Parametrized Constant-Depth Quantum Neuron

Jonathan H. A. de Carvalho and Fernando M. de Paula Neto

Abstract—Quantum computing has been revolutionizing the development of algorithms. However, only noisy intermediate-scale quantum devices are available currently, which imposes several restrictions on the circuit implementation of quantum algorithms. In this article, we propose a framework that builds quantum neurons based on kernel machines, where the quantum neurons differ from each other by their feature space mappings. Besides contemplating previous quantum neurons, our generalized framework has the capacity to instantiate other feature mappings that allow us to solve real problems better. Under that framework, we present a neuron that applies a tensor-product feature mapping to an exponentially larger space. The proposed neuron is implemented by a circuit of constant depth with a linear number of elementary single-qubit gates. The previous quantum neuron applies a phase-based feature mapping with an exponentially expensive circuit implementation, even using multiqubit gates. Additionally, the proposed neuron has parameters that can change its activation function shape. Here, we show the activation function shape of each quantum neuron. It turns out that parametrization allows the proposed neuron to optimally fit underlying patterns that the existing neuron cannot fit, as demonstrated in the nonlinear toy classification problems addressed here. The feasibility of those quantum neuron solutions is also contemplated in the demonstration through executions on a quantum simulator. Finally, we compare those kernel-based quantum neurons in the problem of handwritten digit recognition, where this work provides a quantum neuron with improved discriminative abilities. As a consequence, the generalized framework of quantum neurons can contribute toward practical quantum advantage.

Index Terms—Constant-depth quantum circuit, kernel machine, parametrized activation function, quantum computing, quantum neuron framework.

I. INTRODUCTION

QUANTUM computing [1] is expected to achieve the so-called supremacy over classical computing, which will allow us to solve previously intractable problems in a reasonable time [2]. Specifically, that quantum advantage also allows the development of more efficient neural networks [3]. Quantum neurons can implement arbitrary nonlinear functions while taking advantage of quantum properties like superposition and entanglement [4]. However, fault-tolerant quantum computations require encoding the information in many redundant qubits so that error correction codes can be employed [5].

Currently, only noisy intermediate-scale quantum devices are available, so there are not enough resources to protect the system by quantum error correction [6]. Depth, width, and number of operations become critical factors to successfully execute quantum circuits on the current hardware. Thus, aspects like state preparation, oracle expansion, connectivity, circuit rewriting, decoherence, gate infidelity, and measurement errors can compromise the computation [7].

Despite significant efforts at the quantum compiler level [8], [9], recent quantum algorithms are developed considering those present restrictions [10], [11], including the development of quantum neural networks [12], [13], [14], [15], [16], [17], [18], [19]. Particularly, this work focuses on quantum perceptrons implemented efficiently on actual quantum devices. Tacchino et al. [12] took the first step by proposing a scheme that computes the inner product between binary-valued vectors. After, Mangini et al. [13] reformulated that scheme to accept continuous-valued vectors instead of only binary-valued ones. Both schemes have an exponential advantage in terms of circuit width, although the circuit depth and the number of operations grow exponentially. In this work, we propose a framework of quantum neurons where those previous schemes [12], [13] are particular cases. Based on that framework, each quantum neuron implements a kernel machine with a nondeterministic activation function. That activation function depends only on the kernel trick each quantum neuron applies. Thus, the quantum neurons differ from each other by their feature space mappings. It makes room to instantiate other quantum neurons under that generalized framework, including for actual quantum devices.

Generally, quantum kernel methods are used to estimate inner products for classical models in a hybrid setup. The advantage emerges from the fact that the inner products are estimated from classically intractable feature mappings. Those inner products can be estimated by the standard swap-test [20] or, more recently, by quantum kernel estimators [21], [22]. Speedups in classical models can be also obtained by classical sampling techniques inspired by quantum models that generate the kernel matrix of inner products [23].

The framework of quantum neurons proposed here computes the inner product between an input vector and a weight vector, and then explicitly extracts such information to an ancillary qubit. Measuring the ancilla gives a standalone fully-quantum classifier based on kernel machines. If the ancilla is not measured, the inner product can be propagated

Manuscript received 12 April 2022; revised 10 February 2023; accepted 20 June 2023. Date of publication 12 July 2023; date of current version 30 October 2023. This work was supported by the Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco (FACEPE) under Grant IBPG-0084-1.03/20 and Grant APQ-1110-1.03/21. (Corresponding author: Jonathan H. A. de Carvalho.)

The authors are with the Centro de Informática, Universidade Federal de Pernambuco, Recife 50740-560, Brazil (e-mail: jhac@cin.ufpe.br; fernando@cin.ufpe.br).

Digital Object Identifier 10.1109/TNNLS.2023.3290535
forward to other quantum neurons, which gives a quantum neural network [14]. Finally, the proposed framework of quantum neurons is expected to achieve practical quantum advantage, as is expected with quantum kernel estimators [24].

Based on that framework, we also propose a quantum neuron of constant depth, i.e., its circuit depth is independent of the input size. Constant-depth quantum circuits can demonstrate quantum advantage [25], [26] and benefit from error mitigation techniques [27], [28], [29]. The proposed quantum neuron implements local feature mappings [30] by taking advantage of qubit encoding [15], [19]. We demonstrate that encoding strategy actually implements a tensor-product feature mapping to an exponentially larger space, which improves the separating capacity [31].

We further improve that neuron capacity by including two parameters in its activation function. That parametrization can change the activation function shape of the proposed quantum neuron to fit different underlying patterns with no additional cost in the circuit implementation. Therefore, we propose a flexible quantum neuron of constant depth implemented with a linear number of elementary single-qubit gates. The existing quantum neuron [13] is inflexible and exponentially expensive, even with multiqubit gates in its circuit implementation.

Then, we proceed to a visual study that relates some interactions between the input and weight vectors in the original space with the respective neuron outputs in the feature space. Those activation function shapes reveal the problem structures that each quantum neuron can solve. By comparing the best solutions of each quantum neuron in toy classification problems, we demonstrate that parametrization can change the activation function shape to optimally fit all cases, even those that the existing quantum neuron cannot fit. We also demonstrate the feasibility of those quantum neuron solutions through a proof-of-concept experiment on a quantum simulator. A conclusive experiment about the neuron capabilities is finally conducted here as we address the recognition of handwritten digits and compare the results against quantum neurons based on another architecture. Those results in real-life problems attest that the proposed quantum neuron, especially due to the activation function parametrization, has better discriminative power than the previous proposals of quantum neurons.

This article is organized as follows. First, Section II formalizes the quantum neuron framework based on kernel machines. The reader that is not familiar with the basic concepts of quantum computing should refer to [1] and [32]. Under the proposed framework, Section III presents the parametrized quantum neuron of constant circuit depth. The activation function shapes of the quantum neurons are presented in Section IV. Then, Section V compares the quantum neurons in solving some toy classification problems. Section VI addresses the task of handwritten digit recognition, including comparisons with another model of quantum neurons. Final remarks and future directions are discussed in Section VII.

