued NN is more than twice that of the free-radius Complex-valued NN, to compensate for the halving of the number of free parameters in the former as compared to the latter. Even in the optimal radius 8, the results for the radius-restricted Complex-valued NN are inferior to the results obtained for Qubit NN.

Next, we proceed to investigate the point (b).

To clarify the advantages of the quantum description, i.e. the quantum superposition in Eq.(11), we evaluate Qubit NN from which the quantum superposition is removed. Figure 15 shows the results of simulations on

---

**Figure 13. Ability of the network to approximate function P(x)**

![Output value vs. Input value graph]

**Figure 14. Learning performance in the 4-bit parity check problem by a radius-restricted complex-valued NN, and Qubit NN (Point (a))**

![Average iterations vs. Success rate graph]

- ○ Radius-restricted Complex-valued NN (4–9–1;55, r=8.0)
- ● Qubit NN (4–7–1;51)

(Condition for success: \(E_{lower}=0.005, L_{upper}=3000\)
Qubit Neural Network

Figure 15. Quantum superposition effect in Learning performance (♦: Qubit NN, □: Qubit NN without quantum superposition)

![Diagram](image)

(a) The case of 4-bit parity check problem  (b) The case of 6-bit parity check problem

The models with and without quantum superposition: in this case, the concrete values of the phase of the qubit state on the hidden layer is $n\pi/2$ ($n=0,1,2,\ldots$). From Figure 15, we see the removal of superposition substantially downgraded the performance of the model in both the 4-bit and 6-bit cases. In the removal case of 6-bit parity check, we cannot complete the training successfully. The simulation conditions for the 4-bit parity check and 6-bit parity check problems are the same as in Figure 10 and Figure 11, respectively. Lastly, we discuss the point (c).

By changing the output $o_n = |\text{Im}(z_n^n)|^2$ into $|\text{Im}(z_n^n)|$ in Eq.(15), we realized a model without a probability interpretation in the quantum description. The results in the 4-bit and 6-bit cases are shown in Figure 16. We find, in the removal case of the 6-bit parity check, the success rate stays below 10%, even in the optimal case. From these figures, we see that the removal of the partial description or features from Qubit NN resulted in a decrease of its performance.

The above results thus indicate that the high learning efficiency of Qubit NN is due not to the restriction of the polar radius but to the description in terms of the quantum superposition and the probability interpretation inspired by quantum computation.

PRACTICAL APPLICATIONS

We have discussed the basic computational power of Qubit NN by solving benchmark problems in the previous section. In this section we investigate the solvable performance of Qubit NN in practical and knotty engineering problems. We have already found that this model outperforms to Real-valued NN in solving the image compression problem (Kouda, Matsui, & Nishimura, 2002) and the control problem of inverted pendulum (Kouda, Matsui, Nishimura, & Peper, 2005b) and confirmed its efficiency. Here, in order to confirm its solvable ability and expand its industrial applications we examine its information processing power in the well-known Iris data classification problem and the night vision processing.
Iris Data Classification Problem

Let's start our simulations with solving the Iris data classification problem. This problem is to classify the iris data set into three species. The iris data set consists of 4 measurements from 150 random iris flower samples (Blake, & Merz, 1998): 50 setosa (class1), 50 versicolor (class2), and 50 virginica (class3). As an example of the Iris data distribution shown in Figure 17, we see this distribution is uneven and very difficult to classify between the class2 and the class3.

First, we determine the optimal learning rates $\eta_q = 0.1$ and $\eta_r = 0.5$ at which performances of Qubit NN and Real-valued NN are best. The performances in this case are evaluated by the averaged $NCE$s for 30 trials with $L_{upper} = 1500$. Here, $NCE$ is a normalized cumulative error defined by

$$NCE = \frac{1}{L_{upper}} \sum_{l=1}^{L_{upper}} E(l), \quad E(l) = \sum_{p=1}^{P} E_p(l).$$

(33)

Where $E(l)$ is the total squared error at $l$-th learning iteration. In each learning iteration, the network accepts $P$ training samples and modifies its network parameters. The networks undergo $L_{upper}$ learning iterations, i.e., $l \leq L_{upper}$.

In order to evaluate the network performances in this Iris problem, we introduce the following normalized cumulative right answer $NCR$:

$$NCR = \frac{1}{L_{upper}} \sum_{l=1}^{L_{upper}} R_s(l), \quad R_s(l) = \frac{\text{the number of correct classification data}}{\text{the number of all test data}},$$

(34)

where $R_s(l)$ is the rate of success to the correct classification in the $l$-th learning iteration.

