Nonlinear quantum neuron: A fundamental building block for quantum neural networks

Shilu Yan,[1] Hongsheng Qi,[2,3] and Wei Cui[1,*]

[1]*School of Automation Science and Engineering, South China University of Technology, Guangzhou 510640, China*

[2]*Key Laboratory of Systems and Control, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China*

[3]*School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China*

(Received 1 August 2020; accepted 4 November 2020; published 25 November 2020)

Quantum computing enables quantum neural networks (QNNs) to have great potential to surpass artificial neural networks. The powerful generalization of neural networks is attributed to nonlinear activation functions. Although various models related to QNNs have been developed, they are facing the challenge of merging the nonlinear, dissipative dynamics of neural computing into the linear, unitary quantum system. In this paper, we establish different quantum circuits to approximate nonlinear functions and then propose a generalizable framework to realize any nonlinear quantum neuron. We present two quantum neuron examples based on the proposed framework. The quantum resources required to construct a single quantum neuron are polynomial in function of the input size. Finally, both IBM Quantum Experience results and numerical simulations illustrate the effectiveness of the proposed framework.

DOI: 10.1103/PhysRevA.102.052421

I. INTRODUCTION

Artificial neural networks are intelligent computing models vaguely inspired by the biological neural networks that constitute animal brains. Various artificial neural network (ANN) models have been widely applied in diverse fields such as computer vision, speech recognition, machine translation, and medical diagnosis [1]. As a new type of computing paradigm with quantum properties, quantum computing has been exhibited to be exponentially or geometrically faster than classical counterparts to solve certain computational problems [2–4]. Combining neural networks with quantum computing, QNNs are considered to be superior to the ANNs in memory capacity, information processing speed, network scale, stability, and reliability [5]. There have been many approaches to build QNNs from different perspectives [6,7]. In recent years, connecting quantum circuit models to neural network architectures has attracted extensive research interest [8–17].

Even the most complicated neural networks are built by a regular connection of identical units called neurons. Usually, a neuron includes two operations: one is the inner product and the other is represented by an activation function. The inner product is a linear operation and generally the activation function is nonlinear. Linear operations are easily realized with quantum computing. As for activation functions, most quantum neurons would like to use the threshold function due to its easy implementation. Reference [10] proposes a type of quantum neuron that can simulate a sigmoid-like nonlinear function in light of developed repeat-until-success (RUS) techniques [18–20]. Reference [12] further proposes a nonperiodic nonlinear activation function based on RUS circuit and its quantum neuron can be trained with efficient gradient descent. Reference [21] theoretically proposes an architecture to realize any nonlinear quantum neuron using quantum phase estimation. More existing solutions to nonlinear quantum neurons include the quadratic form of the kinetic term [22], dissipative quantum gates [23] and reversible circuits [11], etc. Although there has been a lot of discussions about nonlinear quantum neurons, these solutions are restricted to exhibit full nonlinear properties except Ref. [21]. However, Ref. [21] does not give full play to the quantum advantage to obtain well-performed quantum neurons, and it still has the problem of high quantum resource cost.

To explore universal methods of building any well-functioning and resource-saving quantum neuron, we establish different quantum circuits to approximate nonlinear functions. Moreover, a quantum framework with strong generalization is proposed to obtain any nonlinear quantum neuron. We present two quantum neuron examples based on the proposed framework. The neurons satisfy the criteria proposed by Ref. [6] in a way that naturally combines the advantages of quantum computing and neuron networks. The quantum resources required to construct a single quantum neuron are polynomial functions in the size of the input. Finally, both IBM Quantum Experience results and numerical simulations illustrate the effectiveness of the proposed framework.

This paper is organized as follows. Section II describes two circuits of approximating nonlinear functions. In Sec. III, we propose a generalized framework to implement any nonlinear quantum neuron and present the analysis of neuron examples. Section IV verifies our results by IBMQ experiments and numerical simulations. Conclusions and discussions are presented in Sec. V.

Notation. X, Y, and Z are Pauli matrices. I represents the identity matrix and H is the Hadamard gate. Throughout the paper, the base of the logarithm defaults to 2. $[\alpha, \beta]$ indicates an interval from decimal α to decimal β. The norm $\|\cdot\|$ always refers to the 2-norm of vectors. The bold italic $\boldsymbol{i}$ represents the imaginary unit of the complex field. x^T is the

*aucuiwei@scut.edu.cn

transpose of the column vector x. y^D indicates the decimal representation of the binary y. The symbol $\oplus$ represents the bitwise XOR. $FT^\dagger$ is the inverse quantum Fourier transform. The S gate and $R_z(\alpha)$ gate are defined as

$$S = \begin{bmatrix} 1 & 0 \\ 0 & \boldsymbol{i} \end{bmatrix}, \quad R_z(\alpha) = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i\alpha} \end{bmatrix}.$$

II. QUANTUM CIRCUITS OF APPROXIMATING NONLINEAR FUNCTIONS

The basic task of a computer is assigning values to Boolean functions, which is to give a one-bit output for an n-bit input [24]. Digital computers compute any complicated function by combining such Boolean functions. Similarly, quantum computers can do the task with black-boxed quantum oracles. There has been some research on quantum oracles [25–28]. An oracle may be a circuit, a physical device, or a pure theory, which helps transform a system from a state to another state.

For a general Boolean function $f : \{0,1\}^n \to \{0,1\}^m$, when $m = 1$, a standard oracle S_f is defined to act on two input states and return two outputs,

$$S_f : |x\rangle |y\rangle \mapsto |x\rangle |y \oplus f(x)\rangle, \tag{1}$$

where $|x\rangle \in (\mathbb{C}^2)^{\otimes n}$, $|y\rangle \in \mathbb{C}^2$. The following two oracles are examples of standard oracles: The function $f_D : \{0,1\} \to \{0,1\}$ is calculated by Deutsch's oracle

$$S_{f_D} : |x\rangle |-\rangle \mapsto |x\rangle |- \oplus f_D(x)\rangle, \tag{2}$$

where $|- \oplus f_D(x)\rangle = e^{\pi i f_D(x)} |-\rangle$. The function $f_G : \{0,1\}^n \to \{0,1\}$ is calculated by Grover's oracle

$$S_{f_G} : |x\rangle |-\rangle \mapsto |x\rangle |- \oplus f_G(x)\rangle, \tag{3}$$

where $|- \oplus f_G(x)\rangle = e^{\pi i f_G(x)} |-\rangle$.

