Implementing any nonlinear quantum neuron

Fernando M de Paula Neto, *Member, IEEE,* Teresa B Ludermir, *Senior Member, IEEE,* Wilson R de Oliveira, and Adenilton J da Silva

***Abstract*—The ability of artificial neural networks (ANNs) to adapt to input data and perform generalisations are intimately connected to the use of nonlinear *activation and propagation functions*. Quantum versions of ANN have been proposed to take advantage of the possible supremacy of quantum over classical computing. To date, all proposals faced the difficulty of implementing nonlinear activation functions since quantum operators are linear. This paper presents an architecture to simulate the computation of an arbitrary nonlinear function as a quantum circuit. This computation is performed on the phase of an adequately designed quantum state, and quantum phase estimation recovers the result, given a fixed precision, in a circuit with linear complexity in function of ANN input size.**

***Index Terms*—Quantum neuron, Quantum neural networks, Quantum Fourier transform, Quantum computing**

I. Introduction

THE artificial neuron operates in two stages. The first one is the interaction between inputs and internal weights of the neuron. The function which produces this interaction is called *propagation function*. In the second stage, the *activation function* acts on the result of the propagation function and produces the output of the neuron [1]. Equation 1 shows f and g as propagation and activation functions, respectively, of the classical neuron N applied to the k inputs $x_1, ..., x_k$, and and weights $w_1, ..., w_k$.

$$N(x_1, ..., x_k, w_1, ...w_k) = g(f(x_1, ..., x_k, w_1, ..., w_k)) \quad (1)$$

Quantum computing is providing new tools for processing information [2] which seems to supersede their classical counterparts in efficiency, at least for a certain class of problems. In particular, there exists a growing effort to design quantum machine learning algorithms [3]. One challenge in quantum neural networks is that the quantum operations in the circuit model are unitary, but the artificial neuron uses nonlinear functions [4].

There are many proposals of quantum neural networks implementing the artificial neuron in quantum circuit model [4], [5], [6], [7], [8], [9]. For an updated review of the many quantum neural networks models see [6].

Fernando M de Paula Neto, Teresa B Ludermir and Adenilton J da Silva are with Centro de Informática, CIn, Universidade Federal de Pernambuco, Brazil, e-mail: {fernando, tbl, ajsilva}@cin.ufpe.br.

Wilson R de Oliveira is with Departamento de Estatística e Informática, DEINFO, Universidade Federal Rural de Pernambuco, Brazil, e-mail: wilson.rosa@gmail.com

Manuscript received X, X; revised X X, X.

Existing contributions are related to some internal function of quantum neuron models. In this paper, our main contribution is to discuss a general model of a quantum neuron with any activation and propagation functions. The computation of the neuron information propagation is performed on the phase of a quantum state processed by quantum inverse Fourier transform. The inputs and weights of the neuron are quantum registers of a quantum circuit, and the neuron is a quantum operator parameterised by its precision and internal functions. The proposed neuron has the properties of classical neurons, executes with linear cost depending on the size of the input and uses quantum features such as superposition and entanglement. This makes it possible to explore all parameters space, with a given fixed precision, which in a classic computer would be unfeasible.

II. Quantum computation

A. Quantum bits

A *quantum bit* (qubit) is a unit of information in quantum computation - the quantum analogue of a binary bit. A qubit is a two-state quantum mechanical system. A qubit is represented as a two-dimensional vector in the complex vector space $\mathbb{C}^2$. The two independent (physically distinguishable) quantum states of the system are represented as basis states (or basis vectors). The canonical (or computational) basis is composed of the vectors $|0\rangle = [1,0]^T$ and $|1\rangle = [0,1]^T$, where $|\cdot\rangle$ is a notation introduced by Dirac to represent quantum states [2]. The $\langle\cdot|$ notation represents the complex conjugate of the vector $|\cdot\rangle$. As such any qubit can be seen as the linear combination (usually called *superposition*) of the basis vectors, $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers. Qubits are normalised, which requires that $|\alpha|^2 + |\beta|^2 = 1$. This notation also means that the qubit has probability $|\alpha|^2$ to be measured as 0 and $|\beta|^2$ to be measured as 1 and α and β are then called *probability amplitudes* or simply *amplitudes*.

The tensor product $\otimes$ is used to represent quantum systems composed of two or more qubits $|\mathbf{g}\rangle = |ab\rangle = |a\rangle \otimes |b\rangle$. Here we will use the bold font for the representation of quantum states with more than one qubit. Quantum states with more than one qubit are also called quantum registers.

B. Quantum operators

Quantum states are modified by quantum operators which change the amplitude values of the qubits. A

quantum operator $\mathbf{U}$ acting over a n qubits system can be represented as a unitary complex matrix of order $2^n \times 2^n$. A matrix $\mathbf{U}$ is unitary if $\mathbf{U} \circ \mathbf{U}^\dagger = \mathbf{U}^\dagger \circ \mathbf{U} = \mathbf{I}$, where $\mathbf{U}^\dagger$ is the adjoint (conjugate transpose) of $\mathbf{U}$, $\mathbf{I}$ is the identity matrix and $\circ$, usually omitted, is the usual matrix product.

