Neural Comput & Applic (2005) 14: 114–121
DOI 10.1007/s00521-004-0446-8

ORIGINAL ARTICLE

Noriaki Kouda · Nobuyuki Matsui · Haruhiko Nishimura
Ferdinand Peper

Qubit neural network and its learning efficiency

Received: 20 January 2004 / Accepted: 1 June 2004 / Published online: 13 January 2005

Abstract Neural networks have attracted much interest in the last two decades for their potential to realistically describe brain functions, but so far they have failed to provide models that can be simulated in a reasonable time on computers; rather they have been limited to toy models. Quantum computing is a possible candidate for improving the computational efficiency of neural networks. In this framework of quantum computing, the Qubit neuron model, proposed by Matsui and Nishimura, has shown a high efficiency in solving problems such as data compression. Simulations have shown that the Qubit model solves learning problems with significantly improved efficiency as compared to the classical model. In this paper, we confirm our previous results in further detail and investigate what contributes to the efficiency of our model through 4-bit and 6-bit parity check problems, which are known as basic benchmark tests. Our simulations suggest that the improved performance is due to the use of superposition of neural states and the use of probability interpretation in the observation of the output states of the model.

N. Kouda
Hyogo Prefectural Institute of Technology, 3-1-12 Yukihira-cho, Suma, Kobe Hyogo 654-0037, Japan
E-mail: kouda@hyogo-kg.go.jp

N. Matsui (✉)
Division of Computer Engineering,
Graduate School of Engineering, Himeji Institute of Technology, University of Hyogo, 2167 Shosha, Himeji, Hyogo 671-2201, Japan
E-mail: matsui@eng.u-hyogo.ac.jp
Tel.: +81-792-674993
Fax: +81-792-674993

H. Nishimura
Graduate School of Applied Informatics, University of Hyogo, 1-3-3 Higashikawasaki-cho, Chuo-ku, Kobe 650-0044, Japan
E-mail: haru@ai.u-hyogo.ac.jp

F. Peper
National Institute of Information and Communications Technology, 588-2 Iwaoka, Iwaoka-cho, Nishi-ku, Kobe 651-2492, Japan
E-mail: peper@nict.go.jp

Keywords Quantum computing · Neural network · Qubit · Learning · Parity check · Back-propagation

1 Introduction

Quantum computing and neural computing, both considered as promising nonstandard computation models, have attracted much research.

In the 1980s, Deutsch and others started the study of quantum computing by proposing a computer model that operates according to the principles of quantum mechanics. Since then, the study on quantum computing has intensified in order to make its computational properties clear [1]. In 1994, Shor [2] proposed a way to factorize large integers in polynomial time by using a quantum computing algorithm. This proved not only to be a milestone in quantum computing, but also considerably increased interest in it.

Neural information processing has also been intensively studied with useful results, e.g. in applications of pattern recognition [3]. Neural computing, though successful in modeling information processing in the brain, faces 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. That is, 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, we need a new computational principle.

Kak [4] and Peruš [5] 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 in accordance with quantum theory. Similar approaches have also been explored by Berman [6], Menneer [7], and Ventura [8]. To make progress with regard to this issue, we have attempted to establish a correspondence between neural networks and

quantum circuits by proposing a Qubit neuron model [9, 10].

In our neuron model, the states of neurons and their interactions with other neurons are based on the laws of quantum physics. Called the Qubit neuron model, this model has been shown, through various benchmark simulations [11, 12] and applications [13, 14], to significantly outperform classical neural networks with respect to its information processing capacity. This paper aims to examine through simulations the utility of the proposed model with respect to quantum characteristics such as quantum superposition and a probability interpretation. In the next section, we review the qubit model and its neural network. This is followed in Sect. 3 by simulations to examine the performance of the Qubit neural network. Then, in Sect. 4, we discuss what factors in the Qubit neuron model contribute to the efficient training performance on 4-bit and 6-bit parity check problems, which are basic benchmark tests.

2 Qubit neural network

The Qubit neuron model is a neuron model inspired by quantum physics and quantum computing: its neuron states are connected to quantum states, and transitions between neuron states are based on operations derived from quantum logic gates. To make the connection between the neuron states and the quantum states, we assume that the state of a firing neuron is defined as qubit state $|1\rangle$, the state of a nonfiring neuron is defined as qubit state $|0\rangle$, and the state of an arbitrary neuron is the coherent superposition of the two.

2.1 Quantum bit

In quantum computers, "qubits" are the counterparts of "bits" in classical computers [15], and they are used to store the states of circuits used for quantum computations. Labeled $|0\rangle$ and $|1\rangle$, the two quantum states express one bit of information: $|0\rangle$ corresponds to the bit 0 of classical computers, and $|1\rangle$ to the bit 1. Qubit state $|\phi\rangle$ maintains a coherent superposition of states.