When the Boolean function f satisfies $m \geqslant 2$, we present the following general oracle:

$$P_f : |x\rangle |0\cdots01\rangle \mapsto e^{\frac{2\pi i}{2^m} f(x)} |x\rangle |0\cdots01\rangle, \tag{4}$$

where $|0\cdots01\rangle \in (\mathbb{C}^2)^{\otimes m}$ and only the last qubit of the $|0\cdots01\rangle$ is in the state $|1\rangle$. The oracles in Eqs. (2)–(4) have the minimal forms, which we denote as minimal phase oracles. Without any auxiliary qubit, a minimal phase oracle M_f is defined to act on one input state and return one output with a specific phase,

$$M_f : |x\rangle \mapsto e^{\frac{2\pi i}{2^m} f(x)} |x\rangle. \tag{5}$$

Since S_f and P_f are equivalent to M_f in the sense that they can denote the same Boolean function f, S_f and P_f can be converted to M_f. In Table I, the Boolean functions are needed to be computed by corresponding oracles. Deutsch's oracle, Grover's oracle, and the general oracle (4) can be transformed to minimal phase oracles, which are explicitly expressed as diagonal unitary matrices.

In Fig. 1, we demonstrate two natural quantum circuits to implement the Boolean function $f : \{0,1\}^n \to \{0,1\}^m$ with a general oracle

$$U_f : |x\rangle |0\rangle^{\otimes m} \mapsto |x\rangle |f(x)\rangle. \tag{6}$$

After U_f acting on two input states $|x\rangle |0\rangle^{\otimes m}$, we could recover

$$f(x) = f_{m-1}(x) \times 2^{m-1} + \cdots + f_0(x) \times 2^0 \tag{7}$$

TABLE I. Transformations of quantum oracles.

	Boolean functions	Minimal phase oracles
Deutsch's oracle	$\{0,1\} \to \{0,1\}$	$\sum_{x=0}^{2^n-1} e^{\pi i f_D(x)} \|x\rangle \langle x\|$
Grover's oracle	$\{0,1\}^n \to \{0,1\}$	$\sum_{x=0}^{2^n-1} e^{\pi i f_G(x)} \|x\rangle \langle x\|$
The oracle in Eq. (4)	$\{0,1\}^n \to \{0,1\}^m$	$\sum_{x=0}^{2^n-1} e^{\frac{2\pi i}{2^m} f(x)} \|x\rangle \langle x\|$

by introducing measurements to the second output state. For $i = 0, 1, \ldots, m-1$, $f_i(x) \in \{0,1\}$ denotes the binary expansion of the $f(x)$. Equation (7) can be regarded as a nonlinear function whose binary input and first m bits of binary output are equal to x and $f(x)$, respectively. Thus, by Eqs. (6) and (7), we note that Boolean functions approximate the corresponding nonlinear functions through quantum oracles.

Figure 1(a) shows the first quantum circuit of U_f in Eq. (6), where $U_{f_i(x)} \in \{I, X\}$. The circuit in Fig. 1(a) can be understood as a process of function assignment. Each bit of the input string x controls each bit of the output string $f(x)$. Then a specific output $f(x)$ is carried out.

To explore the second quantum circuit of U_f in Eq. (6), the basic procedures are stated as follow. First, we construct a typical minimal phase oracle

$$O_f = \sum_{x=0}^{2^n-1} e^{\frac{2\pi i}{2^m} f(x)} |x\rangle \langle x|, \tag{8}$$

which adds a phase factor related with the $f(x)$ to any input $|x\rangle$. Second, we use the inverse quantum Fourier transform to recover the $f(x)$ with a certain precision. Figure 1(b) shows the second quantum circuit of U_f in Eq. (6). The controlled-O_f^r gates are treated as a composition of a series of controlled-$O_f^{2^s}$ gates by considering the sth qubit in the first register as the control qubit, where $s = 0, 1, \ldots, m-1$.

III. NONLINEAR QUANTUM NEURONS

Deep learning needs a lot of computation resources to train a model. As the amount of transistors in silicon chips

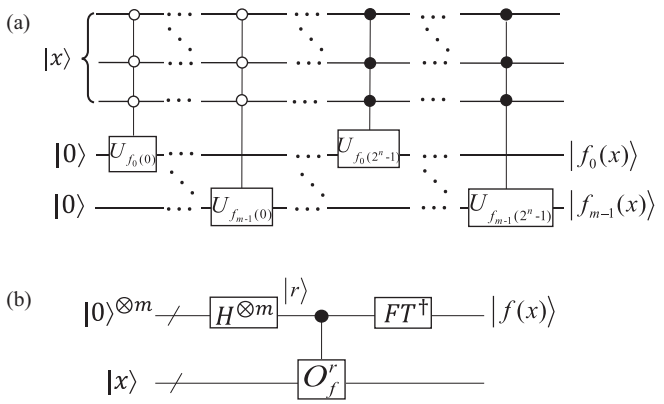

FIG. 1. Panels (a) and (b) show two quantum circuits for implementing U_f in Eq. (6).

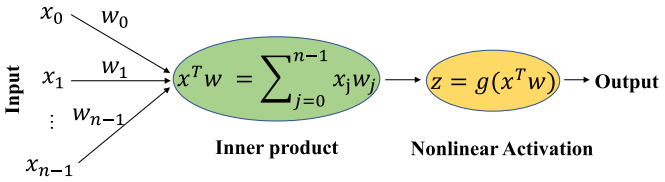

FIG. 2. A schematic of a classical neuron.

approaches the physical limit, QNN is a potential solution to deal with massive and ultrahigh-dimensional data. However, a recognized definition of QNN has not been proposed on account of the different dynamics between ANNs and quantum computing.