Figure 18 shows the transitions of the total squared error $E(l)$ and the correct classification rate $R_s(l)$. This
Qubit Neural Network

**Figure 17. Example of Iris data distribution**

**Figure 18. Qubit NN vs. Real-valued NN in performances on $R_A(l)$ and $E(l)$**
Qubit Neural Network

value becomes 1.0 when all the input data are correctly classified. From this figure, we see that Qubit NN has better classification ability than Real-valued NN, and Qubit NN can avoid the standstill states in learning while the learning in Real-valued NN may get stuck at one of local minima. We furthermore investigate the performances for Qubit NN and Real-valued NN from the viewpoint of generalization ability. It is conducted by applying 5-fold cross validation to the learning and evaluating processes. First we investigate the performances for evenly divided data set. In evenly divided condition, the data set for evaluating consists of 10 data for the setosa class (class1), 10 data for the versicolor class (class2) and 10 data for virginica (class3), and remaining 120 data (40 data for each class) are used for training. In other words, under this condition, the number of data for each class is set to be same. For 100 trials, we obtain $NCR=92.7\%$ for Qubit NN and $NCR=91.7\%$ for Real-valued NN. Next we conduct the performance evaluations under biased conditions. In these conditions, the data set is divided into two subsets for training and evaluating, but the number of data for one class may be different from others. We prepare three cases for dividing data set, as shown in Table 1. For example in the training data set in case 1, the number of data for class 1 is less than those for other classes and there are only class 1 data set for evaluation. For 500 trials for each case, we obtain $NCR=72.5\%$ for Qubit NN and 46.5% for Real-valued NN. Under biased conditions, the networks have difficulty for acquiring the characteristics from the training data set. But in these conditions, the performance of Qubit NN tops that of Real-valued NN, so we can conclude that our Qubit NN has better ability at acquiring the feature even from less data set, than Real-valued NN.

**Night Vision Processing**

The night vision processing is an image processing in which finer pixel values are estimated from gloomy ones as shown in Figure 19. In this work, we follow the scheme of the night vision processing with neural network that has been proposed in (Kusamichi et.al., 2004). The night vision processing has aimed to extract information from night scenes, in other words, to estimate the value of pixel in image taken in enough illumination from the value of pixel in image taken in night or twilight. Therefore, to train the network, the images from one scene taken in different illumination are necessary. For example, the image for input to the network is shown as Dark image in Figure 19 and the image for correspond to target is shown as Output image in Figure 19. These images has resolution of 160×120 (=19,200 pixels) and its intensity value is normalized from 0.0 to 1.0.

The network structure for this task is 9-n-1. The input data to the network consist of the value of the pixel located at $(x, y)$ and its eight direct neighboring pixels located at $(x+i,y+i)$ of the input image, where $(i,j) = \{(-1,-1),(0,-1),(1,-1),(-1,0),(1,0),(-1,1),(0,1),(1,1)\}$. The corresponding output is the pixel value located at $(x, y)$ of the target image. For the image with the resolution of 160×120, the number of the training pattern is 18,644 (=158×118) due to the incompleteness of input on the edge pixels. An outline of training the network is shown in Figure 20, where the pixels corresponding to each location are applied to the network in training.

The network structures of Qubit NN and Real-valued NN used in the following simulations are 9-5-1 and 9-6-1, and the neural parameters are 62 and 67, respectively. For the evaluation of images output from the networks, we introduce the PSNR (peak-signal to noise ratio) of these images with respect to the target image. The PSNR between two images, named $\Phi$ and $\Phi'$, consist of $X\times Y$ pixels is defined as

$$PSNR = 10 \log_{10} \left( \frac{1}{MSE} \right), \quad MSE = \frac{1}{X\times Y} \sum_{x} \sum_{y} \left( I(x,y) - I'(x,y) \right)^2,$$

(35)

where $I(x,y)$ and $I'(x,y)$ represent the intensities of the pixels at the location $(x,y)$ in the images $\Phi$ and $\Phi'$ respectively. First, as well as the iris data classification, we investigate the optimal learning rate. By calculating the averages of $NCE$ on 30 trials under $L_{opt}=500$, we find the optimal values in this application $\eta_q = 0.01$ for Qubit NN and $\eta_l = 1.0$ for Real-valued NN. In Figure 21, we show the averages of $E(l)$ at each optimal parameter.

By using these optimal values, we compare the output images of Qubit NN with those of Real-valued NN. In Figure 22, for example, we show typical images obtained at $l=500$ together with $E(500)$ values and their PSNR values. From Figs.21 and 22 we see that Qubit NN outperforms to Real-valued one.
Next, we consider the output images as shown in Figure 23. In this case, the output image from Qubit NN seems to be better than that from Real-valued NN, although we compare the best error value image for Real-valued NN with the average error one for Qubit NN. Figure 24 shows the distribution of pixel values for images after training and the relations between input and output images. We observe from Figs24 (a), (b) that values of pixels in output image for Qubit NN present linear behavior, but ones for Real-valued NN seem to reach saturation. This behavior of Real-valued NN is one of causes lowering the quality of image. In other words, it is insufficient in only error as the evaluation of the images. To explain this point, we import “margin” as follows:

\[
\text{margin} (x, y) = |V(x, y) - V(x + 1, y)| + |V(x, y) - V(x, y + 1)|
\]  

(36)

where, \( \text{margin}(x, y) \) and \( V(x,y) \) are the “margin” and the value of pixel located at \((x, y)\) in the image. Now, we create the histogram of this “margin” as shown in Figure 25. In Figure 25, the histogram for Qubit NN is.
Figure 21. Averages of total squared error $E(l)$

![Graph showing averages of total squared error $E(l)$ for Real-valued NN and Qubit NN over different learning iterations.]