In the same way we can combine quantum states, quantum operators can also be combined using tensor product. For two (n_0, m_0)-dimensional matrix $\mathbf{U}$ and (n_1, m_1)-dimensional matrix $\mathbf{V}$, their tensor product, $\mathbf{U} \otimes \mathbf{V}$, is a $(n_0 n_1, m_0 m_1)$-dimensional matrix. We denote as $\mathbf{A}^{\otimes s}$ the s-fold tensor product of $\mathbf{A}$.

Operators are defined by saying how they behave on the basis states. The quantum NOT operator $\mathbf{X}$) is defined as $\mathbf{X}|0\rangle = |1\rangle$ and $\mathbf{X}|1\rangle = |0\rangle$. The Hadamard operator $\mathbf{H}$ is defined as $\mathbf{H}|0\rangle = 1/\sqrt{2}(|0\rangle + |1\rangle)$ and $\mathbf{H}|1\rangle = 1/\sqrt{2}(|0\rangle - |1\rangle)$. The **CNOT** is a two qubits operator. It acts on a control qubit and a target qubit. If the control qubit is $|0\rangle$, the target qubit is left unchanged. Otherwise, if the control qubit is $|1\rangle$ the $\mathbf{X}$ operator is applied to the target qubit.

We can generalise and define an $(n+1)-$ary **CNOT** having n control qubits and requiring all the control qubits to be $|1\rangle$ for applying $\mathbf{X}$ on the target qubit. Any quantum operator can be constructed using a sequence of $1-$qubit quantum operators and **CNOT** operators. We say that this set of gates is *universal* for quantum computation, *i.e* any quantum operators can be realised by this set of gates, in the same vain that the NAND gate is universal for classical Boolean computation.

The idea of the **CNOT** operator can be generalised using an arbitrary $1-$qubit operator $\mathbf{U}$ in place of $\mathbf{X}$. A further useful generalisation has m control qubits with n targets qubits and an $n-$qubits operator $\mathbf{U}$.

C. Quantum circuit

We can represent quantum operations by quantum circuits. This graphical representation considers the qubits as wires and quantum operators as boxes. The flow of the execution, as in the classical case, is from left to right.

Figure 1 has a quantum circuit composed of a **CNOT**, where a filled circle depicts the control qubit and a circle with a plus inside depicts the target qubit, one $\mathbf{X}$ operator and one controlled-$\mathbf{H}^{\otimes 2}$ operator.

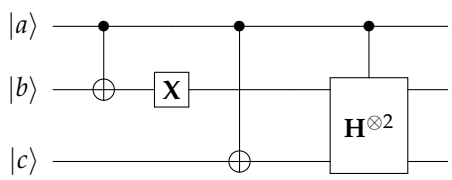


Fig. 1: An example of quantum circuit with two **CNOT** operators, one $\mathbf{X}$ operator and one controlled-$\mathbf{U}$ operator where $\mathbf{U}$ here is $\mathbf{H}^{\otimes 2} = \mathbf{H} \otimes \mathbf{H}$.

D. Quantum Fourier Transform

The quantum Fourier transform (QFT) is a tool widely used in quantum computing and an essential step in the Shor algorithm, a well-known quantum algorithm that can efficiently find prime factors of a given integer [10]. In this paper, the QFT is an essential tool used in the recovery procedure of our proposed memory.

The QFT, inspired by the discrete Fourier transform, is the linear operator defined over an orthonormal basis $|0\rangle, \cdots, |N-1\rangle$ of a N-dimensional complex vector space, as:

$$|a\rangle \xrightarrow{QFT} |a'\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i}{N} ak} |k\rangle \tag{2}$$

By means of the QFT we can represent a given quantum state $|a\rangle$ in a new quantum state $|a'\rangle$ where its phase, $e^{\frac{2\pi i}{N} ak}$, stores the information of the original quantum state $|a\rangle$. As the QFT is a unitary operator, its inverse operator can recover the information in the phase of a quantum state, transforming it into a respective quantum state.

III. Quantum neuron model

The proposed quantum neuron circuit simulates any classical activation and propagation functions within a given precision. This circuit is shown in Figure 2 and is inspired in the one proposed in [8]. The quantum neuron is composed of two main components: a quantum operator $\mathbf{N}_\tau$ and the phase estimation by the $\mathbf{QFT}^{-1}$ operator. The main idea is to control the output of the neuron in the phase of quantum state depending on the input and weights. The $\mathbf{N}_\tau$ operator performs the quantum neuron functions and is defined as

$$\mathbf{N}_\tau(p,q,g,f) = \sum_{x_1=0}^{2^p-1} \cdots \sum_{x_n=0}^{2^p-1} \sum_{w_1=0}^{2^q-1} \cdots \sum_{w_n=0}^{2^q-1} Ph(\tau,g,f)|\chi,\omega\rangle\langle\chi,\omega| \tag{3}$$

where $Ph(\tau,g,f) = e^{\frac{2\pi i}{2^\tau} g(f(\chi,\omega))}$ is the phase shift element according the input and weights values and $\chi = x_1, ..., x_n$ and $\omega = w_1, ..., w_n$ are the possible values of input and weights, respectively. Each input x_i and each weight w_i has a string of qubits with a given precision.