$|0\rangle$ and $|1\rangle$ according to the expression

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{1}$$

where α and β are complex numbers called probability amplitudes. That is, when qubit state $|\phi\rangle$ collapses into either the $|0\rangle$ state or the $|1\rangle$ state, it does so with probabilities $|\alpha|^2$, $|\beta|^2$, respectively, where

$$|\alpha|^2+|\beta|^2= 1. \tag{2}$$

2.2 Quantum gates and their representations

Two frequently used quantum logic gates are the 1-bit rotation gate and the 2-bit controlled NOT gate. The 1-bit rotation gate ψ, shown in Fig. 1a, rotates a quantum state input to it over the angle in the complex plane. The controlled NOT gate, shown in Fig. 1b, performs an XOR operation. According to this operation, if quantum bit 'a' is $|0\rangle$, the output on the lower line is equal to quantum bit 'b', while if quantum bit 'a' is $|1\rangle$, the output is the reverse of quantum bit 'b' (through the NOT operation). It is possible to construct any arbitrary quantum logic gate using the 1-bit rotation gate and the 2-bit controlled NOT gate as primitive elements [16].

Another way to express the quantum state in (1) is

$$f(\varphi) = \mathrm{e}^{i\varphi} = \cos\varphi + i\cdot\sin\varphi, \tag{3}$$

where i is the imaginary unit $\sqrt{-1}$ and ϕ is the phase which describes the quantum state. Equation (3) makes it more convenient to express the following operations in terms of the operations of the rotation gate and the controlled NOT gate.

(a) The phase rotation operation: the rotation gate is a phase shifting gate that transforms the phase of quantum states. As the quantum state is represented by (3), the following relation holds:

$$f(\varphi_1 + \varphi_2) = f(\varphi_1)\cdot f(\varphi_2) \tag{4}$$

(b) The phase reverse operation: this operation is defined with respect to the controlled input parameter γ as follows:

$$f\left(\frac{\pi}{2}\gamma - \varphi\right) = \begin{cases} \sin\varphi + i\cos\varphi\ (\gamma = 1), \\ \cos\varphi - i\sin\varphi\ (\gamma = 0), \end{cases} \tag{5}$$

where $\gamma = 1$ corresponds to reversal rotation, and $\gamma = 0$ to nonrotation. In the case of $\gamma = 0$, the phase of the probability amplitude of quantum state $|1\rangle$ is reversed. However, its observed probability is invariant, so we are able to regard this case as nonrotation.

2.3 Qubit neuron model

Our Qubit neuron model is defined as follows (see Fig. 2).

The neuron state z that receives inputs from K other neurons is given by

$$u = \sum_{k}^{K} f(\theta_k)\cdot x_k - f(\lambda) = \sum_{k}^{K} f(\theta_k)\cdot f(y_k) - f(\lambda), \tag{6}$$

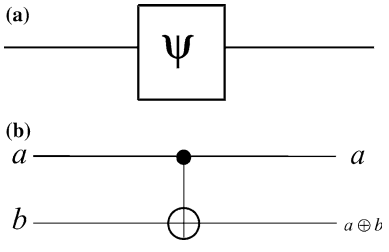

Fig. 1 **a** Single-bit rotation gate. **b** Two-bit controlled NOT gate

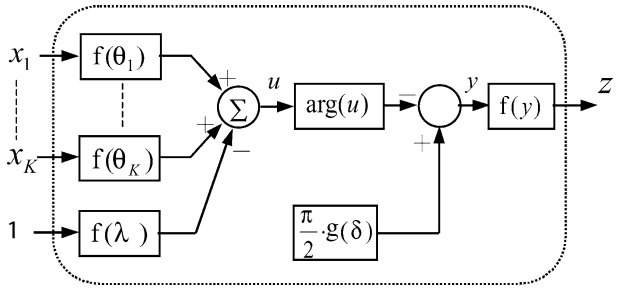

Fig. 2 Diagram of the Qubit neuron model

$$y = \frac{\pi}{2} \cdot g(\delta) - \arg(u), \tag{7}$$

$$z = f(y). \tag{8}$$

Here, f is the same function as defined in (3) and g is the sigmoid function which has the range [0, 1]. y_k is the quantum phase of the k-th neuron state x_k.

Two kinds of parameters exist in this neuron model: phase parameters in the form of weight connection θ_k and threshold λ, and the reversal parameter δ. The phase parameters correspond to the phase of the rotation gate, and the reversal parameter corresponds to the controlled NOT gate. By substituting $\gamma = g(\delta)$ in (5), we obtain a generalized reversal representation operating as the controlled NOT gate, the basic logic gate in quantum computing. Equation (6) 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. Equation (7) adjusts output qubit phase y more roughly than (6). In (7), arg(u) means the argument of the complex number u, and is implemented by arctan(Im(u)/Re(u)). Unlike in classical neural networks, the multiplication of weights $f(\theta_k)$ to the inputs $x_k = f(y_k)$ results in the rotation of the neuron state based on the rotation gate.