II. QUANTUM NEURON FRAMEWORK

The classical neuron model basically consists of two steps. First, the inner product between the input vector \( \vec{i} \) and the weight vector \( \vec{w} \) is computed, and then that inner product \( \sum_j w_j i_j \) is passed to an activation function \( \varphi(\cdot) \) to define the neuron output \( y \) [33]. Based on that model, one can construct different classical neurons by only changing the activation function that processes the obtained inner product. Examples of common activation functions are the Heaviside function and the logistic sigmoid function, which are presented in the following equations, respectively,

\[
\varphi(\vec{w} \cdot \vec{i}) = \begin{cases} 
1, & \text{if } \vec{w} \cdot \vec{i} \geq 0 \\
0, & \text{if } \vec{w} \cdot \vec{i} < 0 
\end{cases}
\]

\[
\varphi(\vec{w} \cdot \vec{i}) = \frac{1}{1 + e^{-(\vec{w} \cdot \vec{i})}}.
\]

Inspired by that classical neuron model, Tacchino et al. [12] proposed a scheme for actual quantum devices that encodes input and weight vectors, computes the inner product between them, and finally extracts the activation function output. Here, we generalize that scheme to a framework of quantum neurons based on kernel machines. Given two vectors \( \vec{\phi} \) and \( \vec{\theta} \) in an input space, a kernel machine \( k(\cdot, \cdot) \) maps those vectors to other two vectors \( \vec{w} \) and \( \vec{i} \) in a feature space by a nonlinear transformation \( \Phi(\cdot) \), and then computes the inner product between the transformed vectors, i.e., \( k(\vec{\phi}, \vec{\theta}) = \Phi(\vec{\phi})^T \Phi(\vec{\theta}) = \vec{w} \cdot \vec{i} \) [34]. The quantum neurons constructed by the framework are deeply related to that kernel trick, which paves the way to the quantum neuron that we propose in this work.

The quantum neurons implement kernel methods because the given \( m \)-dimensional classical vectors \( \vec{\theta} \) and \( \vec{\phi} \) are first mapped to \( N \)-dimensional quantum vectors \( \vec{i} = \Phi(\vec{\theta}) \) and \( \vec{w} = \Phi(\vec{\phi}) \). Those quantum vectors can be directly encoded in legitimate quantum states in the following way:

\[
|\psi_i\rangle = \sum_{j=0}^{N-1} i_j |j\rangle \quad \text{and} \quad |\psi_w\rangle = \sum_{j=0}^{N-1} w_j |j\rangle.
\]

Then, the inner product \( \vec{w}^* \cdot \vec{i} = \langle \psi_w | \psi_i \rangle \) is computed by the quantum neurons. The final neuron output is given by the nondeterministic activation function presented in the following equation:

\[
\varphi(\vec{w}^* \cdot \vec{i}) = \begin{cases} 
1, & \text{with probability } |\vec{w}^* \cdot \vec{i}|^2 \\
0, & \text{with probability } 1 - |\vec{w}^* \cdot \vec{i}|^2.
\end{cases}
\]

In this way, the quantum activation function depends on the inner product \( \vec{w}^* \cdot \vec{i} \) that in turn depends on the feature space mapping \( \Phi(\cdot) \). Based on that framework, one can realize different quantum neurons, including on actual quantum computers, by only changing the mapping \( \Phi(\cdot) \) that each quantum neuron implements.

Fig. 1 shows the circuit implementation of the framework of quantum neurons based on kernel machines. Specifically, the quantum neurons differ from each other by the quantum operators \( E \) and \( D \), which depend on the classical vectors \( \vec{\theta} \) and \( \vec{\phi} \), respectively. The operator \( E \) maps \( \vec{\theta} \) to \( \vec{i} \) and then encodes it in \( |\psi_i\rangle \) from \( |+\rangle^\otimes n \), where \( |+\rangle = (1/\sqrt{2})(|0\rangle + |1\rangle) \) and \( n \) is the number of encoding qubits. Thus,

\[
E(\vec{\theta}) |+\rangle^\otimes n = |\psi_i\rangle.
\]
The operator $D$ maps $\phi$ to $\vec{w}$ and then decodes it from $|\psi_w\rangle$ to $|+\rangle^\otimes n$. Thus,

$$D(\phi) |\psi_w\rangle = |+\rangle^\otimes n.$$  

At a higher level of abstraction, the operator $U_i$ encodes $|\psi_i\rangle$ from a blank register $|0\rangle^\otimes n$

$$U_i |0\rangle^\otimes n = E(\theta) H^\otimes n |0\rangle^\otimes n = E(\theta) |+\rangle^\otimes n = |\psi_i\rangle.$$  

On the other hand, $U_w$ decodes $|\psi_w\rangle$ to $|1\rangle^\otimes n$

$$U_w |\psi_w\rangle = X^\otimes n H^\otimes n D(\phi) |\psi_w\rangle = X^\otimes n H^\otimes n |+\rangle^\otimes n = X^\otimes n |0\rangle^\otimes n = |1\rangle^\otimes n.$$  

The operator $D(\phi)$ and then $U_w$ are defined according to the action on $|\psi_w\rangle$, but those operators are applied in $|\psi_i\rangle$ in the circuit flow. Applying $U_w$ to $|\psi_i\rangle$ actually computes the inner product between $\vec{i}$ and $\vec{w}$ by taking advantage of an interesting property of unitary operators. Unitary operators preserve the space geometry, which includes inner products [32]. If $A$ is a unitary operator, the inner product between a quantum state $|\psi\rangle$ and another quantum state $|\phi\rangle$ is equal to the inner product between $A |\psi\rangle$ and $A |\phi\rangle$, i.e., $\langle \psi |\phi \rangle = \langle A^\dagger |A\psi| A^\dagger |A\phi \rangle$. Let $A = U_w$, $|\psi\rangle = |\psi_i\rangle$, and $|\phi\rangle = |\psi_w\rangle$. We can rewrite the expression as

$$\langle \psi_w |\psi_i \rangle = \langle \psi_w U_i U_w |\psi_i \rangle = \langle |0\rangle^\otimes n |U_w |\psi_i \rangle.$$  

Thus, the projection of $U_w |\psi_i\rangle$ on the basis state $|1\rangle^\otimes n$ stores the inner product $\langle \psi_w |\psi_i \rangle$. In other words, that inner product is stored in the component of $|1\rangle^\otimes n$ into the superposition of $U_w |\psi_i\rangle = |\psi_i,w\rangle$. In this way

$$|\psi_i,w\rangle = c_0 |0\rangle + c_1 |1\rangle + \cdots + c_{N-2} |N-2\rangle + \langle \psi_w |\psi_i \rangle |N-1\rangle.$$  

To extract such information, a multicontrolled NOT gate is applied targeting an ancillary qubit. Note that only the basis state $|N-1\rangle$ in the superposition flips the ancilla to the state $|1\rangle$. Consequently, the final quantum state $|\psi_f\rangle$ is

$$|\psi_f\rangle = c_0 |0\rangle + c_1 |1\rangle + \cdots + c_{N-2} |N-2\rangle + \langle \psi_w |\psi_i \rangle |N-1\rangle |1\rangle.$$  

Since $|\psi_w |\psi_i \rangle = \vec{w}^* \cdot \vec{i}$, measuring that ancillary qubit finally gives the activation function already presented in (3).

Therefore, that quantum neuron framework applies the kernel trick and activates following a nondeterministic function of the computed inner product, which was to be shown. To instantiate a quantum neuron, one only needs to realize a feature mapping $\Phi(\cdot)$ by means of the operators $E$ and $D$.