The execution of the neuron requires three quantum registers. The first quantum register will store the output of the neuron, the second register stores the input values to the neuron, and the third register stores the neuron weights. Let us consider a neuron with n inputs $x_1, x_2, ..., x_n$ and weights $w_1, w_2, ..., w_n$, with an output precision of τ qubits. Each input x_i has a correspondent weight w_i. Each input and weight has p and q qubits, respectively. In the circuit shown in Figure 2. The activation and propagation functions are denoted as $f(x_1, x_2, ..., x_n, w_1, w_2, ...w_n)$ and $g(f(x_1, x_2, ..., x_n, w_1, w_2, ...w_n))$, respectively. It is possible to see these three registers in the circuit shown in Figure 2 with this configuration. The output signal provided by the neuron is placed on quantum phase of the first

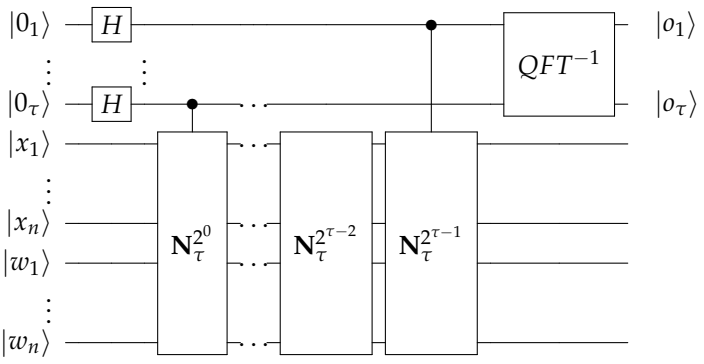


Fig. 2: Quantum neuron architecture (qN). The operator $\mathbf{N}_\tau$ executes the nonlinear function of the neuron which is created as a function of the inputs $|x_1,..x_n\rangle$ and weights $|w_1,...w_n\rangle$ and puts the result on the first quantum register phase through successive τ Controlled-$\mathbf{N}_\tau$ operations. The QFT^{-1} operator recovers this phase information and transforms it in the quantum state $|o_1,...,o_\tau\rangle$.

register and it is recovered by the quantum inverse Fourier transform operator.

The proposed quantum neuron operator $\mathbf{N}_\tau$ is parameterised by the τ value because it is necessary to know in how many qubits the output of the neuron will be placed. In other words, the τ value indicates the precision of the computation of the activation and propagation function. It must fit in a quantum state with a predefined number of qubits τ. For the simulation of this quantum neuron, as it happens in a common digital computer, it would require a precision of 64 qubits for example. The quantum neuron has also f and g functions as parameters, but we removed it of the subscription to not overload the notation.

After the application of the operators $\mathbf{N}_\tau^{2^0}$, $\mathbf{N}_\tau^{2^1}$,...,$\mathbf{N}_\tau^{2^\tau-1}$ on the last two registers $|x_1,...,x_n\rangle$ and $|w_1,...,w_n\rangle$, the first register can be processed by the quantum inverse Fourier transform to capture the result of the quantum neuron output. This transformation is shown in Equation 4.

$$\begin{aligned}&\sum_{s=0}^{2^\tau-1} e^{\frac{2\pi i}{2^\tau}.s.g(f(x_1,...,x_n,w_1,...,w_n))}|s\rangle|x_1,...,x_n\rangle|w_1,...,w_n\rangle\\ &\xrightarrow{QFT^{-1}} |g(f(x_1,...,x_n,w_1,...,w_n))\rangle|x_1,...,x_n,w_1,...,w_n\rangle\end{aligned} \tag{4}$$

To deal with negative numbers, we can consider that the first qubit in a quantum state represents the sign of the number. In this case, when a quantum state has m qubits, this means that we can represent the numbers from $-2^{m-1}-1$ to $2^{m-1}-1$. We use a superscription D in the number inside the *ket* and *bra* notation to indicate that we are considering the decimal representation with a sign. For example, we have the states $|011\rangle = |\mathbf{3}^D\rangle$ and $|111\rangle = |-\mathbf{3}^D\rangle$ with that representation. The decimal representation can also be used considering the representation $|j_1,j_2,...,j_p\rangle = \sum_p j_n 2^{-p}$[2]. We present here five examples of activated and propagation functions to be implemented in the quantum neuron using this architecture. These examples can be extended to any function considering the structure of the domain and codomain of the neuron.

A. Inner product propagation function and linear activation function

We show here an example for a quantum neuron that has a linear activation and inner product between input and weight as propagation function. For this case, the domain of f is in $[-2^{p-1}-1,2^{p-1}-1]^n \times [-2^{q-1}-1,2^{q-1}-1]^n$ and its codomain is in $[-n(2^{p-1}-1)(2^{q-1}-1), n(2^{p-1}-1)(2^{q-1}-1)]$. The domain of g is in $[n(2^{p-1}-1)(2^{q-1}-1), n(2^{p-1}-1)(2^{q-1}-1)]$ and its codomain is in $[0,2^\tau)$. Let us consider $n=3$ inputs/weights each one having $p=q=m=4$ qubits. We have a function f that calculates the inner product between input and weights $\sum_n x_i w_i$ that generates the maximum value $n(2^{m-1}-1)(2^{m-1}-1) = 3(7)(7) = 147$ and minimum value -147. Considering that g is a linear function $g(x)=x$, then, the outputs to be processed by the neuron is 147 in maximum or -147 in minimum. This information is loaded in a state with at least 9 qubits (1 for sign and 8 to the number). Then, τ value chosen is $\tau = 9$. The recovery procedure using quantum Fourier transform will recover the state between the state $|010010011\rangle = |\mathbf{147}^D\rangle$ to the state $|110010011\rangle = |-\mathbf{147}^D\rangle$.