2.4 Qubit neural network

The neural network employing qubit neurons (called Qubit NN) is shown in Fig.3.

The sets $\{I_l\}(l = 1,2, \ldots, L)$, $\{H_m\}(m = 1,2, \ldots, M)$, and $\{O_n\}$ $(n = 1,2, \ldots, N)$ represent neuron elements, whereby the variables I, H, and O indicate the Input, Hidden, and Output layers, respectively. L, M, and N are the numbers of neurons in the input, hidden, and output layers, respectively. When input data (denoted by $input_l$) 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, π/2]. The output z_l^I of input neuron I_l:

$$z_l^I = f\left(\frac{\pi}{2} \cdot input_l\right) \tag{9}$$

becomes input to the hidden layer. The hidden and output layers with neurons from the sets $\{H_m\}$ and $\{O_n\}$, respectively, obey (6), (7), and (8). We obtain the output

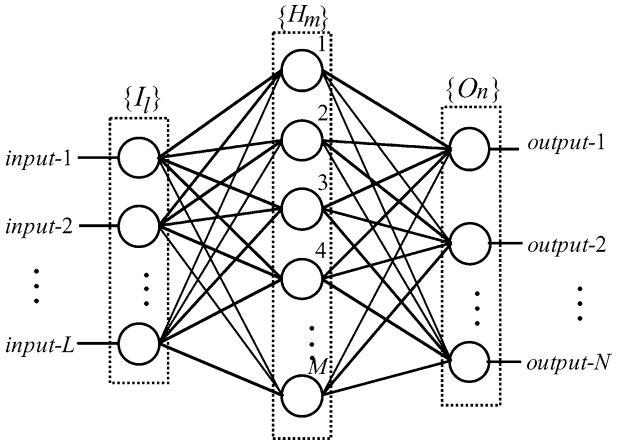

Fig. 3 Three-layer network for back-propagation learning

to the network, denoted by $output_n$, with the probability of the state in which $|1\rangle$ is observed from the nth neuron in the output layer:

$$output_n = \left|\mathrm{Im}(z_n^0)\right|^2. \tag{10}$$

2.5 Back-propagation algorithm on Qubit neural network

To incorporate learning ability in the neural network, we define a quantum version of the well-known Back-Propagation (BP) algorithm. The gradient-descent method, often used in the BP algorithm, is employed as the learning rule. This rule is expressed by the following equations:

$$\theta_k^{\mathrm{new}} = \theta_k^{\mathrm{old}} - \eta \frac{\partial E_{\mathrm{total}}}{\partial \theta_k}, \tag{11}$$

$$\lambda^{\mathrm{new}} = \lambda^{\mathrm{old}} - \eta \frac{\partial E_{\mathrm{total}}}{\partial \lambda}, \tag{12}$$

$$\delta^{\mathrm{new}} = \delta^{\mathrm{old}} - \eta \frac{\partial E_{\mathrm{total}}}{\partial \delta}, \tag{13}$$

where η is a leaning rate. E_{total} is the squared error function

$$E_{\mathrm{total}} = \frac{1}{2} \cdot \sum_p^P \sum_n^N (t_{p,n} - output_{p,n})^2, \tag{14}$$

which is to be minimized as part of the learning process. Here, P is the number of learning patterns, $t_{p,n}$ is the target signal for the n-th neuron and $output_{p,n}$ means $output_n$ of the network when it learns the p-th pattern.

3 Evaluating the network's performance by simulations

3.1 Definitions

In order to investigate the information processing ability of Qubit NN, we use benchmark learning problems for

training our network and compare its performance with that of a classical and of a complex NN. Here, a Classical NN is a network constructed with the well-known conventional neuron model expressed by the equations:

$$u = \sum_{m=1}^{M} w_m \cdot x_m + \theta, \tag{15}$$

$$y = (1 + \mathrm{e}^{-u})^{-1}. \tag{16}$$

A Complex NN is like a conventional NN model except that the neuron parameters are extended to the complex numbers W_l as the weight, X_l and Y as the neuron state, V as the threshold, and so on [17], giving rise to the following equations that correspond to (15) and (16), respectively:

$$U = \sum_{l}^{L} W_l \cdot X_l + V, \tag{17}$$

$$Y = (1 + \mathrm{e}^{-\mathrm{Re}(U)})^{-1} + i \cdot (1 + \mathrm{e}^{-\mathrm{Im}(U)})^{-1}. \tag{18}$$

Both NNs use the BP learning algorithm mentioned in Sect. 2.5. The neuron parameters w_m, θ of the Classical NN and W_l, V of the Complex NN are updated in the same way as in (11), (12) and (13), with the footnote that in the Complex NN the real parts, $\mathrm{Re}(W_l)$, $\mathrm{Re}(V)$, and the imaginary parts, $\mathrm{Im}(W_l)$ and $\mathrm{Im}(V)$, are updated independently.