### A. Binary-Valued Quantum Neuron (BVQN)

That generalized framework can be first instantiated in the BVQN [12]. Given $m$-dimensional classical vectors $\theta = (\theta_0, \theta_1, \ldots , \theta_{m-2}, \theta_{m-1})$ and $\phi = (\phi_0, \phi_1, \ldots , \phi_{m-2}, \phi_{m-1})$, where $\theta_j, \phi_j \in \{-1, 1\}$, the BVQN feature mapping simply normalizes those classical vectors, which leads to the following $N$-dimensional quantum vectors, where $N = m$:

$$\vec{i} = \Phi(\theta) = \frac{1}{\sqrt{N}} (\theta_0, \theta_1, \ldots , \theta_{m-2}, \theta_{m-1})$$  

$$\vec{w} = \Phi(\phi) = \frac{1}{\sqrt{N}} (\phi_0, \phi_1, \ldots , \phi_{m-2}, \phi_{m-1}).$$  

Those quantum vectors live in the Hilbert space spanned by $n$ qubits, where $N = 2^n$. As $N$ equals $m$, the BVQN has an exponential advantage in storing the information.

Note that $|+\rangle^\otimes n = (1/\sqrt{N}) \sum_{j=0}^{N-1} |j\rangle$, so $E(\theta)$ only needs to flip the sign of the basis states $|j\rangle$ where $\theta_j = -1$ to encode $\vec{i}$ in $|\psi_i\rangle$. Similarly, $D(\phi)$ only needs to cancel the sign of the basis states $|j\rangle$ where $\phi_j = -1$ to decode $\vec{w}$ from $|\psi_w\rangle$. Those operations are accomplished by applying sign-flip blocks one by one or, more efficiently, by applying the hypergraph states generation subroutine [12], [35]. However, those two strategies are exponentially expensive in terms of circuit depth. Finally, the BVQN fires with the probability presented in the following equation:

$$|\vec{w}^* \cdot \vec{i}|^2 = \frac{1}{N} \sum_{j=0}^{m-1} \theta_j \phi_j.$$  

### B. Continuous-Valued Quantum Neuron (CVQN)

A first alternative to accepting real vectors instead of only binary ones is to instantiate the framework in the CVQN [13], which implements a phase-based feature mapping. That mapping encodes each classical component in the phase of a complex number written in the exponential form with modulus $1/\sqrt{N}$. Thus, the classical vectors are mapped to quantum vectors of equal size in the following manner:

$$\vec{i} = \Phi(\theta) = \frac{1}{\sqrt{N}} (e^{i\theta_0}, e^{i\theta_1}, \ldots , e^{i\theta_{m-2}}, e^{i\theta_{m-1}})$$  

$$\vec{w} = \Phi(\phi) = \frac{1}{\sqrt{N}} (e^{i\phi_0}, e^{i\phi_1}, \ldots , e^{i\phi_{m-2}}, e^{i\phi_{m-1}}).$$  

Periodicity in that phase-based scheme and the square modulus in the activation function make the CVQN equally recognize values that are actually different. To really distinguish such values, the classical vectors are scaled to the interval $[0, \pi/2]$. Thus, the feature mapping is applied to classical vectors $\theta$ and $\phi$, where $\theta_j, \phi_j \in [0, \pi/2]$. The $N$-dimensional quantum vectors, where $N = n$, are described by $n$ qubits, where $N = 2^n$, which represents an exponential advantage again in information storage.
To implement $E(\theta)$ and $D(\phi)$, the blocks that flip the sign in the BVQN are used here to encode each $e^{i\theta_j}$ and to decode each $e^{i\phi_j}$, respectively, by applying multicontrolled phase gates instead of multicontrolled $Z$ gates. The phase gate $P(\lambda)$ is represented in the matrix form by $\begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{pmatrix}$. As sign-flip blocks, phase shift blocks are also exponentially expensive in terms of circuit depth. Finally, the CVQN fires with the probability presented in the following equation:

$$|\bar{w}^* \cdot \vec{i}|^2 = \frac{1}{N} \sum_{j=0}^{m-1} e^{i(\theta_j - \phi_j)}^2.$$  

(5)

### III. Proposed Quantum Neuron

Encoding data in the amplitudes of quantum states provides a gain in storage, but the circuit depth grows exponentially. The implication is that amplitude encoding requires a few qubits that would be controlled for a long time and would suffer many manipulations. However, in the current era of quantum computing, the circuit depth and the number of operations really matter due to the poor quality of the qubits available in the quantum devices.

Here, we propose to use a qubit encoding scheme under that framework of quantum neurons, differently from the amplitude encoding used by the BVQN and the CVQN. In this way, an $m$-dimensional classical vector is encoded in $m$ qubits, which represents less efficient storage, but the state preparation is definitely efficient in terms of circuit depth as each qubit only requires a single rotation to encode the data [15], [19]. That is, qubit encoding requires more qubits that would be controlled for less time and would suffer fewer manipulations.

In light of the present circuit requirements, qubit encoding emerges as a suitable encoding strategy to explore. By taking advantage of the qubit encoding efficiency, we can propose the constant-depth quantum neuron (CDQN).

Instead of a qubit encoding from rotations about the $y$-axis [15], [19], we apply that phase shift gate $P(\lambda)$ already used in amplitude encoding previously [13]. Thus, the classical input vector $\theta$ is directly encoded by the operator $E$ as follows:

$$E(\theta) = \otimes_{j=0}^{m-1} P(\theta_j).$$

Similarly, the classical weight vector $\phi$ is directly encoded by the operator $D$ in the following way:

$$D(\phi) = \otimes_{j=0}^{m-1} P(-\phi_j).$$

As a result, $U_i$ comes down to

$$U_i = \otimes_{j=0}^{m-1} P(\theta_j) H^{\otimes m}.$$

In turn, $U_w$ comes down to

$$U_w = X^{\otimes m} H^{\otimes m} \otimes_{j=0}^{m-1} P(-\phi_j).$$

Therefore, with a circuit depth independent of the input size, the proposed quantum neuron computes the inner product between two vectors that are transformed by a tensor product of local feature mappings [30]. Those classical vectors are also preprocessed to the interval $[0, \pi/2]$, as for the CVQN.

Applying that $E(\theta)$ in a basis state $|s\rangle$ produces the following state:

$$E(\theta) |s\rangle = \prod_{j=0}^{m-1} (e^{i\theta_j})^{b_{j,m}} |s\rangle$$

where $b_{j,m}$ is the $j$th bit in the binary representation of the integer value $s$ with $m$ bits. In this way, $b_{j,m}$ selects the phase shifts to be applied. For example, with $m = 4$, applying $E(\theta)$ in $|5\rangle$ produces $e^{i\theta_3} e^{i\theta_2} |5\rangle$, using the big-endian convention. Applying $E(\theta)$ in $|+\rangle^{\otimes m}$ stores a conditioned product of $e^{i\theta_j}$ on the amplitude of each possible state $|s\rangle$, which represents the following feature mapping, where $N = 2^m$:

$$\vec{i} = \Phi(\theta) = \frac{1}{\sqrt{N}} \prod_{j=0}^{m-1} (e^{i\theta_j})^{b_{j,0,m}}, \prod_{j=0}^{m-1} (e^{i\theta_j})^{b_{j,1,m}}, \ldots,$$

$$\prod_{j=0}^{m-1} (e^{i\theta_j})^{b_{j,N-1,m}}, \prod_{j=0}^{m-1} (e^{i\theta_j})^{b_{j,N-1,m}}$$

and

$$\bar{w} = \Phi(\phi) = \frac{1}{\sqrt{N}} \prod_{j=0}^{m-1} (e^{i\phi_j})^{b_{j,0,m}}, \prod_{j=0}^{m-1} (e^{i\phi_j})^{b_{j,1,m}}, \ldots,$$

$$\prod_{j=0}^{m-1} (e^{i\phi_j})^{b_{j,N-1,m}}, \prod_{j=0}^{m-1} (e^{i\phi_j})^{b_{j,N-1,m}}.$$