All the usual activation functions used in the classical ANN literature [1] such as the Step activation function, the sigmoid activation function, the radial basis activation and propagation function, the ReLU (rectified linear) activation function when implemented in a digital computer can all be considered as a Boolean function (from bits to bits) and as such have a *quantum oracle* version. Given a Boolean function $f:\{0,1\}^m \to \{0,1\}^n$, define the unitary operator $U_f : \mathbb{C}^{\otimes 2(m+n)} \to \mathbb{C}^{\otimes 2(m+n)}$ which takes

$$|a_1a_2\cdots a_m\rangle|b_1b_2\cdots b_n\rangle$$

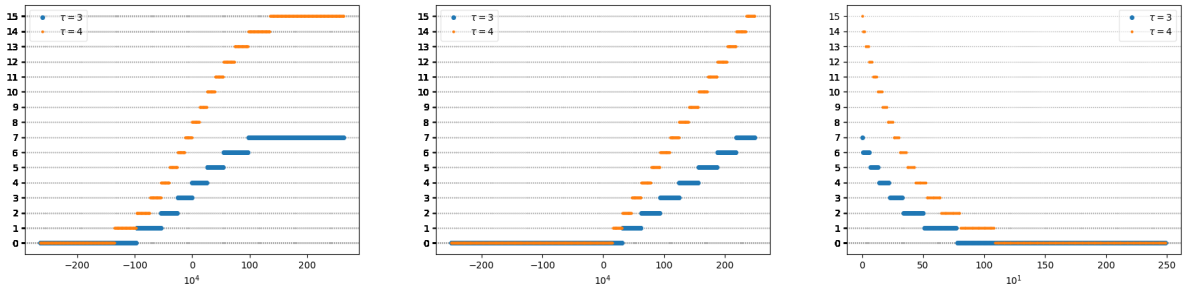
to

$$|a_1a_2\cdots a_m\rangle|(b_1b_2\cdots b_n)\oplus f(a_1a_2\cdots a_m)\rangle,$$

where $\oplus$ is the bitwise XOR. U_f is the quantum oracle version of f. The value $f(a_1a_2\cdots a_m)$ is recovered when $b_1b_2\cdots b_n$ is equal to the $n-$bits string of zeros.

B. Examples

The well known XOR problem [1] - to set an ANN to compute the exclusive OR boolean function - is hard for ANNs. Networks with one layer of perceptrons cannot compute the XOR function or any other function which is not linearly separable [11]. However, a general single artificial neuron can compute XOR depending on the activation function.



(a) (b) (c)

Fig. 3: Graphs of (a) sigmoid, (b) ReLu and (c) exponential propagation functions for $\tau = 3$ and $\tau = 4$.

Let us consider the input register $|\mathbf{x}\rangle$ and weights register $|\mathbf{w}\rangle$ with 2 qubits. As the possible values for the output is either 0 or 1, it is enough τ be set to 1, since there are $2^\tau = 2$ different expected states. We configure the weights to be $|11\rangle$ and then the neuron architecture is defined with the linear propagation function, and the activation function as $g_{ex1}(x) = 1$ when $x \in \{3, 6\}$ and 0 otherwise. The output of the f_{ex1} function when the input is 00 and 11 is 0 and 2 , respectively, and the output of the neuron is 0 in both case. When the input is either 01 or 10 the f_{ex1} output is 1 in both case and the neuron output is then 1. The neuron operator is shown in Equation 5 represents this operation.

$$\mathbf{N}_1 = \sum_{\mathbf{x}=01,10} e^{\frac{2\pi i}{2}} |\mathbf{x}, 11\rangle\langle\mathbf{x}, 11| + \sum_{\mathbf{x}\neq 01,10} |\mathbf{x}, 11\rangle\langle\mathbf{x}, 11| + \sum_{\mathbf{x}} \sum_{\mathbf{w}\neq 11} |\mathbf{x}, \mathbf{w}\rangle\langle\mathbf{x}, \mathbf{w}| \tag{5}$$

Consider now a neuron with ten inputs and weights, each one having ten qubits where the first qubit represents the negative sign. For the inner product propagation function, the possible minimum and maximum values are $-10 \cdot 2^9 = -2621440$ and $10 \cdot 2^9 = 2621440$, respectively. Figures 3a and 3b show the sigmoid and ReLu activation functions plots, respectively. The neuron operator is $\mathbf{N}_\tau(p = 10, q = 10, g_{ex1}, f_{ex1})$.