Figure 2 shows the schematic of a classical neuron. As the fundamental building block of ANNs, a neuron classically maps an input vector $x = (x_0, \ldots, x_{n-1})^T \in \mathbb{R}^n$ to an output $z = g(x^T w)$, where $w = (w_0, \ldots, w_{n-1})^T \in \mathbb{R}^n$ is the weight vector. $x^T w$ is the inner product, and g is usually a nonlinear activation function.

Motivated by this schematic, we propose a generalizable framework of implementing nonlinear quantum neurons in Fig. 3. The framework maps the existing neuron designed for classical computers to quantum circuits and achieves the nonlinear mapping of classical data through three processes: encoding, evolution, and measurement. The encoding is to represent the features of classical data by a quantum system. Following the postulates of quantum mechanics, the evolution maps a quantum state $|x\rangle$ onto $|z\rangle$ with unitary operators, which are the key to successfully implement nonlinear quantum neurons. Reducing the dimension of the state space, the measurement is necessary to induce nonlinear quantum neurons. When quantum neurons are connected to build QNNs, the measurement of each neuron in all middle layers should be postponed until the last layer.

A. Neuron examples

Let $\mathcal{M} \triangleq \{x^i : i = 1, 2, \ldots, q\} \in \mathbb{R}^n$ be a classical dataset. We denote a sample vector as $x^i = (x_0^i, x_1^i, \ldots, x_{n-1}^i)^T$, where x_j^i for $j = 0, 1, \ldots, n-1$ represents the jth component value of the ith sample in $\mathcal{M}$. $\mathcal{M}$ has been processed to ensure that each x_j^i satisfies $0 \leqslant x_j^i < 1$. Reference [29] has systematically introduced encoding methods and discussed state preparation routines.

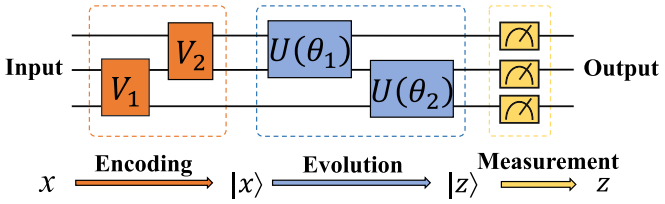

FIG. 3. The proposed framework of implementing a nonlinear quantum neuron. V_1 represents the initialization of auxiliary qubits. V_2 refers to the preparation of quantum dataset. $U(\theta_1)$ denotes the calculation of the inner product of the input vector and the weight vector. $U(\theta_2)$ denotes the approximation of nonlinear functions. θ_1 and θ_2 are the circuit parameters.

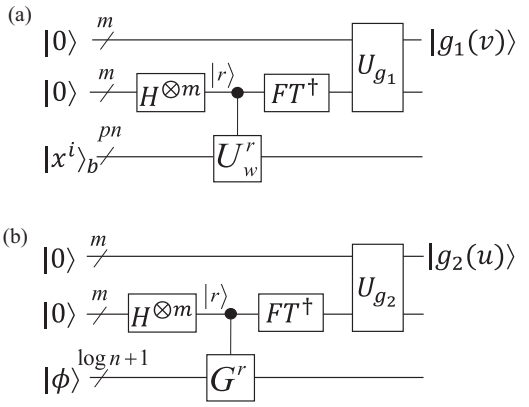

FIG. 4. Panels (a) and (b) show nonlinear quantum neuron examples based on basis encoding and amplitude encoding, respectively.

Since data can be encoded into computational basis states or amplitudes, we apply basis encoding and amplitude encoding to our framework.

Figure 4 shows two nonlinear quantum neuron examples. Both neurons are composed of three quantum registers, which are labeled as the first register, the second register, and the third register (from top to bottom). If we perform measurements on the first register, we can obtain the output of a quantum neuron. The second register is an auxiliary register to calculate the inner product of the input vector and the weight vector. The third register is the sample encoding register, which is used to encode the classical data into quantum states. The parameter m is an integer that always relates to precision. In the following analysis, we denote the weight vector w to compute inner product with the sample x^i.

Figure 4(a) is an example of nonlinear quantum neurons based on basis encoding. Basis encoding transforms each x^i into a product state:

$$|x^i\rangle_b = \left|x_0^i, x_1^i, \ldots, x_{n-1}^i\right\rangle \in (\mathbb{C}^2)^{\otimes pn}, \tag{9}$$

where p is a fixed number of qubits for a given precision to approximate the component x_j^i, $x_j^i = x_{j_1}^i/2^1 + \cdots + x_{j_p}^i/2^p$, $x_{j_k}^i \in \{0, 1\}$ for $k = 1, 2, \ldots, p$. By encoding classical data into an orthonormal basis of the Hilbert space, quantum parallelism allows a linear combination of the basis to represent $\mathcal{M}$ in only one quantum state, which means it is possible to process all data in parallel.

When the controlled-phase gate $CR_z(\frac{\alpha}{2^m})$ acts on two quantum states $|+\rangle |x_{j_k}^i\rangle$, we obtain

$$CR_z\left(\frac{\alpha}{2^m}\right)\left(|+\rangle \left|x_{j_k}^i\right\rangle\right) = \frac{1}{\sqrt{2}}\left(|0\rangle \left|x_{j_k}^i\right\rangle + e^{\frac{2\pi i}{2^m} x_{j_k}^i \alpha} |1\rangle \left|x_{j_k}^i\right\rangle\right), \tag{10}$$

where $\frac{\alpha}{2^m}$ is a quantum phase factor and $CR_z(\frac{\alpha}{2^m})$ can be written as

$$CR_z\left(\frac{\alpha}{2^m}\right) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{\frac{2\pi i}{2^m}\alpha} \end{bmatrix}. \tag{11}$$

Thus, in Fig. 4(a), the controlled-U_w^r gates are constructed with a series of $CR_z(2^{-(k+m)} w_j)$ gates, which are performed

on the kth qubit of the jth component of x^i. Meanwhile, the controlled-U_w^r gates can be regarded as a composition of a series of controlled-$U_w^{2^s}$ gates by considering the sth qubit in the second register as the control qubit. After the application of the controlled-$U_w^{2^s}$ on the second and third registers, these two registers become

$$\frac{1}{\sqrt{2^m}}\sum_{t=0}^{2^m-1} e^{\frac{2\pi i}{2^m}t(x^i)^T w}\,|t\rangle\,|x^i\rangle_b\,.$$