Different from the previous quantum neurons, the CDQN maps $m$-dimensional classical vectors to $N$-dimensional quantum vectors with $N > m$. Specifically, the CDQN maps the vectors to an exponentially larger space. Thus, the CDQN is a kernel method that takes advantage of Cover’s theorem [31], which states that mapping to feature spaces of high dimensionality improves the separating capacity. Nonlinearly separable problems are more likely to become linearly separable in a high-dimensional space. Finally, the CDQN fires with the probability presented in the following equation:

$$|\bar{w}^* \cdot \vec{i}|^2 = \frac{1}{N} \sum_{j=0}^{N-1} \left| \prod_{j=0}^{m-1} (e^{i(\theta_j - \phi_j)})^{b_{j,m}} \right|^2.$$  

(6)

We can simplify the relation presented in (6) by observing that it is actually a combination of $m$ local inner products. Each local inner product is computed with respect to a local feature mapping that maps a $\theta_j$ to $\vec{i}_j = (1/\sqrt{2})(1, e^{i\theta_j})$ and a $\phi_j$ to $\vec{w}_j = (1/\sqrt{2})(1, e^{i\phi_j})$. As a result, the relation presented in (6) is equivalent to a product of $m$ local inner products in the form $\bar{w}^*_j \cdot i_j = (1/2)(1 + e^{i(\theta_j - \phi_j)})$. Therefore, the CDQN fires with a probability that can be simplified as presented in the following equation:

$$|\bar{w}^* \cdot \vec{i}|^2 = \frac{1}{N} \prod_{j=0}^{m-1} \left| 1 + e^{i(\theta_j - \phi_j)} \right|^2.$$  

(7)

Building on the CDQN, we also propose to parametrize its activation function by a multiplicative factor $\tau$ and an additive constant $\delta$. Those parameters act on the componentwise differences between the classical input and weight
The CVQN, the orange curve represents the CDQN, and the blue curve represents the PCDQN.

As can be seen, the CVQN grows linearly with the input size $m$ in terms of depth and number of operations. Since $m = 2^n$ for the CVQN, that quantum neuron grows exponentially with the number of qubits $n$. On the other hand, the CDQN and the PCDQN have a constant depth and a lower number of operations. Additionally, the CDQN and the PCDQN are efficiently implemented with elementary single-qubit gates only, while the CVQN is expensive, even using complex phase shift blocks.

IV. Activation Function Shapes

Classical neuron outputs depend on some interaction between the input and weight vectors. For example, the larger the inner product between the vectors, the larger the output of the sigmoid activation function. As another example, the larger the Euclidean distance between the vectors, the smaller the output of the radial basis activation function. On the other hand, the quantum neuron outputs depend on the inner product between the input and weight vectors in the feature space. In this work, we study the quantum neuron outputs as a function of some relations between the original vectors, which can be seen as an indirect way to visualize quantum activation shapes.

Then, we defined a set of 2-D input vectors by splitting the interval $[0, \pi/2]$ into ten equidistant values and taking all possible combinations, which gives an input set of one hundred vectors. The weight vector is fixed here as $(\pi/2, \pi/2)$. Thus, we compute the classical relations and the corresponding quantum neuron outputs for each input vector with respect to that weight vector of reference. Input vectors that give the same measure to that reference vector do not necessarily give the same neuron outputs. Specifically, we considered the following metrics provided by scikit-learn [37]: Manhattan distance, Euclidean distance, linear kernel, polynomial kernel, RBF kernel, and sigmoid kernel.

Fig. 4 contrasts the CVQN activation with the CDQN activation as functions of the Manhattan distance, the Euclidean distance, and the linear kernel. The CVQN activation presents the same qualitative behavior in all cases but with a shift to the right for the Euclidean distance, as can be seen in Fig. 4(a)–(c). In all cases, high values of activation are generated for the input vectors with extreme measures. Moving away from the extremes, the measures generate smaller and smaller values of activation in addition to the previous high values. Regarding the CDQN, its activation monotonically decreases as the Manhattan and Euclidean distances increase, as shown in Fig. 4(d) and (e), and monotonically increases as the linear kernel also increases, as shown in Fig. 4(f).

In a similar way, Fig. 5 contrasts the CVQN activation with the CDQN activation as functions of the polynomial kernel, the RBF kernel, and the sigmoid kernel. The CVQN activation presents relatively the same qualitative behavior already presented in Fig. 4 but with a shift to the left for the polynomial and RBF kernels, as shown in Fig. 5(a) and (b), and a considerable shift to the right for the sigmoid kernel, as shown in Fig. 5(c). The CDQN activation presents relatively the same qualitative behavior already presented in Fig. 4(f) but with shapes that suggest a logarithmic growth.
in Fig. 5(d), a linear growth in Fig. 5(e), and an exponential growth in Fig. 5(f).

Therefore, the CVQN and the CDQN can solve classification problems where those activation shapes fit correctly. It turns out the nature of each quantum neuron is preserved for any of those metrics. Thus, we follow the analysis in this work using only the Euclidean distance due to its intuitiveness.

Fig. 6 shows the PCDQN activation as a function of the Euclidean distance between the original input and weight vectors for (a) \( \tau = 1/4 \) and \( \delta = \pi/2 \), (b) \( \tau = 1/2 \) and \( \delta = \pi/4 \), (c) \( \tau = 1 \) and \( \delta = \pi/4 \), (d) \( \tau = 1 \) and \( \delta = 3\pi/2 \), (e) \( \tau = 2 \) and \( \delta = 5\pi/4 \), and (f) \( \tau = 4 \) and \( \delta = 0 \).

Fig. 7 shows the PCDQN activation as a function of the Euclidean distance between the original input and weight vectors for (a) \( \tau = 1/4 \) and \( \delta = \pi/2 \), (b) \( \tau = 1/2 \) and \( \delta = \pi/4 \), (c) \( \tau = 1 \) and \( \delta = \pi/4 \), (d) \( \tau = 1 \) and \( \delta = 3\pi/2 \), (e) \( \tau = 2 \) and \( \delta = 5\pi/4 \), and (f) \( \tau = 4 \) and \( \delta = 0 \).

The demonstration consists of solving toy problems, followed by quantitative and qualitative comparisons of the best solutions of the quantum neurons. On one hand, quantitatively, the solutions are compared here by a classification metric. On the other hand, qualitatively, the solutions are compared here by the effectiveness of the activation function shapes applied to the problems. Finally, we demonstrate the feasibility of those numerical solutions by executing the quantum neuron circuits on a quantum simulator.

A. Datasets

The three datasets used in this demonstration are depicted in Fig. 7(a)–(c). Here, the positive class is represented by the black blobs, while the negative class is represented by the red blobs, although there is no previous indication of that labeling. Thus, we decide to swap the labels, deriving three more classification problems where the old negative classes become the new positive classes and vice versa, as depicted in Fig. 7(d)–(f). Note that a line is not able to separate the classes of any of those 2-D binary classification problems. The two attributes of each of those artificial and toy problems are merely coordinates, \( \theta_0 \) and \( \theta_1 \), that are defined in a way that visually organizes the data in a diagonal, in circles, or in a square, respectively. It is worth mentioning that \( \theta_0 \) and \( \theta_1 \) are scaled to the interval \([0, \pi/2]\). We discuss those classification problems and how to generate them in the following.