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 the respective NN models are as follows:

$$\text{Qubit NN:} \quad NUM_q = LM + MN + 2M + 2N, \tag{19}$$

$$\text{Classical NN:} \quad NUM_{\mathrm{cls}} = LM + MN + M + N, \tag{20}$$

$$\text{Complex NN:} \quad NUM_{\mathrm{cmp}} = 2(LM + MN + M + N). \tag{21}$$

The subscripts, q, cls, cmp mean Qubit NN, Classical NN, Complex NN, respectively. In these simulations, we consider a network to have succeeded the training when the value of the squared error function E_{total} decreases to less than a certain bound 'E_{lower}', and it is considered to have failed when the network does not succeed within a certain number of learning iterations, denoted by 'L_{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 an 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_{upper} iterations. The success rates and the average numbers of learning iterations are indicators of the processing efficiency, by which the performances of the different models can be compared. The simulations in accordance with 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.

3.2 The 4-bit and 6-bit parity check problems

First of all, we investigate the dependence of training performance on the learning rates for the 4-bit and 6-bit parity check problems. For this simulation, we use a 4-6-1 Qubit NN, i.e., a 3-layered qubit NN that has four neurons in the input layer, six neurons in the hidden layer and one neuron in the output layer. According to (19), the 4-6-1 Qubit NN has 44 neural parameters to be trained. A 4-8-1 Classical NN with 49 neural parameters and a 2-6-1 Complex NN with 50 neural parameters are used for comparing their efficiency.

As in the case of the 4-bit parity check problem, for the simulations with the Classical NN and the 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 the scalars 0 and 1 obtained from XORing all elements of the 4-tuples inputs as target signals. In the case of the Complex 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. A difficult problem when conducting simulations on NNs is to find appropriate learning rates. 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 η_q, η_{cls}, η_{cmp} of the Qubit NN, the Classical NN, and the Complex 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, it follows that 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 range $[-\pi, \pi]$ for the Qubit NN, $[-1, 1]$ for the Classical NN, and $[-1-i1,\ 1+\ i1]$ for the Complex NN. L_{upper} is set to 3000 and E_{lower} is set to 0.005 in the 4-bit parity check problem.

In Fig. 4, the optimal average numbers of required iterations of the respective models are about 800 iterations at $\eta_q = 0.1$, about 1900 iterations at $\eta_{\mathrm{cls}} = 1.3$, and about 1380 iterations at $\eta_{\mathrm{cmp}} = 1.3$. Of all the models, the

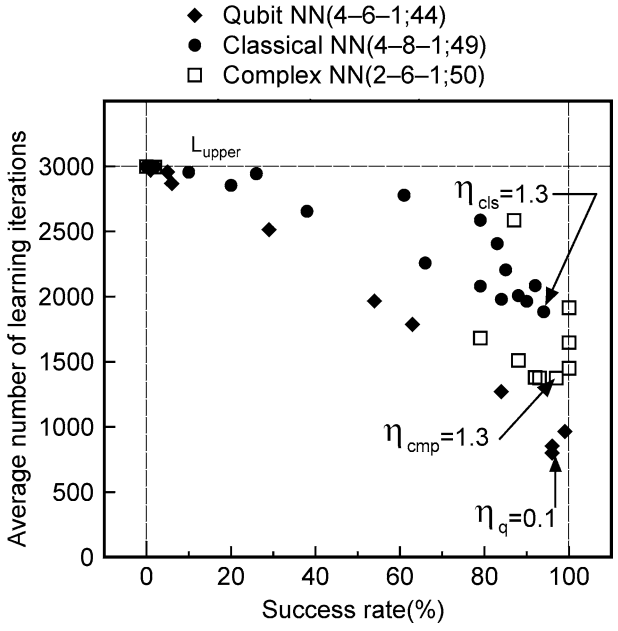

Fig. 4 Learning performance in the 4-bit parity check problem (average number of learning iterations vs. success rate). The condition for success is $E_{\text{lower}} = 0.005$, $L_{\text{upper}} = 3{,}000$

dot of the Qubit NN is the nearest to the corner right below.

We investigate 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. Figure 5 shows the dependency of the optimal average number of learning iterations on the number of neural parameters, and Fig. 6 shows that of the success rate. From these figures, we see that the Qubit NN excels over other 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 the Qubit NN decreases more than that of the other models.