To apply the inverse quantum Fourier transform to the second register, the output of the second register is $|v\rangle$, where $v \in \{0,1\}^m$. Since $(x^i)^T w$ can be positive or negative, we denote the v as the original code of a number, whose complement is an approximation of $(x^i)^T w$. Then we use the Boolean function $g_1(v) : \{0,1\}^m \to \{0,1\}^m$ to approximate any activation function. The $g_1(v)$ is calculated by the oracle

$$U_{g_1} : |v\rangle\,|0\rangle^{\otimes m} \mapsto |v\rangle\,|g_1(v)\rangle, \tag{12}$$

which can be implemented by the circuit in Fig. 1(a) or 1(b).

Figure 4(b) is an example of nonlinear quantum neurons based on amplitude encoding. Amplitude encoding encodes the sample x^i into an entangled state

$$|x^i\rangle_a = \frac{1}{\|x^i\|}\sum_{j=0}^{n-1} x_j^i\,|j\rangle \in \mathbb{C}^n. \tag{13}$$

This process only uses log$n$ qubits to represent a sample consisting of n features, where without loss of generality, we have assumed that the value of $\log n$ is an integer for convenience.

In Fig. 4(b), the weight vector w is encoded in

$$|w\rangle = \frac{1}{\|w\|}\sum_{j=0}^{n-1} w_j\,|j\rangle \in \mathbb{C}^n. \tag{14}$$

The quantum states $|x^i\rangle_a$ and $|w\rangle$ could be carried out by quantum random access memory (qRAM) [30], which is an efficient algorithm to prepare the quantum state under certain data structures. The inner product $\langle w|x^i\rangle_a$ could be estimated with swap test which has been widely used in quantum machine learning [13,16,31,32]. First, we can prepare the quantum state,

$$|\phi\rangle = \frac{1}{\sqrt{2}}(|+\rangle\,|x^i\rangle_a + |-\rangle\,|w\rangle), \tag{15}$$

which can be rewritten as

$$|\phi\rangle = \tfrac{1}{2}[|0\rangle\,(|x^i\rangle_a + |w\rangle) + |1\rangle\,(|x^i\rangle_a - |w\rangle)]. \tag{16}$$

The amplitude of $|0\rangle$ is

$$\sin\gamma = \sqrt{1+\langle w|x^i\rangle_a}/\sqrt{2}, \tag{17}$$

and the amplitude of $|1\rangle$ is

$$\cos\gamma = \sqrt{1-\langle w|x^i\rangle_a}/\sqrt{2}, \tag{18}$$

where $\gamma \in [0, \frac{\pi}{2}]$. With the Schmidt decomposition, $|\phi\rangle$ is decomposed into

$$|\phi\rangle = \frac{-i}{\sqrt{2}}(e^{i\gamma}\,|w_+\rangle - e^{-i\gamma}\,|w_-\rangle), \tag{19}$$

TABLE II. Comparison of different quantum neurons.

Quantum neurons	Input features	Number of qubits	Gate complexity
Fig. 4(a)	Binary	$pn+2m$	$O(pn+m2^m)$
Fig. 4(b)	Continuous	$\log n+2m+1$	$O(mn^2+m2^m)$
Ref. [21]	Binary	$2pn+m$	$O(m2^{2pn}+m^2)$

where

$$|w_\pm\rangle = \frac{1}{\sqrt{2}}[|0\rangle\,(|x^i\rangle_a + |w\rangle) \pm \boldsymbol{i}\,|1\rangle\,(|x^i\rangle_a - |w\rangle)]. \tag{20}$$

Second, we can construct a unitary transformation

$$G = (I^{\otimes(\log n+1)} - 2\,|\phi\rangle\,\langle\phi|)(Z\otimes I^{\otimes\log n}). \tag{21}$$

The eigenvalues of G are $e^{\pm i2\gamma}$ and the corresponding eigenvectors are $|w_\pm\rangle$. Then we can use quantum phase estimation to estimate γ. The controlled-G^r gates denote a composition of a series of controlled-G^{2^s} gates by considering the sth qubit in the second register as the control qubit. After applying quantum phase estimation, the output of the second and third register is

$$|\psi\rangle = \frac{-i}{\sqrt{2}}(e^{i\gamma}\,|u\rangle\,|w_+\rangle - e^{-i\gamma}\,|2^m-u^D\rangle\,|w_-\rangle), \tag{22}$$

where the first qubit of $|u\rangle$ is in state $|0\rangle$ and $u \in \{0,1\}^m$. $u^D\pi/2^{m-1}$ is an approximation of 2γ. By Eq. (18), we obtain

$$u^D \approx \arccos(-\langle w|x^i\rangle_a)2^{m-1}/\pi\,. \tag{23}$$

Then we use the Boolean function $g_2(u) : \{0,1\}^m \to \{0,1\}^m$ to approximate any activation function. The $g_2(u)$ is calculated by the oracle

$$U_{g_2} : |u\rangle\,|0\rangle^{\otimes m} \mapsto |u\rangle\,|g_2(u)\rangle, \tag{24}$$

which can be implemented by the circuit in Fig. 1(a) or 1(b).

By Eqs. (12) and (24), we note that the outputs of the second registers can be seen as the inputs of the Boolean functions. Other existing quantum computing models can apply to our framework. So long as the outputs of the second registers are the bijective functions with respect to $(x^i)^T w$, a specific oracle would be available to approximate any corresponding activation function.

B. Comparisons

The generalizable framework has been proposed to implement nonlinear quantum neurons, which are suitable to deal with massive and ultrahigh-dimensional data. The numbers of qubits and elementary quantum gates measure the quantum resources to implement a quantum neuron. Table II shows the comparisons of different nonlinear quantum neurons, with n representing the input size (the dimensions of input vectors).