We refer to Fig. 7(a) and (d) as the diagonal blobs due to the formed structure. The difference is that the target is the center blobs in Fig. 7(a), while the target is the corner blobs in Fig. 7(d). That dataset is provided by the scikit-learn make_blobs function, letting centers be \([-5, 0, (5, 5, 8)]\), cluster_std be 1.2, and random_state be 0. Each class has 50 samples. Alternatively, the target is the inner circle in Fig. 7(b), while the target is the outer circle in Fig. 7(e). Those concentric circles are provided by the scikit-learn make_circles function, letting noise be 0.05, factor be 0.4, and random_state be 0. Each circle has 50 samples. Finally, the blobs in the coordinates \([0, 0], (0, \pi/2), (\pi/2, 0), (\pi/2, \pi/2)\] form a square, where the target is like the EXCLUSIVE OR (XOR) problem in Fig. 7(c), while the target is like the NOT EXCLUSIVE OR (NXOR) problem in Fig. 7(f).
of $(\tau, \delta)$ 10,000-vector weight space. Then, we choose the first values targeting the class Center. The CVQN also solved the NXOR and the CDQN perfectly solved the diagonal blobs when neuron achieved for each classification problem. The space of parameters is formed here by all combinations $(\tau, \delta)$ that maximize the AUC ROC together in the nested search. Actually, the combination $(1, 0)$ is not allowed since it produces the CDQN. Consequently, there are 34 allowed combinations to search. Those combinations explore a variety of lengths and shifts along the activation function period, which is $2\pi$. For each combination of $\tau$ and $\delta$, we search that 10,000-vector weight space. Then, we choose the first values of $(\tau, \delta)$ and $(\phi_0, \phi_1)$ that maximize the AUC ROC together in the nested search.

B. Search for Weights and Parameters

In this demonstration, we conduct a grid search for the best weight vector of each quantum neuron for each classification problem. The space of weight vectors is formed here by all vectors $(\phi_0, \phi_1)$ where each component assumes one-hundred equidistant values in the interval $[0, \pi/2]$, which gives a 10,000-vector weight space. Then, we choose the first weight vector in the search that maximizes the area under the receiver operating characteristic curve (AUC ROC). The AUC ROC allows measuring the classification quality in a threshold-free manner. We compute that metric by means of the scikit-learn roc_auc_score function.

Specifically for the PCDQN, we conduct a nested grid search for the best parameter combination and weight vector for each classification problem. The space of parameters is formed here by all combinations $(\tau, \delta)$, where

\[ \tau \in \left\{ \frac{1}{4}, \frac{1}{2}, 1, 2, 4 \right\} \]

and

\[ \delta \in \left\{ 0, \pi, \frac{\pi}{2}, \frac{3\pi}{4}, \pi, \frac{5\pi}{4}, \frac{3\pi}{2} \right\}. \]

Actually, the combination $(1, 0)$ is not allowed since it produces the CDQN. Consequently, there are 34 allowed combinations to search. Those combinations explore a variety of lengths and shifts along the activation function period, which is $2\pi$. For each combination of $\tau$ and $\delta$, we search that 10,000-vector weight space. Then, we choose the first values of $(\tau, \delta)$ and $(\phi_0, \phi_1)$ that maximize the AUC ROC together in the nested search.

C. Results and Discussion

Table I shows the maximum AUC ROC that each quantum neuron achieved for each classification problem. The CVQN and the CDQN perfectly solved the diagonal blobs when targeting the class Center. The CVQN also solved the NXOR-like square blobs, while the CDQN performed like a random model. On the other hand, while the CVQN did not solve the concentric circles when targeting the class Inner, the CDQN did. In summary, the CVQN and the CDQN only solved two problems each. The PCDQN produced optimal solutions for all problems, even for those problems where the other neurons performed like merely random models. Those results confirm the hypothesis that parametrization gives flexibility.

The best parameter combinations and weight vectors of each quantum neuron for each classification problem are reported in Table II. Actually, the weights have about 16 decimal places. We report here only the first two decimal places, which is enough to retrieve the original weights.

In the following, we use those best parameters and weights to study the quantum neuron solutions. We then plot the activation shape of each quantum neuron for each classification problem, which depends on the best parameters for the PCDQN. Those shapes are drawn as functions of the Euclidean distance from the input data to the corresponding best weight vector. A problem is solved if the neuron outputs for the black blobs are higher than the ones for the red blobs, regardless of the Euclidean distances to the best weight vector. In this way, there will be a threshold that separates the classes perfectly.

Fig. 8 contrasts the quantum neuron solutions for the two problems derived from the diagonal blobs. The CVQN solves the problem when targeting the class Center by positioning the weight vector closer to the center cluster, which gives values of activation for that cluster higher than the ones for the corner clusters, as shown in Fig. 8(a). That CVQN solution required the ability to also distinguish the black blobs from the red blobs at the intersection of Euclidean distances between the clusters. A similar CVQN solution correctly separated the upper left cluster from the center cluster when targeting the class Corner, although the bottom right cluster has incorrectly obtained the lowest values, as shown in Fig. 8(d). The CDQN and the PCDQN solve the problem when targeting the class Center by positioning the weight vector close to the target cluster, and then implementing a monotonic decay as the Euclidean distance increases in any direction, as shown in Fig. 8(b) and (c). A monotonic decay does not solve the problem when targeting the class Corner, as shown in Fig. 8(e) for the CDQN. Actually, a monotonic growth from a weight vector within the center cluster solves that problem, as shown in Fig. 8(f) for the PCDQN, which can be achieved by means of parametrization.

Fig. 7. Toy classification problems where the quantum neurons are applied in this demonstration. The positive class is represented by the black blobs, while the negative class is represented by the red ones. All inputs are in the interval $[0, \pi/2]$. The problems are formed by data in (a) and (d) diagonal, (b) and (e) circles, and (c) and (f) square.

TABLE I

| Dataset       | Target   | CVQN | CDQN | PCDQN |
|---------------|----------|------|------|-------|
| Diagonal blobs| Center   | 1.0  | 1.0  | 1.0   |
|               | Corner   | 0.5  | 0.5  | 1.0   |
| Concentric circles | Inner | 0.8244 | 1.0  | 1.0   |
|                | Outer   | 0.502 | 0.4088 | 0.9524 |
| Square blobs   | XOR-like| 0.5  | 0.5  | 1.0   |
|                | NXOR-like| 1.0  | 0.5  | 1.0   |
Fig. 8. Solutions obtained by the CVQN, the CDQN, and the PCDQN, respectively, for the diagonal blobs when targeting the classes (a)–(c) Center and (d)–(f) Corner.

Fig. 9. Solutions obtained by the CVQN, the CDQN, and the PCDQN, respectively, for the concentric circles when targeting the classes (a)–(c) Inner and (d)–(f) Outer.