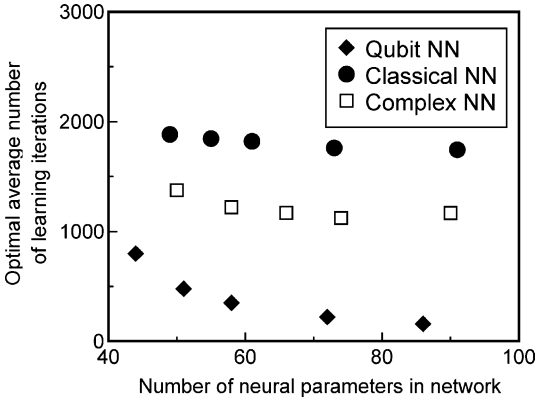

Fig. 5 Dependence of optimal average number of learning iterations on the number of neural parameters in the 4-bit parity check problem. Learning is considered successful if $E_{\text{lower}} = 0.005$, $L_{\text{upper}} = 3{,}000$. Only the Qubit NN accomplishes training with an average number of iterations lower than 500

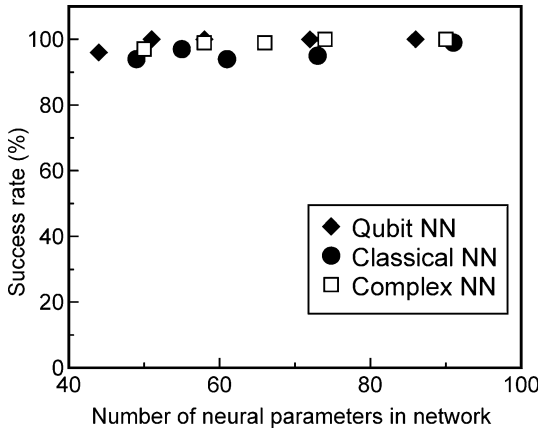

Fig. 6 Dependence of success rate, by which learning is accomplished within the optimal average number of iterations, on the number of neural parameters, in the 4-bit parity check problem

The result for the 6-bit parity check problem, which is a more complicated problem, is shown in Fig. 7. The input patterns for the 6-bit problem are established by following the same principles as in the 4-bit parity check problem. L_{upper} is set to 10,000 and E_{lower} is set to 0.006. The results for the Qubit NN are the nearest to the corner right below, similar to the 4-bit parity check. From Figs. 7–9, we conclude that the Qubit NN shows better efficiency than other neural networks for the 6-bit parity check problem and the 4-bit parity check problem.

3.3 General function identification problem

In this simulation, we test the abilities of the neural networks to learn the function defined by:

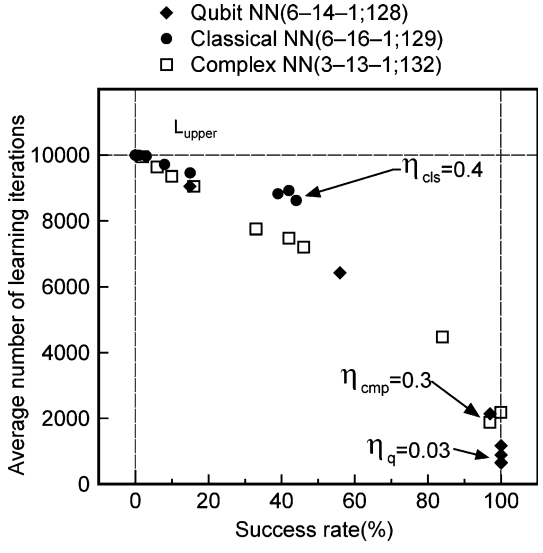

Fig. 7 Learning performance in the 6-bit parity check problem. Learning is considered successful if $E_{\text{lower}} = 0.006$, $L_{\text{upper}} = 10{,}000$. The Qubit NN finishes the training within 1,000 iterations

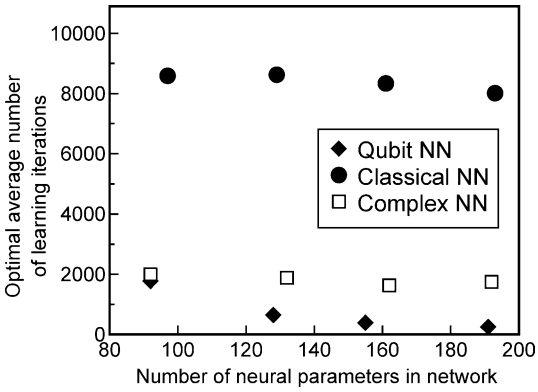

Fig. 8 Dependence of optimal average number of learning iterations on the number of neural parameters in the 6-bit parity check problem

$$P(x) = \frac{\sin \pi x + \sin 2\pi x + 2.0}{4.0} \quad (0 \leqslant x < 2). \tag{22}$$

We use a 2-14-1 Qubit NN which has 72 neural parameters, a 2-18-1 Classical NN with 73 parameters, and a 1-12-1 Complex NN with 74 parameters. Similar to the previous section, the networks have almost the same degrees of freedom. The input is set to $\{input_1, input_2\} = \{x, 0.0\}$ in the range $0.0 \leq x \leq 1.0$ and $\{input_1, input_2\} = \{0.0, x - 1.0\}$ in the range $1.0 < x \leq 2.0$, 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 Sect. 3.2. The value of E_{lower} is 0.01, and the value of L_{upper} is 10,000.