Considering the space cost, the m and p are parameters that influence the binary representation precisions of the numbers. For example, the long int types and double-precision floating-point types are represented by fixed 64 bits in a common digital computer. Reference [21] encodes input vectors and the corresponding weight vectors into the basis states. If each component of the vectors is represented by p qubits,

Ref. [21] needs $2pn + m$ qubits to implement its quantum neuron. From the analysis in Sec. III A, Fig. 4(a) requires $pn + 2m$ qubits for a neuron and Fig. 4(b) needs $\log n + 2m + 1$ qubits. When n is much larger than the parameters m and p, Fig. 4(a) uses about half fewer qubits than Ref. [21]. Moreover, Fig. 4(b) exponentially reduces the requirement of qubits.

Turning to the time cost, the elementary gates are physically implementable gates, include single-qubit rotations and entangling two-qubit gates. A general unitary operator can be decomposed into elementary gates following the general principles presented by Refs. [33,34]. We use $C^{d-1}R_z$ gates to represent multiple controlled-R_z gates acting on $d-1$ control qubits and one target qubit. A $C^{d-1}R_z$ gate can be simulated by $O(d^2)$ elementary gates or else $O(d)$ gates with one auxiliary qubit ([33], Corollary 7.6). The accurate evaluation of diagonal unitary operators is often the most resource-intensive element of quantum algorithms. There are some discussion of decomposition methods for arbitrary diagonal unitaries [35–37]. Generally, an arbitrary d-qubit diagonal unitary operator could be decomposed into $2^d\, C^{d-1}R_z$ gates. Therefore, constructing quantum circuits for diagonal computations generally requires $O(d^2 2^d)$ elementary gates without any auxiliary qubit. Specially, Ref. [35] provides circuits of size $O(2^d)$ for arbitrary d-qubit diagonal unitaries, which is the best-known compiling algorithm. An efficient implementation of the phase estimation includes $O(m^2)$ elementary gates for an inverse quantum Fourier transform and one call to controlled unitary operator black box [38]. Here, we would discuss the elementary gate complexity of constructing a black box rather than the query complexity. Reference [21] consists of arbitrary $2pn$-qubit diagonal operations, which should be decomposed into $O(2^{2pn})$ elementary gates with the asymptotically optimal circuits of standard techniques [35]. Therefore, Ref. [21] needs $O(m2^{2pn} + m^2)$ gates to implement a single neuron. In our cases, the corresponding unitary operator black box in Figs. 4(a) and 4(b) are the U_w^r gate and the G^r gate, respectively. The analysis in Sec. III A shows that the U_w^r gate can exactly be decomposed into $pn\, R_z$ gates. The G^r gate can be expressed as $O(n^2)$ elementary gates according to the decomposition method for general multiqubit gates in Ref. [34]. If U_{g_1} and U_{g_2} are implemented by Fig. 1(b), the gate complexity is $O(m2^m)$. In summary, the total gate complexity of the Figs. 4(a) and 4(b) is $O(pn + m2^m)$ and $O(mn^2 + m2^m)$, respectively.

As mentioned before, the parameters m and p represent the precisions of the inputs and outputs of neurons, respectively. In neural networks, the activation output of a neuron can be regarded as the input of the neuron connected to it. Therefore, the values of m and p can be considered equal (or approximately equal). In this way, our quantum neurons always use fewer quantum resources than Ref. [21], even exponentially reduce the costs when n is much larger than m and p.

IV. EXPERIMENTS

A. IBM Quantum Experience experiments

Nowadays, noisy intermediate-scale quantum (NISQ) technology has been available [39]. IBM Quantum Experience (IBMQ for short), a cloud-based quantum computer, has become a popular platform for quantum computing. We implement the discrete rectified linear unit (ReLU) activation function with different quantum circuits based on the platform.

Figures 5(a) and 5(b) show the results of running the two circuits in Fig. 1. Both the circuits are implemented with four qubits, where half the qubits are for inputs and the other half are for outputs. The first qubit of inputs (and outputs) is regarded as the sign of a number and the second represents the one-bit integer. Then the domain and codomain of discrete ReLU can be computed classically. The input states of the two circuits are initialized to superposition states with equal weights. Figure 5(c) presents an example of a nonlinear quantum neuron based on Fig. 4(a), and its running results are shown in Fig. 5(d).

Since the inputs and outputs of our typical circuits are integers, they can be written in a binary form precisely, and the errors are only caused by the quantum gate errors and readout errors. We obtain 100% accuracies on *ibmq_qasm_simulator* backend. As the gate errors and readout errors on real quantum hardware reduce the executed reliability of quantum circuits, the accuracies on *ibmq_rome* backend and *ibmq_santiago* backend are about 70% and 60%, respectively.

From the experimental results, we conclude that the two circuits in Fig. 1 have similar performance in the case of using the same number of qubits. The results on the simulator verify the feasibility of the circuits, specifically including the circuit of a typical quantum neuron with the ReLU activation function. The results on quantum computers illustrate that our specific circuits still have a positive performance on real quantum hardware in the NISQ era of quantum computing. Certainly, the accuracies on quantum computers can be improved with the circuit optimization theory and well performance of the real quantum hardware. We just show the mapping experiment of discrete ReLU. It can be extended to any type of function.

B. Numerical simulations of a quantum neuron

The performance of ANNs is significantly related to the selection of activation functions. In many scenarios, it is necessary to trade off both the advantages and disadvantages of different activation functions. There are some commonly used classical activation functions, such as sigmoid, hyperbolic tangent (Tanh), Gaussian error linear units (GELU), and so on. Figure 6 shows the results of simulating these activation functions based on the nonlinear quantum neuron proposed in Fig. 4(a), where the input is the value of $(x^i)^T w$. The neuron has two approximate calculations: one is the approximation of $(x^i)^T w$ and the other is the approximation of the activation function, which is computed by the classical 64-bit computer. These two approximations are carried out by the first and second registers, respectively. We stipulated that both of these two approximations are accurate to eight qubits, one qubit for representing the sign, three qubits for integers, and four qubits for decimals.