Fig. 10. Solutions obtained by the CVQN, the CDQN, and the PCDQN, respectively, for the square blobs when targeting (a)–(c) XOR problem and (d)–(f) NXOR problem.
neuron solutions are validated by the circuit realizations in about 0.82 in the second problem. Therefore, the quantum PCDQN generated optimal results, while the CVQN was reported in Table I for those two problems. The CDQN and In fact, the quantum neurons reproduced the reference values representing the CDQN, and the green bars represent the PCDQN. Fig. 11. The blue bars represent the CVQN, the orange bars circuit-generated values of AUC ROC are contrasted in each classification problem on the quantum simulator. Those errors in the activation estimates.

twelve classification problems of 100 samples each is 12 000 000. of circuit executions for three quantum neurons applied in each classification problem to estimate the activation with a quantum neuron is executed 20 000 times for each input vector of vectors, we took the proportion of 1’s in the circuit outputs after 20 000 executions of the circuit. Thus, each quantum neuron is executed 20000 times for each input vector of each classification problem to estimate the activation with respect to the corresponding best weight vector. The amount of circuit executions for three quantum neurons applied in two classification problems of 100 samples each is 12 000 000. Nevertheless, finite numbers of execution can still cause small errors in the activation estimates.

After estimating the activation, we can compute the circuit-generated AUC ROC of each quantum neuron for each classification problem on the quantum simulator. Those circuit-generated values of AUC ROC are contrasted in Fig. 11. The blue bars represent the CVQN, the orange bars represent the CDQN, and the green bars represent the PCDQN. In fact, the quantum neurons reproduced the reference values reported in Table I for those two problems. The CDQN and the PCDQN generated optimal results, while the CVQN was optimal in the first problem but generated an AUC ROC of about 0.82 in the second problem. Therefore, the quantum neuron solutions are validated by the circuit realizations in the quantum simulator.

D. Execution on a Quantum Simulator

Finally, we demonstrate the physical feasibility of the previous numerical solutions through a proof-of-concept experiment on a quantum simulator. We address here the two classification problems used in [13]: Diagonal blobs targeting the class Center and Concentric circles targeting the class Inner. The quantum neuron solutions for those two problems are then implemented in quantum circuits that are executed on the Qiskit QASM simulator.

To estimate the quantum neuron activation for a given pair of vectors, we took the proportion of 1’s in the circuit outputs after 20 000 executions of the circuit. Thus, each quantum neuron is executed 20000 times for each input vector of each classification problem to estimate the activation with respect to the corresponding best weight vector. The amount of circuit executions for three quantum neurons applied in two classification problems of 100 samples each is 12 000 000. Nevertheless, finite numbers of execution can still cause small errors in the activation estimates.

After estimating the activation, we can compute the circuit-generated AUC ROC of each quantum neuron for each classification problem on the quantum simulator. Those circuit-generated values of AUC ROC are contrasted in Fig. 11. The blue bars represent the CVQN, the orange bars represent the CDQN, and the green bars represent the PCDQN. In fact, the quantum neurons reproduced the reference values reported in Table I for those two problems. The CDQN and the PCDQN generated optimal results, while the CVQN was optimal in the first problem but generated an AUC ROC of about 0.82 in the second problem. Therefore, the quantum neuron solutions are validated by the circuit realizations in the quantum simulator.

VI. RECOGNIZING HANDWRITTEN DIGITS

Since we validated the quantum neurons in toy datasets, the next step is to verify their performances in a real dataset in pursuit of more conclusive evidence. In this way, we addressed the handwritten digits dataset provided by the scikit-learn load_digits function. Each one of the ten digits has approximately 180 samples in the dataset. For illustrative purposes, the first occurrence of each digit is shown in Fig. 12, where Fig. 12(a) shows the digit 0, Fig. 12(b) shows the digit 1, and radial-basis activation functions, which are defined, respectively, as

$$\phi_1(i \cdot \hat{w}) = i \cdot \hat{w}, \quad \phi_2(i \cdot \hat{w}) = (1 + e^{-i \cdot \hat{w}})^{-1}, \quad \text{and} \quad \phi_3(\|i - \hat{w}\|) = e^{-i/2\|i - \hat{w}\|^2},$$

with $\|i - \hat{w}\|$ all of them with a precision of four qubits, i.e., subdividing the output interval into $2^4 = 16$ discrete parts. The implementations of these three discrete activation functions are named here as linear discrete quantum neuron (LDQN), sigmoid discrete quantum neuron (SDQN), and radial-basis discrete quantum neuron (RBDQN). Since this quantum architecture encodes information in the basis states as bit strings, we can use the data in the original integer interval $[0, 16]$ for each entry as well as the entries for the 10 000 random weight vectors sampled from NumPy with a seed of 0. However, due to exploding inner products and Euclidean distances, we scaled the input and weight entries to $[0, 0.5]$ in the activation function computations. Finally, for each classification problem, each discrete quantum neuron can search for the AUC ROC maximization by trying all weight vectors.

As a natural consequence, it is possible to see that the quantum neurons that approximate existing activation functions do not represent any advantage over their classical counterparts in terms of discriminative power. Nonetheless, those discrete quantum neurons provide a quantum baseline for the kernel-based quantum neurons with no classical analog.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Fig. 13. Best weight vector and activation function concerning such weight vector for each quantum neuron in the task of recognizing the digit 0 for (a) and (g) LDQN, (b) and (h) SDQN, (c) and (i) RBDQN, (d) and (j) CVQN, (e) and (k) CDQN, and (f) and (l) PCDQN.

Table III

| Target | LDQN | SDQN | RBDQN | CVQN | CDQN | PCDQN |
|--------|------|------|-------|------|------|-------|
| 0      | 0.7584 | 0.7444 | 0.9717 | 0.9852 | 0.9958 | 0.9977 |
| 1      | 0.6916 | 0.7139 | 0.7039 | 0.7857 | 0.8244 | 0.9444 |
| 2      | 0.7346 | 0.7488 | 0.8819 | 0.9237 | 0.9485 | 0.9585 |
| 3      | 0.724  | 0.728  | 0.8944 | 0.9352 | 0.9535 | 0.9567 |
| 4      | 0.7639 | 0.7646 | 0.8888 | 0.9572 | 0.9715 | 0.9717 |
| 5      | 0.6836 | 0.7164 | 0.8618 | 0.9499 | 0.9771 | 0.9774 |
| 6      | 0.7373 | 0.8086 | 0.9373 | 0.9518 | 0.9699 | 0.9694 |
| 7      | 0.6875 | 0.7456 | 0.9026 | 0.9527 | 0.9638 | 0.9642 |
| 8      | 0.7345 | 0.7296 | 0.8361 | 0.8161 | 0.9065 | 0.9092 |
| 9      | 0.6878 | 0.736  | 0.8708 | 0.9003 | 0.9302 | 0.9316 |

A deeper analysis of the results can be made by examining both the best weight vectors found in the random search and the activation functions with respect to such vectors, as evidenced in Fig. 13 for the digit 0. Fig. 13(a)–(f) show that the best weight vector of each quantum neuron has traces of the target digit. Such a similarity is confirmed in Fig. 13(g) for the LDQN and in Fig. 13(h) for the SDQN because the highest inner products were obtained by the instances of the digit 0. In the same way, the lowest Euclidean distances were obtained by the black blobs in Fig. 13(i)–(l) for the RBDQN, the CVQN, the CDQN, and the PCDQN, respectively. Since the LDQN and the SDQN implement monotonic increments in a discrete manner as the inner product increases, the digit-0 instances obtained the highest values of activation predominantly. However, for the task of recognizing the handwritten digit 0, as supported by Table III, the activation function shapes implemented by the RBDQN and the kernel-based quantum neurons fit even better. Those shapes are similar to exponential decrements as the Euclidean distance increases for the RBDQN, the CDQN, and the PCDQN, while the CVQN implements a linear decay. For the other digits, the neurons also searched for weight vectors similar to the target digit and implemented these neuron activation shapes. Table III already showed how well the neurons performed in fitting their activation shapes with respect to their best weight vectors and the target digits.