Another example is the quantum neuron with RBF as propagation function. It represents in its weights the values of a centroid. We can see the radial basis functions as a propagation function f_{ex2} and the exponential with variance equal to one to be the activation function g_{ex2}. The centroid components w_1, w_2, w_3, w_4, w_5 are represented in the quantum states $|w_1\rangle, |w_2\rangle, |w_3\rangle, |w_4\rangle, |w_5\rangle$. Assume that each input and weight has ten qubits. The neuron circuit for this example is $\mathbf{N}_\tau(p = 10, q = 10, g_{ex2}, f_{ex2})$. The maximum value of the distance calculated by the neuron is $\sqrt{5 \cdot (1023^2)} \approx 2287.50$ and the minimum distance is 0 for a given input set. The graphics of the function g is shown in Figure 3c for $\tau = 3$ and 4.

C. Computational complexity

The computational complexity of the quantum generalised neuron is linear, in function of the input size. The quantum inverse Fourier transform requires $\frac{\tau(\tau+1)}{2} + 3\frac{\tau}{2}$ gates [2] and the τ operations are done before the calculus of the phase estimation. Since τ is a parameter that can be considered fixed in the circuit [8], the final complexity becomes linear.

D. Training and feasibility

Any function implemented in a digital computer can be implemented in a quantum computer. However, the quantum computer has distinct features, such as entanglement and parallelism. This allows to perform information processing with features that classical computing apparently does not have[1].

1) Parallelism: By representing classical data as an orthonormal basis of a vector (Hilbert) space, a general linear combination of the basis represents all classical data in just one quantum state. An operator which acts upon that quantum state is actually acting in parallel, by linearity, to all classical data. This phenomenon is called *quantum parallelism*. Thus, we can use all possible sets of weights and evaluate the neural network for all possible parameter values. The challenge in this action is to retrieve the pertinent information from the resulting superposition without destroying it - a phenomenon called the *wave function collapse* and is associated to the *measurement problem* [2]. By measurements or observations, only one of the superposed classical data can be obtained and with a certain probability[2]. Clever ways have been discovered to increase the probability of successfully recover data but at a computational cost such e.g. Grover's algorithm [18].

A quantum neural network with several layers of neurons can be created using the proposed neuron (see Subsection III-D5). Thus, the weights of this network need to be adjusted to model, based on some performance criteria, a given training set. This problem for example can be solved with an exhaustive quantum search whose complexity is sublinear as a function of

[1]The backpropagation algorithm used to train classical neural networks can be implemented as a quantum circuit, since this algorithm is actually implemented as a Boolean function on a digital computer. [2].

[2]Although there are some studies on the use of non-unitary operators that makes access to the amplitudes of the base states in a nonlinear form [12], [13], [14], [15], [16], [17].

the size of the search space. In [19], a quantum algorithm has been proposed with complexity $O(\sqrt{N/t})$, where N is the number of possible elements and t is the number of possible solutions. This allows exploring the entire search space, without getting stuck in local minima, with a quadratic gain over the existing classical algorithms. This is possible because the neural network will be executed with all possible weights in parallel. The quest is guided to amplify the quantum states of the base that have the desired performance. Other algorithms exploit quantum parallelism through the use of quantum search algorithms [20], [21]. Recent work has shown that it is possible to select neural network architecture using probabilistic quantum memories [22]. Other quantum algorithms that use parallelism, as the core of its operation, can also be used in this network.

2) Entanglement: A neural network that has in its features the entanglement is able to store information of weight that are interconnected with each other. In [23], it is commented that entanglement can exponentially speed up the calculation of classical functions. Entanglement can be applied in the ANN, for example, if a given weight of a connection has a value v_1, forces another connection have value v_2, whereas if the first connection has value v_3 the second connection is v_4. This means that the weights can be represented by the entangled quantum state $|w_1, w_2\rangle = \frac{1}{\sqrt{2}}(|v_1, v_2\rangle + |v_3, v_4\rangle)$. This situation can be useful for example to model several ANNs (such as an ensemble) with the same operational cost as one ANN.

3) Other quantum neurons: Other neurons models present limitations of representation of other activation functions other than those for which they were constructed. Table I has the existing quantum neuron models and the activation functions they perform. They perform some particular activation function. It is known that activation functions alter the computability of the neural network [24] and modify the speed of convergence and performance during its training [25], [26], [27].

Quantum neuron	Activation functions
Proposed neuron	Any non-linear function
qP [28]	Step function
qGFNN [29]	Step function
qPNP [30]	Non-periodic arcotangent function
qPN [31]	Arcotangent function
qAN [32]	Reversible unitary functions and non-unitary
qMPN [33]	Reversible unitary functions and non-unitary
qRAM [34]	Non-linear with one output function

TABLE I: Different activation functions for different models of quantum neurons.

4) Realisation: The realisation of the proposed model depends on the existence of a quantum computer with a reasonable number of qubits available for use and the availability of a reasonable number of operators. The quantum computational models available for simulation have only few qubits and quantum operators available, although small ANNs can be executed in available quantum simulators [35], [36] or with few qubits without losing the precision [37].

5) Quantum neural network example: An example of the quantum neural network using the proposed neuron model works is described. In terms of the quantum circuit, a layer of a neural network with the proposed neuron model is shown in Figure 4. For a layer that has p entries and q neurons, pq weights are required. A quantum neural network can then be formed by a sequence of those layers where the output of a previous layer serves as input to the next layer.