The results exhibit that the activation functions computed by the nonlinear quantum neuron can simulate classical activation functions with a few qubits. In the numerical

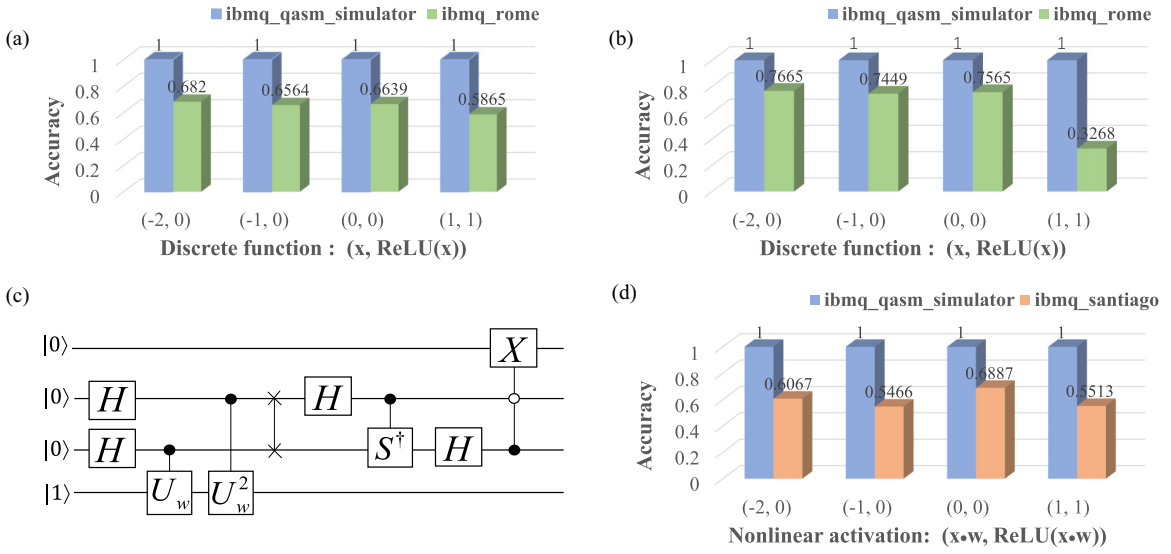

FIG. 5. Implementing discrete ReLU with different quantum circuits. All circuits are executed 8192 times (the maximum allowed) on IBM's quantum simulator and quantum computers. The accuracy indicates the proportion of the correct outputs of all outputs for given inputs. (a) The results of using four qubits to implement discrete ReLU with the circuit in Fig. 1(a). (b) The results of using four qubits to implement discrete ReLU with the circuit in Fig. 1(b). (c) An example of a typical quantum circuit for a nonlinear quantum neuron based on the Fig. 4(a) with four qubits, where the sign bit of the activation output is ignored due to ReLU's non-negative output. In this example, the input data is $x = 1$ and the $U_w = R_z(w/4)$, $w \in \{-2, -1, 0, 1\}$. (d) The running results of the circuit in Fig. 5(c).

simulations, we only consider the ideal case where the quantum phase estimation runs successfully and returns accurate results. So the approximate error is only caused by the insufficient number of output qubits. Actually, the errors would infinitely approach zero as the number of qubits increases.

C. Numerical simulation of a simple quantum neural network

As the proposed quantum neurons are proved to be able to simulate any nonlinear activation function, one of their common usages is similar to classical neurons. The quantum neurons of one layer are connecting to quantum neurons of the immediately preceding and immediately following layers.

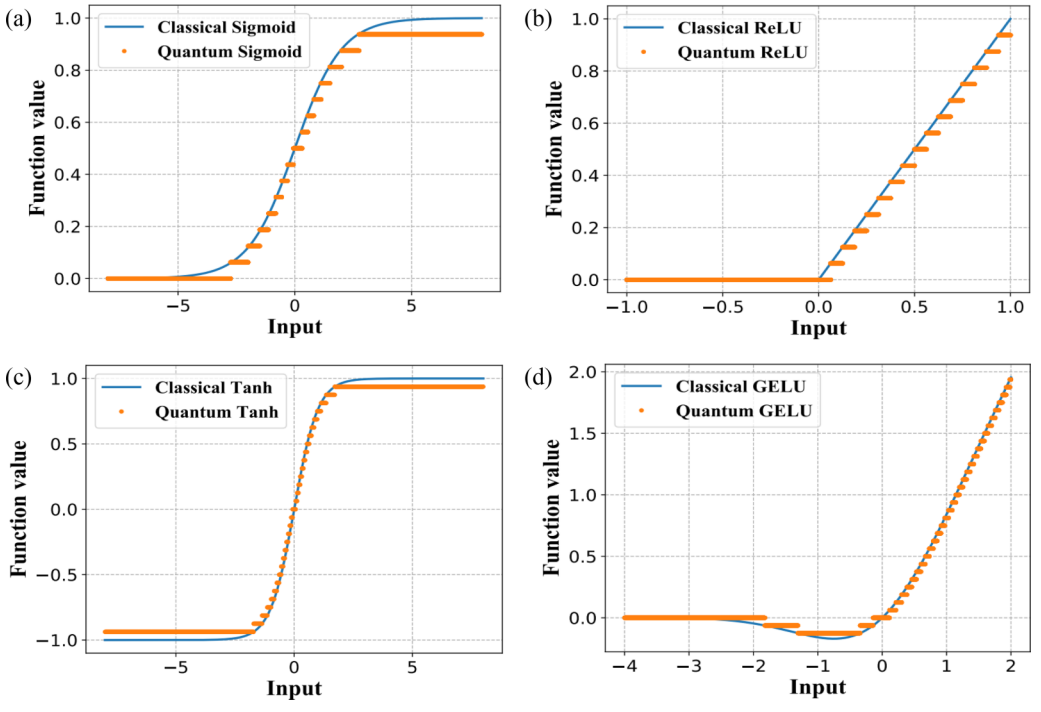

FIG. 6. Different types of activation functions: (a) Sigmoid, (b) ReLU, (c) Tanh, and (d) GELU.