Figure 22. Original image and output images from networks

(a) Target (Original) image  
(b) Image from Real-valued NN  
(c) Image from Qubit NN

$[E(500) = 1212, PSNR=14.28]$  
$[E(500) = 833, PSNR=15.91]$  

Figure 23. The image output from neural networks

(a) The image from Real-valued NN  
(b) The image from Qubit NN

$[E(500) = 297, PSNR=17.98]$  
$[E(500) = 919, PSNR=13.07]$
more similar to one for target than one for Real-valued NN. In order to evaluate quantitatively, we calculate the Kullback-Leibler distance (K-L distance) $D_{KL}$ by the thought that histogram is probability density function. The K-L distance is a natural distance measure from a “true” probability distribution $P$ to an arbitrary probability distribution $Q$, and widely used in information theory. For probability distributions $P$ and $Q$, the K-L distance of $Q$ from $P$ is defined as

$$D_{KL}(P, Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}.$$  

(37)

As the result, K-L distance for Real-valued NN $D_{KL,R}$ is 0.2457 and one for Qubit NN $D_{KL,Q}$ is 0.0904. We confirm the similar results by using the test image: Paper bag (provided from Seiko Instruments Inc.) in Figure 26. In this case, the error of image is $E=467$ for Real-valued NN, and $E=1009$ for Qubit NN and the higher quality of image and linear behavior are shown in only Qubit NN solution. In addition, K-L distance $D_{KL,R}$ and $D_{KL,Q}$ are 0.1156 and 0.0960 for the paper bag image respectively. That is, even if the error and the $PSNR$ of Real-valued are both better than those of Qubit one, we can obtain that Qubit NN has better image quality than Real-valued. This result is confirmed by using another image such as anime character. This is one of the advantageous features of Qubit NN.
Qubit Neural Network

CONCLUSION

In this chapter, we have introduced a scheme of quantum neural computing, namely, Qubit NN inspired by quantum computing. We compared its information processing performances with those of Classical NNs by means of solving the 4 and 6-bit parity check problems and the general function identification problem as a benchmark. In all simulations for the benchmark problems, we have observed that Qubit NN has more efficient processing abilities than Classical NNs and also revealed that the excellent computational ability of Qubit NN is not simply due to the complex-valued neural parameters or the introduction of the restricted polar radius, but to the characteristics inspired by quantum superposition and probabilistic interpretation. Furthermore, we have shown that Qubit NN outperforms Real-valued NN in the two practical applications. In the iris data classification by using non-uniformly data, there is a difference between Qubit NN and Real-valued NN in the classification performance. In the night vision processing, Qubit NN has a linear-like behavior between input and output images, which, we have observed, in turn contributes to enhancing the quality of output images. Thus we have made clear the advantage features of our Qubit NN in these numerical experiments.

FUTURE RESEARCH DIRECTIONS

From simulation experiments examined in our works, we conclude the efficiency of Qubit NN as probable. Of course, it is an open question that whether our Qubit NN outperforms Classical NNs in all information-processing problems or not, and that whether it is confirmed, without any mathematical proof, quantum computing descriptions enhance the computational power in neural information processing. At the present, to finalize this proof seems to be impossible. It is left for future studies. However, in order to validate the advantages of Qubit NN, it is worth investigating its computational ability in learning chaos systems and then improving its learning method. In addition, we will try to apply our qubit neuron model not only to multi-layered networks but also to the other networks such as a Hopfield-type and a cellular one, considering the other schemes of learning. We will challenge the integer prime-factorization problem using Qubit NN. At present, Classical NN method seems to be not sufficient for such a trial (Jansen, & Nakayama, 2005). This may be interesting and give a breakthrough for creating the scheme of neural quantum computing to establish these schemes as a step for a novel quantum algorithm provided that the comparison with Shor’s algorithm is possible. It is also attractive to incorporate Grover’s algorithm into the learning method of Qubit NN. By trying these, we expect to establish Qubit NN as a step for a novel quantum algorithm.

Other future work will focus on the use of Qubit NN in practical applications with more neural parameters to establish this method concretely. We, furthermore, will compare Qubit NN with Hypercomplex-valued NNs including Complex-valued NN in detail. It is true that Qubit NN method is an interesting method as a new complex-valued NN, and has a great potential for enlarging applications of complex-valued NN.
ACKNOWLEDGMENT

The contribution for this chapter is based on the studies in (Kouda, Matsui, Nishimura, & Peper, 2005a) (Mori, Isokawa, Kouda, Matsui, & Nishimura, 2006). Part of this research is supported by Grant-in-Aid for Scientific Research(C-16500135) Japan Society for the Promotion of Science.

REFERENCES


**ADDITIONAL READING**

**Qubit Neural Network**