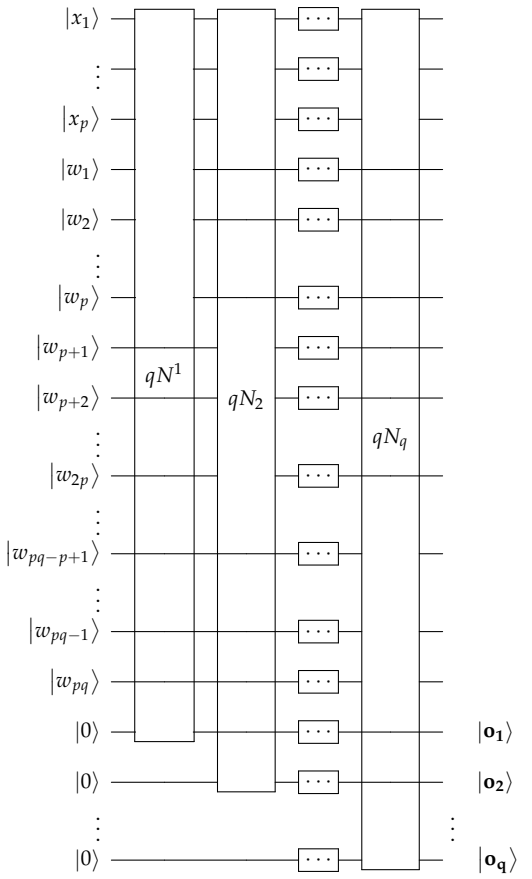


Fig. 4: One layer of quantum neurons with p inputs and q neurons, $qN_1, qN_2, ..., qN_q$. Each quantum register has n bits, which defines the numerical accuracy of the stored content. Considering each of the p entries stored in the registers, $|x_1\rangle, ..., |x_p\rangle$, and q neurons, there are in total pq registers to store the weights. When the wires are below a given operator, it means that they are used by that operator. Otherwise, they are drawn from above.

Consider that a neural network must model the following function $f(x) = ax^2 + bx$. In the network shown in Figure 5, we can see an ANN configuration that has 4 weights and is able to model this problem. The network output function is $f(x, w_1, w_2, w_3, w_4) = y = w_1^2 w_3 x^2 + w_2 w_4 x$, if the activation functions are set to $g_1(x) = x^2$ and $g_2(x) = g_3(x) = x$ and their activation functions

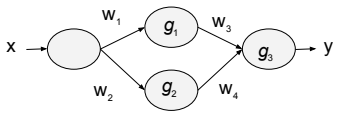


Fig. 5: Representation of a neural network with two neurons in the hidden layer.

are calculated by internal product between weight and input. The values of the weights of this network define the coefficients of the function, $a = w_1^2 w_3$ and $b = w_2 w_4$. An ANN training needs to find such weights for a given problem.

Considering that the input and weights are binary (may only have values 0 or 1) and the output has 2 qubits of precision, we see in Equation 6 the possible calculation of all superposed weights. The three neurons in this network are represented by $\mathbf{N}_{\tau=1}(p = 1, q = 1, f_1(x, w_1) = x \cdot w_1, g_1(x) = x^2)$, $\mathbf{N}_{\tau=1}(p = 1, q = 1, f_2(x, w_2) = x \cdot w_2, g_2(x) = x)$ and $\mathbf{N}_{\tau=2}(p = 1, q = 1, f_3(x_1, x_2, w_3, w_4) = x_1 \cdot \ w_3 + x_2 \cdot \ w_4, g_3(x) = x)$. The normalisation operator N is introduced to omit the amplitudes of the states of the base to simplify the notation of the equations. The values of the α_1, α_2 and α_3 amplitudes will depend on the input x.

$$
\begin{aligned}
&|y\rangle = |f_3(g_1(f_1(x,w_1), f_2(x,w_2), w_3, w_4)\rangle \\
&\xrightarrow{w_1=w_2=w_3=w_4=H|0\rangle} N(|f_3(g_1(f_1(x,0)), f_2(x,0), 0, 0)\rangle + \\
&|f_3(g_1(f_1(x,0)), f_2(x,0), 0, 1)\rangle + |f_3(g_1(f_1(x,0)), f_2(x,0), 1, 0)\rangle + \\
&|f_3(g_1(f_1(x,0)), f_2(x,0), 1, 1)\rangle + |f_3(g_1(f_1(x,0)), f_2(x,1), 0, 0)\rangle + \\
&|f_3(g_1(f_1(x,0)), f_2(x,1), 0, 1)\rangle + |f_3(g_1(f_1(x,0)), f_2(x,1), 1, 0)\rangle + \\
&|f_3(g_1(f_1(x,0)), f_2(x,1), 1, 1)\rangle + |f_3(g_1(f_1(x,1)), f_2(x,0), 0, 0)\rangle + \\
&|f_3(g_1(f_1(x,1)), f_2(x,0), 0, 1)\rangle + |f_3(g_1(f_1(x,1)), f_2(x,0), 1, 0)\rangle + \\
&|f_3(g_1(f_1(x,1)), f_2(x,0), 1, 1)\rangle + |f_3(g_1(f_1(x,1)), f_2(x,1), 0, 0)\rangle + \\
&|f_3(g_1(f_1(x,1)), f_2(x,1), 0, 1)\rangle + |f_3(g_1(f_1(x,1)), f_2(x,1), 1, 0)\rangle + \\
&|f_3(g_1(f_1(x,1)), f_2(x,1), 1, 1)\rangle) = \alpha_1|00\rangle + \alpha_2|01\rangle + \alpha_3|10\rangle.
\end{aligned}
\tag{6}
$$