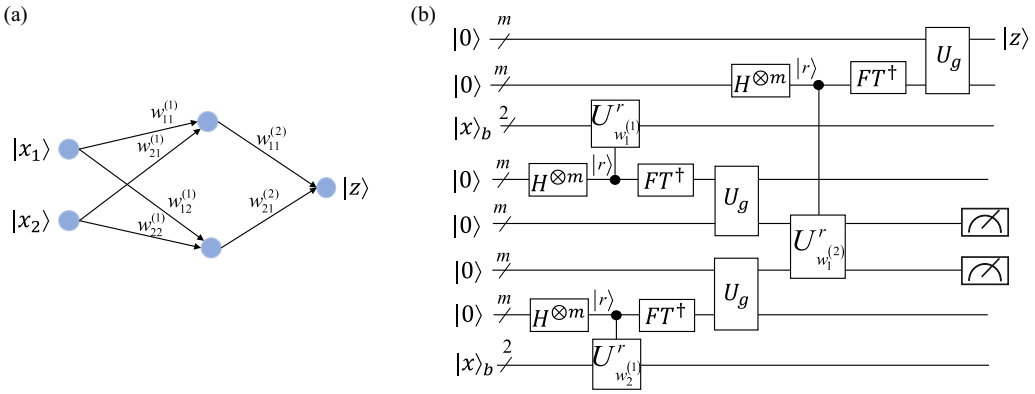

FIG. 7. Construction of a simple quantum feedforward neural network with the proposed nonlinear quantum neurons. Here the input states are $|x_1\rangle$ and $|x_2\rangle$, and the output state is $|z\rangle$. (a) Quantum feedforward neural network model. (b) The quantum feedforward neural network represented by a circuit, where $|x\rangle_b = |x_1\rangle \otimes |x_2\rangle$, $U_{w_i^{(1)}} = R_z(w_{i1}^{(1)}) \otimes R_z(w_{i2}^{(1)})$, $U_{w_i^{(2)}} = R_z(w_{i1}^{(2)}) \otimes R_z(w_{i2}^{(2)}) \otimes \cdots \otimes R_z(w_{im}^{(2)})$.

Figure 7 shows the construction of a quantum feedforward neural network (QFNN) with the three quantum neurons proposed in Fig. 4(a), where two neurons are for the hidden layer and the other one for the output layer. The QFNN has two inputs, which constitute the input layer, i.e., the zeroth layer, of the neural network. For $k = 1, 2$, the jth neuron in the kth layer and the ith neuron in the $(k-1)$st layer are connected by an edge $w_{ij}^{(k)}$, which belongs to the set $\{w_{11}^{(1)}, w_{12}^{(1)}, w_{21}^{(1)}, w_{22}^{(1)}, w_{11}^{(2)}, w_{12}^{(2)}\}$.

With the quantum sigmoid activation function implemented by the oracle U_g, the three-neuron QFNN was trained with backpropagation to solve the well-known XOR problem, which consists of four samples ($q = 4$). During the training process, we applied the batch gradient descent algorithm to minimize the following mean square loss function

$$\mathcal{L} = \frac{1}{q}\sum_{i=1}^{q}(d^i - z^i)^2, \tag{25}$$

where for the ith sample, d^i is the desired output and z^i is the result of performing measurement on the output $|z^i\rangle$ of the QFNN. The update rule for the weights is

$$w_{(l+1)} = w_{(l)} - \eta\frac{\partial\mathcal{L}}{\partial w_{(l)}}, \tag{26}$$

where $w_{(l+1)}$ is the weight after the $(l+1)$st step of the iteration process and η is an adjustable positive step length. The learning curve is presented in Fig. 8, from which we can see that the loss converges to about 0.126. It has been shown that the loss function and gradient can be calculated classically as the activation function is known. Furthermore, since the proposed QFNN is a parametrized quantum circuit, there will be exciting work to train the QFNN in a quantum way.

V. CONCLUSIONS AND DISCUSSIONS

In this paper, we have noted that there are various oracles to implement the same Boolean function. Even for the same oracle, the circuits are not unique. We have established two circuits (see Fig. 1) to implement the Boolean function f, which can be regarded as an approximation of the corresponding nonlinear function. There should be more quantum circuits to approximate nonlinear functions. Moreover, this paper has proposed a generalizable framework to implement nonlinear quantum neurons that fully realizes the combination of quantum computing and neural networks. We have presented two quantum neuron examples that can be used as a fundamental building block for QNNs. The quantum neuron examples have the advantages of processing massive or ultrahigh-dimensional data due to the quantum parallelism and entanglement. The quantum resources required to construct a single quantum neuron are the polynomial, in function of the input size. Both IBM Quantum Experience results and numerical simulations have illustrated the effectiveness of the proposed framework, which could map the existing classical neuron designed for classical computers to quantum circuits.

In addition to using the controlled-phase gates and the swap test, many other existing quantum computing models can apply to our framework to realize more kinds of quantum neurons. The most far-reaching generalization of the proposed framework is not restricted to implement nonlinear quantum neurons, it may have more applications in quantum machine learning [40–43]. In the future, we will connect multiple layers of our nonlinear quantum neurons to build a feedforward

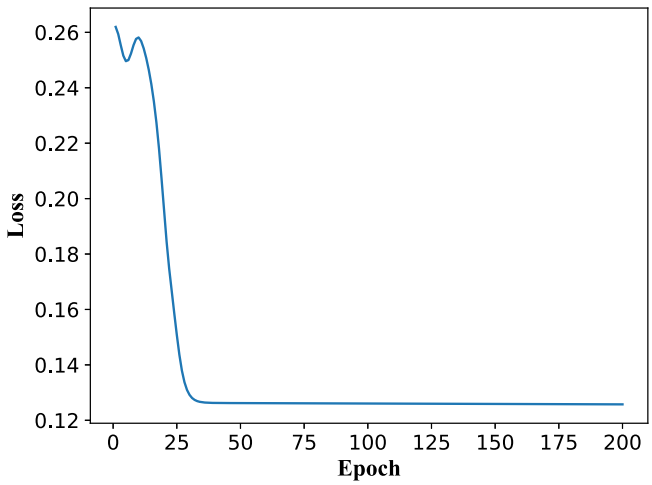

FIG. 8. The learning curve.

deep neural network, which could be fully trained on quantum computers.