The learning abilities of the respective networks are shown in Fig. 10. From the figure, it can be seen that the Qubit NN requires 2,000 iterations to achieve a 100% success rate, while the Classical NN takes 4,500 iterations, and the Complex NN takes 3,100 iterations to achieve this.

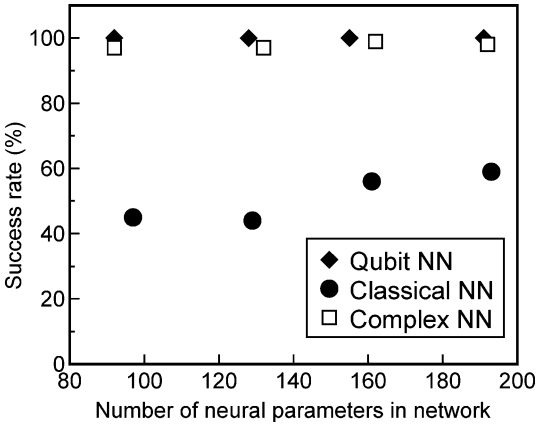

Fig. 9 Dependence of success rate, by which learning is accomplished within the optimal average number of learning iterations, on the number of neural parameters, in the 6-bit parity check problem

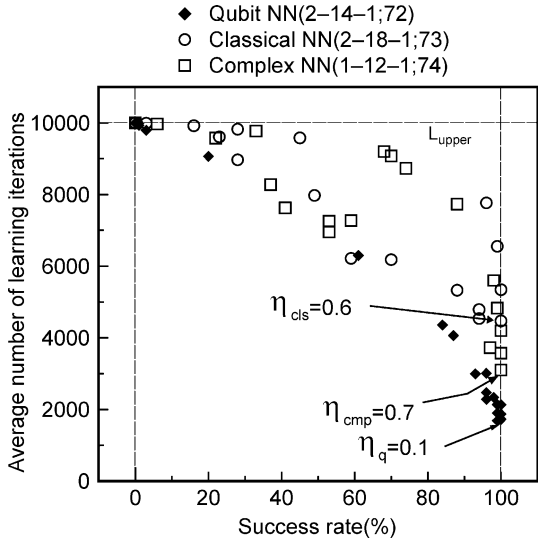

Fig. 10 Learning performance in the identification of an arbitrary function. The condition for success is $E_{\text{lower}} = 0.01$, $L_{\text{upper}} = 10{,}000$

Finally, we check how well the 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 Fig. 11.

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.

4 What contributes to the efficiency of the Qubit neural network model?

In this section, we discuss the factors that contribute to the efficiency of the proposed Qubit NN by analyzing the results of the simulations on the 4-bit and 6-bit parity check problems.

The Qubit NN restricts the use of complex numbers in its states to those with polar radius "1" due to the state descriptions based on quantum superposition,

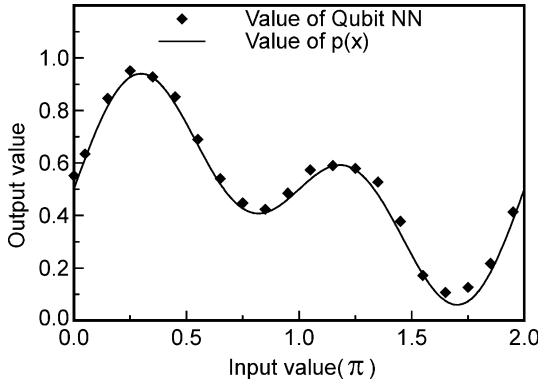

Fig. 11 Ability of the network to approximate the function $p(x)$

whereas the Complex NN uses complex numbers in its states without this restriction. To check whether the better performance of the Qubit NN was not due to its states being restricted to polar radius 1, we carried out simulations on Complex 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 Fig. 12). The number of neurons in the hidden layers of the radius-restricted complex NN is more than twice that of the free-radius Complex 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 NN are inferior to the results obtained for the Qubit NN.

To clarify the advantages of the quantum descriptions, i.e., the quantum superposition in (1) or (2) and the probability interpretation in (8), we evaluate Qubit NNs from which the probability interpretation or the quantum superposition are removed. Figures 13 and 14 show the results of simulations on 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 $n\pi/2$ ($n=0,1,2,\ldots$). The removal of superposition substantially downgraded the performance of the model in both the 4-bit and 6-bit cases. By changing the $output_n = |\mathrm{Im}\,(z_n^0)\,|^2$ into $|\mathrm{Im}\,(z_n^0)|$in (8), we realized a model without a probability interpretation. The results in the 4-bit and 6-bit cases are shown in Figs. 15 and 16. From these figures, we see that the removal of the partial description or features from the Qubit NN resulted in a decrease of its performance.