IV. Conclusion

In this paper, we presented a novel generalised architecture for a weighted quantum neuron parametrised by the propagation and the (nonlinear) activation functions. The input, weights and output precision are also parameters of the neuron. The proposed neuron, as a quantum computational model, can process parallel information and calculate in a single step the output of the neuron for superposed inputs and weights. The quantum inverse Fourier transform operator is used to recover the result encoded in the output of the neuron. The new quantum neuron is a model which can be used as a quantum machine learning unit to load and recover information, using the quantum properties of superposition and entanglement. This means that it is possible to use quantum search algorithms that exhaustively explore the proposed neuron parameters at an sublinear execution cost in relation to the combination of possible inputs, finding optimal architecture configurations and weights.

Future works are focused on the study of algorithms for efficient quantum training that maximise the probability of finding optimal solutions, as well as in the study of entanglement as a learning enhancer of quantum neural networks.

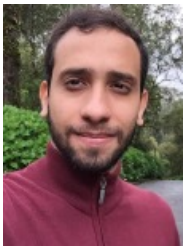
Acknowledgment

This work was supported by the Serrapilheira Institute (grant number Serra-1709-22626), CNPq, (Edital Universal, grant number 421849/2016-9 and 409415/2018-9) and CAPES (Finance Code 001).

References

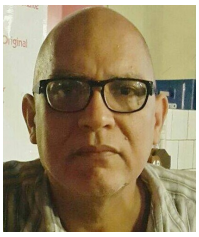
[1] S. Haykin, *Neural Networks*. Prentice Hall, 1999.
[2] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
[3] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, pp. 195 EP –, Sep 2017.
[4] M. Schuld, I. Sinayskiy, and F. Petruccione, "The quest for a quantum neural network," *Quantum Information Processing*, vol. 13, no. 11, pp. 2567–2586, Nov. 2014.
[5] M. Panella and G. Martinelli, "Neural networks with quantum architecture and quantum learning," *International Journal of Circuit Theory and Applications*, vol. 39, no. 1, pp. 61–77, 2011.
[6] S. Jeswal and S. Chakraverty, "Recent developments and applications in quantum neural network: A review," *Archives of Computational Methods in Engineering*, pp. 1–15, 2018.
[7] E. Behrman, L. Nash, J. Steck, V. Chandrashekar, and S. Skinner, "Simulations of quantum neural networks," *Information Sciences*, vol. 128, no. 3–4, pp. 257 – 269, 2000.
[8] M. Schuld, I. Sinayskiy, and F. Petruccione, "Simulating a perceptron on a quantum computer," *Physics Letters A*, vol. 379, no. 7, pp. 660 – 663, 2015.
[9] W. Hu, "Towards a real quantum neuron," *Natural Science*, vol. 10, no. 03, p. 99, 2018.
[10] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997.
[11] M. Minsky and S. A. Papert, *Perceptrons: an introduction to computational geometry*. MIT press, 2017.
[12] M. Czachor, "Remarks on search algorithms and nonlinearity," *acta physica slovaca*, vol. 48, pp. 157–162, 1998.
[13] D. S. Abrams and S. Lloyd, "Nonlinear Quantum Mechanics Implies Polynomial-Time Solution for NP-Complete and P Problems," *Phys. Rev. Lett.*, vol. 81, no. 18, pp. 3992–3995, 1998.
[14] M. Ohya and I. V. Volovich, "Quantum computing, np-complete problems and chaotic dynamics," 1999.
[15] F. M. de Paula Neto, T. B. Ludermir, W. R. De Oliveira, and A. J. da Silva, "Solving np-complete problems using quantum weightless neuron nodes," in *Intelligent Systems, 2015 Brazilian Conference on*, 2015, pp. 258–263.
[16] A. J. da Silva, W. R. de Oliveira, and T. B. Ludermir, "Weightless neural network parameters and architecture selection in a quantum computer," *Neurocomputing*, vol. 183, pp. 13–22, 2016.
[17] M. Panella and G. Martinelli, "Neural networks with quantum architecture and quantum learning," *International Journal of Circuit Theory and Applications*, vol. 39, no. 1, pp. 61–77, 2011.
[18] L. K. Grover, "Quantum Mechanics Helps in Searching for a Needle in a Haystack," *Phys. Rev. Lett.*, vol. 79, no. 2, pp. 325–328, 1997.
[19] M. Boyer, G. Brassard, P. Høyer, and A. Tapp, "Tight bounds on quantum searching," *Fortschritte der Physik: Progress of Physics*, vol. 46, no. 4-5, pp. 493–505, 1998.
[20] A. J. da Silva, W. R. de Oliveira, and T. B. Ludermir, "Classical and superposed learning for quantum weightless neural networks," *Neurocomputing*, vol. 75, no. 1, pp. 52–60, 2012.
[21] ——, "Training a classical weightless neural network in a quantum computer," *22nd European Symposium on Artificial Neural Networks, ESANN 2014*, 2014.