ACKNOWLEDGMENTS

The authors acknowledge the support of IBM Quantum Experience for producing the experimental results. This work was supported partly by the National Key R&D Program of China under Grant 2018YFA0703800, the National Natural Science Foundation of China under Grants 61873317, 61873262, and 61733018, the Guangdong Basic and Applied Basic Research Foundation under Grant 2020A1515011375, and the Youth Innovation Promotion Association of the CAS.

[1] M. A. Nielsen, *Neural Networks and Deep Learning* (Determination Press, San Francisco, 2015), Vol. 2018.
[2] P. W. Shor, in *Proceedings 35th Annual Symposium on Foundations of Computer Science* (1994), pp. 124–134.
[3] L. K. Grover, Phys. Rev. Lett. **79**, 325 (1997).
[4] A. W. Harrow, A. Hassidim, and S. Lloyd, Phys. Rev. Lett. **103**, 150502 (2009).
[5] A. A. Ezhov and D. Ventura, *Quantum Neural Networks* (Physica-Verlag HD, Heidelberg, 2000), pp. 213–235.
[6] M. Schuld, I. Sinayskiy, and F. Petruccione, Quantum Inf. Process. **13**, 2567 (2014).
[7] S. Jeswal and S. Chakraverty, Arch. Comput. Methods Eng. **26**, 793 (2019).
[8] M. Schuld, I. Sinayskiy, and F. Petruccione, Phys. Lett. A **379**, 660 (2015).
[9] A. J. da Silva, T. B. Ludermir, and W. R. de Oliveira, Neural Networks **76**, 55 (2016).
[10] Y. Cao, G. G. Guerreschi, and A. Aspuru-Guzik, arXiv:1711.11240.
[11] K. H. Wan, O. Dahlsten, H. Kristjánsson, R. Gardner, and M. Kim, npj Quantum Inf. **3**, 1 (2017).
[12] H. Wei, Natural Sci. **10**, 99 (2018).
[13] J. Zhao, Y. H. Zhang, C. P. Shao, Y. C. Wu, G. C. Guo, and G. P. Guo, Phys. Rev. A **100**, 012334 (2019).
[14] F. Tacchino, C. Macchiavello, D. Gerace, and D. Bajoni, npj Quantum Inf. **5**, 1 (2019).
[15] I. Cong, S. Choi, and M. D. Lukin, Nat. Phys. **15**, 1273 (2019).
[16] C. Shao, Quantum Inf. Process. **19**, 102 (2020).
[17] M. Schuld, A. Bocharov, K. M. Svore, and N. Wiebe, Phys. Rev. A **101**, 032308 (2020).
[18] A. Paetznick and K. M. Svore, Quantum Info. Comput. **14**, 1277 (2014).
[19] N. Wiebe and M. Roetteler, Quantum Info. Comput. **16**, 134 (2016).
[20] A. Bocharov, M. Roetteler, and K. M. Svore, Phys. Rev. Lett. **114**, 080502 (2015).
[21] F. M. de Paula Neto, T. B. Ludermir, W. R. de Oliveira, and A. J. da Silva, IEEE Trans. Neural Netw. Learn. Syst. **31**, 3741 (2020).
[22] E. C. Behrman, L. Nash, J. E. Steck, V. Chandrashekar, and S. R. Skinner, Inf. Sci. (NY) **128**, 257 (2000).
[23] S. Kak, Inf. Sci. (NY) **83**, 143 (1995).
[24] G. Benenti, G. Casati, and G. Strini, *Principles of Quantum Computation and Information, Volume I: Basic Concepts* (World Scientific, Singapore, 2004).
[25] A. Berthiaume and G. Brassard, J. Mod. Opt. **41**, 2521 (1994).
[26] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, SIAM J. Comput. **26**, 1510 (1997).
[27] E. Kashefi, A. Kent, V. Vedral, and K. Banaszek, Phys. Rev. A **65**, 050304(R) (2002).
[28] N. Johansson and J.-Å. Larsson, Entropy **21**, 800 (2019).
[29] M. Schuld and F. Petruccione, *Supervised Learning with Quantum Computers* (Springer, Cham, Switzerland, 2018).
[30] V. Giovannetti, S. Lloyd, and L. Maccone, Phys. Rev. Lett. **100**, 160501 (2008).
[31] S. Lloyd, M. Mohseni, and P. Rebentrost, Nat. Phys. **10**, 631 (2014).
[32] X. Zhang, X. Zhang, and Z. Xue, Sci. Rep. **6**, 24910 (2016).
[33] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, Phys. Rev. A **52**, 3457 (1995).
[34] M. Möttönen, J. J. Vartiainen, V. Bergholm, and M. M. Salomaa, Phys. Rev. Lett. **93**, 130502 (2004).
[35] S. S. Bullock and I. L. Markov, Quantum Inf. Comput. **4**, 27 (2004).
[36] J. Welch, D. Greenbaum, S. Mostame, and A. Aspuru-Guzik, New J. Phys. **16**, 033040 (2014).
[37] M. Houshmand, M. S. Zamani, M. Sedighi, and M. Arabzadeh, ACM J. Emerg. Technol. Comput. Syst. **11**, 1 (2014).
[38] M. A. Nielsen and I. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, 2010).
[39] J. Preskill, Quantum **2**, 79 (2018).
[40] D. Dong, X. Xing, H. Ma, C. Chen, Z. Liu, and H. Rabitz, IEEE Trans. Cybernetics **50**, 3581 (2020).
[41] W. Cui, and D. Dong, IEEE Trans. Control Syst. Tech. **27**, 2499 (2019).
[42] R. B. Wu, H. Ding, D. Dong, and X. Wang, Phys. Rev. A **99**, 042327 (2019).
[43] C. Chen, D. Dong, R. Long, I. R. Petersen, and H. A. Rabitz, Phys. Rev. A **89**, 023402 (2014).