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

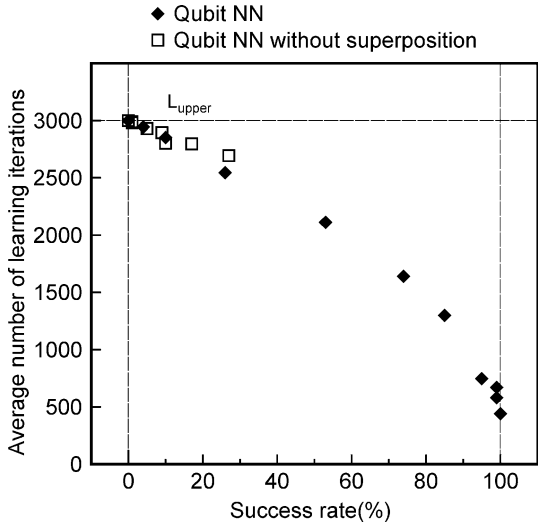

Fig. 13 Learning performance when the superposition of the hidden layer neurons is removed, in the 4-bit parity check problem. The conditions are the same as in Fig. 4

5 Conclusion

We have investigated the learning ability of the Qubit NN through the 4 and 6-bit parity check problems and the General Function Identification Problem. In all simulations, the Qubit NN has more efficient processing abilities than the Classical NN and the Complex NN. The results suggest that the excellent learning performance of our Qubit NN is not simply due to the complex-valued neuron parameters or the introduction

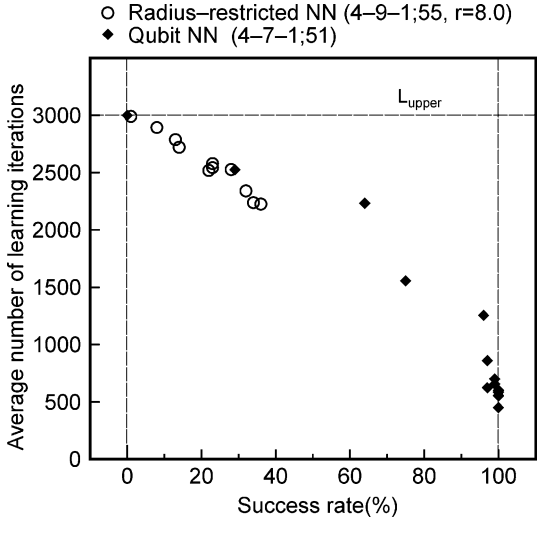

Fig. 12 Learning performance, in the 4-bit parity check problem, of a radius-restricted complex NN and the Qubit NN. Learning is considered successful if $E_{\mathrm{lower}} = 0.005$, $L_{\mathrm{upper}} = 3{,}000$

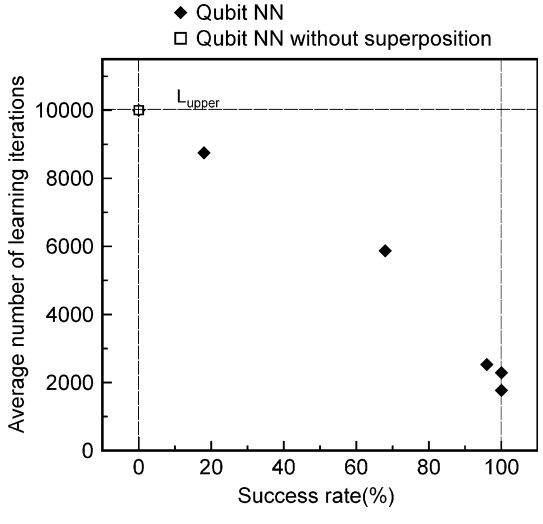

Fig. 14 Learning performance when the superposition of the hidden layer neurons is removed, in the 6-bit parity check problem. The conditions are the same as in Fig. 7. In the removal case, the training cannot be successfully completed

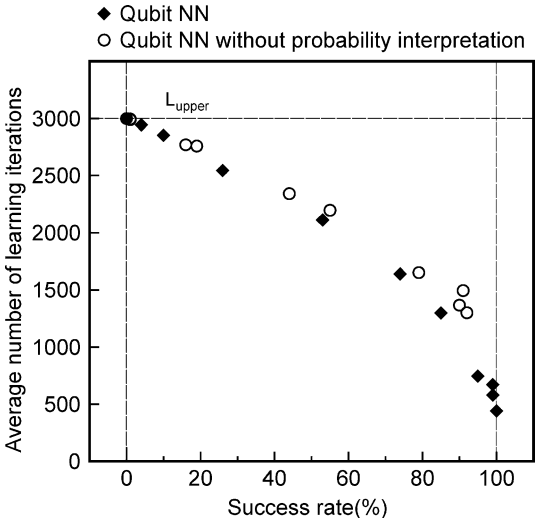

Fig. 15 Learning performance when the probability interpretation is removed from the network output, in the 4-bit parity check problem