Actually, the PCDQN could implement different fitting strategies between target digits by changing the values of \( \tau \) and \( \delta \). However, the best value of \( \delta \) was 0 in all cases, except for the digit 1 where \( \delta \) equaled \( \pi \). That is, the PCDQN took advantage of the parameter \( \tau \) only, which did not provide significant gains in comparison to the CDQN, as shown in Table III. Changing \( \tau \) solely controls the decay smooth from linear to exponential shapes as the parameter value increases. The best values of \( \tau \) were 2, 1/4, 2, 1/4, 1/4, 1/2, 1/4, 1/4, 1/8, and 1/2, respectively. The actual potential emerges when \( \tau \) and \( \delta \) are exploited simultaneously, which occurred in the digit 1 where \( \tau \) equaled 1/4 and \( \delta \) equaled \( \pi \). The PCDQN solution for recognizing digit 1 is exhibited in Fig. 14. As can be observed in Fig. 14(a), the best weight vector does not have traces of the digit 1 at all, which is confirmed in Fig. 14(b) because those target neurons that we support here, namely the ones of constant circuit depth. Comparing against the discrete quantum neurons is an opportunity to investigate the advantages that can emerge using strictly quantum activation functions in classification tasks. Table III shows the maximum AUC ROC achieved by the quantum neurons when targeting each one of the ten digits in a one-versus-all approach. The simplest of the discrete quantum neurons, which is the LDQN, obtained the worst values of AUC ROC in all cases, except when targeting the digit 8 in which the SDQN was the worst. Targeting the digit 1 was the only case where the SDQN outperformed a neuron other than the LDQN. In that case, the RBDQN obtained a value of AUC ROC smaller. Actually, the RBDQN was the discrete neuron to beat here due to its significant improvements over the LDQN and the SDQN in all other cases. It turns out that kernel-based quantum neurons could surpass the performances of such a baseline. For example, the CVQN was better than the RBDQN in all cases, excluding the case of targeting the digit 8. Particularly, targeting the digit 8 was the case where the CDQN surpassed the CVQN the most, which also implied a reasonable improvement over the RBDQN. As the CDQN was better than the CVQN also in the other cases, the neuron that we proposed, even in its nonparametrized form, could better recognize handwritten digits than the previous proposals of quantum neurons. Except for digit 6, the PCDQN improved the results even more, especially for digit 1 where a boost in the AUC ROC really exhibited the potential of the activation function parametrization.
instances have the highest Euclidean distances. Fig. 14(b) also shows that the PCDQN implemented a shape where the activation increases as the Euclidean distance increases instead of the decay that such a neuron implemented in the other cases. In this case, the parametrization changed the activation shape in a way that substantially enhanced the value of AUC ROC. Those results in a real-life problem support the solid conclusion that neurons under the proposed framework can leverage the power of strictly quantum activation functions in classification tasks. As discussed here, we proposed a quantum neuron with better discriminative abilities than the existing ones, especially by parametrizing the activation function.

VII. CONCLUSION

In this article, we proposed a generalized framework to build quantum neurons that apply the kernel trick. Countless quantum neurons can be defined under that framework, including those for actual quantum devices as long as their feature mappings can be implemented in quantum circuits with a limited number of gates. Here, we presented a quantum neuron with a parametrized activation function and an efficient circuit implementation. The parametrization gives flexibility to the neuron since its activation function shape can be changed. That efficient circuit implementation further complies with the constraints of actual quantum devices compared to the existing kernel-based quantum neuron. The first reason is that a circuit with constant depth and a linear number of operations mitigates the errors by decoherence times and gate infidelities. Additionally, qubit encoding generates a straightforward state preparation, where connectivity is not a problem since that encoding strategy generates separable states. Finally, circuit rewriting is facilitated since the circuit uses elementary single-qubit gates.

Each kernel-based quantum neuron implements a fundamentally different activation function. Here, we showed those neuron activation shapes as functions of some relations between the input and weight vectors. While the existing kernel-based neuron is restricted, the parametrized neuron has the flexibility to fit different problem structures. As a first demonstration, the parametrized neuron produced optimal solutions for all nonlinear toy classification problems, including the ones where both the existing kernel-based neuron and the proposed neuron without parametrization performed like random models or worse. We also reproduced those numerical solutions in a simulation environment of quantum circuits as a means to finally validate the neuron implementations. As a last and conclusive test, we contrasted the kernel-based quantum neurons and the discrete quantum neurons in the real problem of recognizing handwritten digits. As a result, the proposed quantum neuron produced the best results in all cases. This repeated evidence of better neuron capabilities in real-life problems, mainly due to the parametrization potential, solidly exhibits the proposed quantum neuron and the proposed framework as effective models for machine learning and its applications.

Future works can discover other quantum neurons under the generalized framework proposed here since the ones used in this work are all special cases. Subsequent works should not limit themselves to studying, extending, and comparing the same kernel-based quantum neurons that we addressed. Using other gates, combining gates, applying heterogeneous gates, or even evolving some evolutionary approach, one can discover better feature mappings for classification problems. A theoretical study can define the bounds of that framework or even prove that it is universal. Defining the functions that the framework can approximate or the decision boundaries that can be drawn is fundamental to understanding the class of problems that one can solve as the framework is explored. That class of problems will reveal whether or not the framework can truly lead to practical quantum advantage.

Another example of future research direction is to extend the applications to more classification problems or even other learning tasks. The community can search for problems, also through experimental studies, where those quantum neurons can stand out from the classical models when the input size increases. However, executions in real quantum devices will be required to efficiently explore neuron computations, which also requires that technological advances occur. Additionally, the neuron weights and parameters can be adjusted by variational quantum algorithms [40], [41]. Grid search and random search are certainly not as effective as updating the weights and parameters based on quantum gradients. Finally, a network of quantum neurons under the proposed framework can be constructed. Since a single neuron demonstrated improved abilities here, the connection of multiple of them should be investigated to obtain further impressive results.

REFERENCES

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. New York, NY, USA: Cambridge Univ. Press, 2010.
[2] J. Preskill, “Quantum computing and the entanglement frontier,” 2012, arXiv:1203.5813.
[3] N. H. Nguyen, E. C. Behrman, M. A. Moustafa, and J. E. Steck, “Benchmarking neural networks for quantum computations,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 7, pp. 2522–2531, Jul. 2020.
[4] F. M. de Paula Neto, T. B. Ludermir, W. R. de Oliveira, and A. J. da Silva, “Implementing any nonlinear quantum neuron,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3741–3746, Sep. 2020.
[5] S. J. Devitt, W. J. Munro, and K. Nemoto, “Quantum error correction for beginners,” *Rep. Prog. Phys.*, vol. 76, no. 7, Jul. 2013, Art. no. 076001.
[6] J. Preskill, “Quantum computing in the NISQ era and beyond,” *Quantum*, vol. 2, p. 79, Aug. 2018.
[7] F. Leymann and J. Barzen, “The bitter truth about gate-based quantum algorithms in the NISQ era.” Quantum Sci. Technol., vol. 5, no. 4, Oct. 2020, Art. no. 044007.