[22] P. G. dos Santos, R. S. Sousa, I. C. Araujo, and A. J. da Silva, "Quantum enhanced cross-validation for near-optimal neural networks architecture selection," *International Journal of Quantum Information*, p. 1840005, 2018.
[23] R. Jozsa and N. Linden, "On the role of entanglement in quantum-computational speed-up," in *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 459, no. 2036. The Royal Society, 2003, pp. 2011–2032.
[24] H. T. Siegelmann and E. D. Sontag, "Turing computability with neural nets," *Applied Mathematics Letters*, vol. 4, no. 6, pp. 77 – 80, 1991.
[25] R. Livni, S. Shalev-Shwartz, and O. Shamir, "On the computational efficiency of training neural networks," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 855–863.
[26] L. M. Almeida and T. B. Ludermir, "A multi-objective memetic and hybrid methodology for optimizing the parameters and performance of artificial neural networks," *Neurocomputing*, vol. 73, no. 7, pp. 1438 – 1450, 2010.
[27] C. Ding, Y. Li, L. Zhang, J. Zhang, L. Yang, W. Wei, Y. Xia, and Y. Zhang, "Fast-convergent fully connected deep learning model using constrained nodes input," *Neural Processing Letters*, Jun 2018.
[28] M. Schuld, I. Sinayskiy, and F. Petruccione, "An introduction to quantum machine learning," *Contemporary Physics*, vol. 56, no. 2, pp. 172–185, 2015.
[29] K. H. Wan, O. Dahlsten, H. Kristjánsson, R. Gardner, and M. Kim, "Quantum generalisation of feedforward neural networks," *npj Quantum Information*, vol. 3, no. 1, p. 36, 2017.
[30] Y. Cao, G. G. Guerreschi, and A. Aspuru-Guzik, "Quantum neuron: an elementary building block for machine learning on quantum computers," *arXiv preprint arXiv:1711.11240*, 2017.
[31] N. Wiebe and V. Kliuchnikov, "Floating point representations in quantum circuit synthesis," *New Journal of Physics*, vol. 15, no. 9, p. 093041, 2013.
[32] M. V. Altaisky, "Quantum neural network," Joint Institute for Nuclear Research, Russia, Technical report, 2001.
[33] R. Zhou, H. Wang, Q. Wu, and Y. Shi, "Quantum associative neural network with nonlinear search algorithm," *International Journal of Theoretical Physics*, vol. 51, no. 3, pp. 705–723, 2012.
[34] W. R. de Oliveira, "Quantum RAM based neural networks," in *ESANN'09: Advances in Computational Intelligence and Learning*, M. Verleysen, Ed., 2009, pp. 331–336.
[35] J. Preskill, "Quantum computing in the nisq era and beyond," *arXiv preprint arXiv:1801.00862*, 2018.
[36] M. Bozzo-Rey and R. Loredo, "Introduction to the ibm q experience and quantum computing," in *Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering*. IBM Corp., 2018, pp. 410–412.
[37] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan, "Training deep neural networks with 8-bit floating point numbers," in *Advances in Neural Information Processing Systems*, 2018, pp. 7686–7695.



Fernando M de Paula Neto received the Bachelor degree (cum laude) in Computer Engineering (2014) and Ph.D (2018) in Computer Science at Universidade Federal de Pernambuco UFPE, Brazil. He joined the Center of Informatics at UFPE in 2019 where is currently an Adjunct Professor. His current research interests include Quantum Computing, Quantum Machine Learning and Dynamical Systems and Chaos.

Teresa B Ludermir received the Ph.D. degree in Artificial Neural Networks in 1990 from Imperial College, University of London, UK. From 1991 to 1992, she was a lecturer at Kings College London. She joined the Center of Informatics at Federal University of Pernambuco, Brazil, in September 1992, where she is currently a Professor and head of the Computational Intelligence Group. She has published over 300 articles in scientific journals and conferences and three books in Neural Networks.



Wilson R de Oliveira received his Ph.D. (2004) in Computer Science at Universidade Federal de Pernambuco (Brazil). He was on sabbatical leave at the School of Computer Science, University of Birmingham, UK (2008).He has been working recently on Quantum Weightless Neural Networks, Categorical Quantum Mechanics and Discrete Manifolds. He joined the "Departmento de Estatística e Informàtica" at the "Universidade Federal Rural de Pernambuco" in 2000 where he is now an Associate Professor.

Adenilton J da Silva received the Licenciate degree in mathematics from Universidade Federal de Pernambuco (UFPE), Brazil and received the Ph.D. degree in Computer Science from Universidade Federal de Pernambuco, Brazil. He joined the Center of Informatics at UFPE in 2019 where is now Adjunct Professor. His current research interests include quantum computation, artificial neural networks, machine learning and hybrid neural systems.