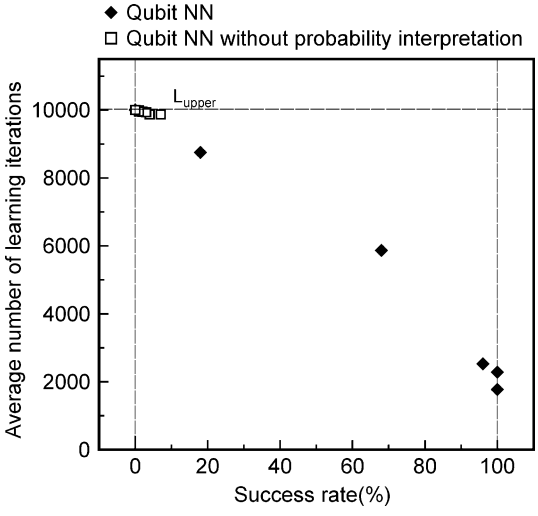

Fig. 16 Learning performance when the probability interpretation is removed from the network output, in the 6-bit parity check problem. In the removal case, the success rate stays below 10%, even in the optimal case

of the restricted polar radius, but to the quantum characteristics, i.e., the quantum superposition and the probability interpretation. The combination of neural computing and quantum descriptions improves the efficiency of learning performances. It is left for future study to clarify this efficiency in mathematical terms.

Other future work will focus on the use of the Qubit NN in more practical applications such as motion control of nonholonomic systems, and the comparison of its processing abilities with other classical NNs.

Acknowledgements This study is financially supported by Grant-in-Aid for Scientific Research (C-13680458), Japan Society for the Promotion of Science.

References

1. Deutsch D, Jozsa R (1992) Rapid solution of problems by quantum computation. Proc R Soc Lond Ser A 439:553–558
2. Shor PW (1994) Algorithm for quantum computation: discrete logarithms and factoring. In: Proceedings of the 35th annual IEEE symposium on foundations of computer science, pp 124–134
3. Arbib MA (2002) The handbook of brain theory and neural networks. MIT press, Cambridge
4. Kak SC (1995) On quantum neural computing. Information Sciences 83:143–163
5. Peruš M (1996) Neuro-quantum parallelism in brain-mind and Computers. Informatica 20:173–183
6. Behrman EC, Niemel J, Steck JE, Skinner SR (1996) A quantum dot neural network. In: Proceedings of the workshop on physics of computation, pp 22–24
7. Menneer TSI, Narayanan A (1995) Quantum-inspired neural networks. Technical report University of Exeter Research Report R 329
8. Ventura D, Martinez T (1997) An artificial neuron with quantum mechanical properties. In: Proceedings of the international conference on neural networks and genetic algorithms, pp 482–485
9. Matsui N, Takai M, Nishimura H (1998) A network model based on qubit-like neuron corresponding to quantum circuit (in Japanese). IEICE J81-A 12:1687–1692. (2000) Electronics and Communications in Japan 3 83 10:67–73
10. Kouda N, Matsui N, Nishimura H (2002) A multi-layered feed-forward network based on Qubit neuron model (in Japanese). IEICE J85-DII 12:641–648. (2004) Systems and Computer in Japan 35 13:43–51
11. Matsui N, Kouda N, Nishimura H (2000) Neural network based on QBP and its performance. In: Proceedings of international joint coference on neural networks '2000 III:247–252
12. Kouda N, Matsui N, Nishimura H (2000) Learning performance of neuron model based on quantum superposition. In: Proceedings of the 2000 IEEE international workshop on robot and human interactive communication, pp 112–117
13. Kouda N, Matsui N, Nishimura H (2002) Image compression by layered quantum neural networks. Neural Processing Lett 16(1):67–80
14. Kouda N, Matsui N, Nishimura H (2002) Control for swing-up of an inverted pendulum using Qubit neural network. In: Proceedings of SICE annual conference 2002 in Osaka, pp 805–810
15. DiVincenzo DP (1995) Quantum Computation. Science 270:255–261
16. Barenco A, Bennet CH, Cleve R, DiVicenzo DP, Margolus N, Shor P, Sleator T, Smolin J, Weinfurter H (1995) Elementary gates for quantum computation. Phys Rev A 52:3457–3467
17. Nitta T (1997) An extension of the back-propagation algorithm to complex numbers. Neural Netw 10(8):1391–1415