[8] A. Zulehner, A. Paler, and R. Wille, “An efficient methodology for mapping quantum circuits to the IBM QX architectures,” IEEE Trans. Comput-Aided Design Integr. Circuits Syst., vol. 38, no. 7, pp. 1226–1236, Jul. 2019.

[9] R. LaRose, “Overview and comparison of gate level quantum software platforms.” Quantum, vol. 3, p. 130, Mar. 2019.

[10] F. Acasiete, F. P. Agostini, J. K. Moqadam, and R. Portugal, “Implementation of quantum walks on IBM quantum computers,” Quantum Inf. Process., vol. 19, no. 12, pp. 1–20, Dec. 2020.

[11] G. Acampora and A. Vitiello, “Implementing evolutionary optimisation on actual quantum processors,” NPJ Quantum Inf., vol. 5, no. 1, p. 26, Mar. 2019.

[12] S. Mangini, F. Tacchino, D. Gerace, and D. Bajoni, “Quantum computing model of an artificial neuron with continuously valued input data,” Mach. Learning, Sci. Technol., vol. 1, no. 4, Dec. 2020, Art. no. 045008.

[13] F. Tacchino, P. Barkoutsos, C. Macchiavello, I. Tavernelli, D. Gerace, and D. Bajoni, “Quantum implementation of an artificial feed-forward neural network,” Quantum Sci. Technol., vol. 5, no. 4, Oct. 2020, Art. no. 044010.

[14] E. Grant et al., “Hierarchical quantum classifiers,” NPJ Quantum Inf., vol. 4, no. 1, p. 65, Dec. 2018.

[15] I. Cong, S. Choi, and M. D. Lukin, “Quantum convolutional neural networks,” Nature Phys., vol. 15, no. 12, pp. 1273–1278, 2019.

[16] P.-L. Dallaire-Demers and N. Killoran, “Quantum generative adversarial networks,” Phys. Rev. A, Gen. Phys., vol. 98, no. 1, 2018, Art. no. 012324.

[17] C. Zoufal, A. Lucchi, and S. Woerner, “Quantum generative adversarial networks for learning and random load distributions,” NPJ Quantum Inf., vol. 5, no. 1, p. 103, Nov. 2019.

[18] D. Konar, S. Bhattacharyya, B. K. Panigrahi, and E. C. Behrman, “Qutrit-inspired fully self-supervised shallow quantum learning network for brain tumor segmentation,” IEEE Trans. Neural Netw. Learn. Syst., vol. 33, no. 11, pp. 6331–6345, Nov. 2022.

[19] H. Buhrman, R. Cleve, J. Watrous, and R. de Wolf, “Quantum fingerprinting,” Phys. Rev. Lett., vol. 87, no. 16, Sep. 2001, Art. no. 167902.

[20] V. Havlíček et al., “Supervised learning with quantum-enhanced feature spaces,” Nature, vol. 567, no. 7747, pp. 209–212, Mar. 2019.

[21] M. Schuld and N. Killoran, “Quantum machine learning in feature Hilbert spaces,” Phys. Rev. Lett., vol. 122, no. 4, Feb. 2019, Art. no. 040504.

[22] C. Ding, T. Bao, and H. Huang, “Quantum-inspired support vector machine,” IEEE Trans. Neural Netw. Learn. Syst., vol. 33, no. 12, pp. 7210–7222, Dec. 2022.

[23] Y. Liu, S. Arunanachalam, and K. Temme, “A rigorous and robust quantum speed-up in supervised machine learning,” Nature Phys., vol. 17, no. 9, pp. 1013–1017, Sep. 2021.

[24] S. Bravyi, D. Gosset, and R. König, “Quantum advantage with shallow circuits,” Science, vol. 362, no. 6412, pp. 308–311, Oct. 2018.

[25] S. Bravyi, D. Gosset, R. König, and M. Tomamichel, “Quantum advantage with noisy shallow circuits,” Nature Phys., vol. 16, no. 10, pp. 1040–1045, Oct. 2020.

[26] Y. Li and S. C. Benjamin, “Efficient variational quantum simulator incorporating active error minimization,” Phys. Rev. X, vol. 7, no. 2, Jun. 2017, Art. no. 021050.

[27] K. Temme, S. Bravyi, and J. M. Gambetta, “Error mitigation for short-depth quantum circuits,” Phys. Rev. Lett., vol. 119, no. 18, Nov. 2017, Art. no. 180509.

[28] A. Kandala, K. Temme, A. D. Córcoles, A. Mezzacapo, J. M. Chow, and J. M. Gambetta, “Error mitigation extends the computational reach of a noisy quantum processor,” Nature, vol. 567, no. 7749, pp. 491–495, Mar. 2019.

[29] E. M. Stoudenmire and D. J. Schwab, “Supervised learning with tensor networks,” in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29, Red Hook, NY, USA: Curran Associates, 2016, pp. 4806–4814.

[30] T. M. Cover, “Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition,” IEEE Trans. Electron. Comput., vol. EC-14, no. 3, pp. 326–334, Jun. 1965.

[31] N. S. Yanofsky and M. A. Mannucci, Quantum Computing for Computer Scientists. New York, NY, USA: Cambridge Univ. Press, 2008.

[32] S. Haykin, Neural Networks and Learning Machines. London, U.K.: Pearson, 2009.

[33] C. M. Bishop, Pattern Recognition and Machine Learning. New York, NY, USA: Springer, 2006.

[34] M. Rossi, M. Huber, D. Bruiß, and C. Macchiavello, “Quantum hypergraph states,” New J. Phys., vol. 15, no. 11, Nov. 2013, Art. no. 113022.

[35] Qiskit Contributors, “Qiskit: An open-source framework for quantum computing,” 2023, doi: 10.5281/zenodo.2573505.

[36] F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” J. Mach. Learn. Res., vol. 12, pp. 2825–2830, Oct. 2012.

[37] C. R. Harris et al., “Array programming with NumPy,” Nature, vol. 585, no. 7825, pp. 357–362, 2020.

[38] S. Yan, H. Qi, and W. Cui, “Nonlinear quantum neuron: A fundamental building block for quantum neural networks,” Phys. Rev. A, Gen. Phys., vol. 102, no. 5, Nov. 2020, Art. no. 052421.

[39] A. K. Kerenzi et al., “Variational quantum algorithms,” Nature Rev. Phys., vol. 3, no. 9, pp. 625–644, 2021.

[40] R. Huang, X. Tan, and Q. Xu, “Learning to learn variational quantum algorithm,” IEEE Trans. Neural Netw. Learn. Syst., early access, Feb. 28, 2022, doi: 10.1109/TNNLS.2022.3151127.

Jonathan H. A. de Carvalho received the B.Sc. degree (cum laude) in information systems from the Universidade Federal Rural de Pernambuco, Recife, Brazil, in 2019, and the M.Sc. degree in computer science from the Universidade Federal de Pernambuco, Recife, in 2022, where he is currently pursuing the Ph.D. degree in computer science.

His research interests include quantum computing, artificial neural networks, and time-series forecasting.

Fernando M. de Paula Neto received the bachelor’s degree (cum laude) in computer engineering, the master’s and Ph.D. degrees in computer science from the Universidade Federal de Pernambuco (UFPE), Recife, Brazil, in 2014, 2016, and 2018, respectively.

He joined the Center of Informatics, Department of Information and Systems, UFPE, in 2019, where he is currently an Adjunct Professor. His current research interests include quantum computing, quantum machine learning, dynamical systems and chaos, neural networks, and machine learning.