Analysis of Quantum Neural Models

Fernando M. de Paula Neto, Adenilton J. da Silva and Teresa B. Ludermir
Universidade Federal de Pernambuco
Centro de Informática
Recife, Pernambuco 50740560
Telephone: (+5581) 2126–84302
Email: {fmpn2, ajs3,tbl}@cin.ufpe.br

Wilson R. de Oliveira
Universidade Federal Rural de Pernambuco
Departamento de Estatística e Informática
Recife, Pernambuco 52171-900
Telephone: (+5581)-33206491
Email: wilson.rosa@gmail.com

***Abstract*—On this paper, we briefly analyze and compare some models of quantum artificial neural networks. Quantum operators must be linear ones; we verify that no unitary operators are used in two models of quantum perceptron. We also analyze a model of quantum weightless neural network and a quantum complex neural network. These models have quantum architecture and learning, but we show that they use nonlinear operators in the learning process. This study, toward a comparative method, tries to clarify important aspects in models of quantum neural networks as well as understand more its operation.**

I. Introduction

Electronic components have decreased of size significantly in the last decades. As Moore's Laws proposed, the number of transistors on chip practically doubles every two years. So this decrease has created non positive perspectives to keep using the classic paradigm of electronic circuit building. The classical physic law does not preserve its operation in atomic particles individually. Meanwhile, quantum computing might be more powerful than classic computing machines, as investigated and proposed by Feynman [2]. Then, many algorithms have been proposed by scientists to provide solutions in the quantum paradigm in many of the computing fields. In the Artificial Intelligence (AI) field this trend grows up specially (a) for the importance of techniques that can analyze a huge data set automatically and cluster in related groups[1]; (b) for the medical applicability, in segmentation of images, pattern recognition and genomes clusterizing that helped to discover and prevent diseases, respectively; (c) for the robotic that is more and more present in the industries of high accuracy as in the daily human activity. Many other reasons are present in literature to encourage more study and investments in the AI.

Since 1943, when the psychologist McCulloch and the mathematical Pitts created the artificial neuron inspired in the biologic work and after the Hopfield network and backpropagation algorithm had been created, the AI field improved significantly when considerable models of Neural Network and Perceptrons were proposing.

So, this study allows to compare, associating these models and proposing a comparison of the quantum neurons in order to provide perspectives and important topics to take in consideration when a network is created to solve some problem. In the section II, there is an exploring introduction of the quantum computing topics, in section III, the operation of a perceptron and neural network in classical methods are explained and in section IV the quantum models are presented. After that, there is a study of comparison in section V towards the methods explained. A conclusion is given in section VI.

II. Quantum Computing

A quantum bit, qubit, is a complex bi-dimensional unitary vector. The vectors $|0\rangle = [1,0]^T$ and $|1\rangle = [0,1]^T$ form a computational basis. Any qubit $|\psi\rangle$ can be written as in Equation (1), where α and β are complex numbers and $|\alpha|^2 + |\beta|^2 = 1$. Tensor products are used to composed systems $|ij\rangle = |i\rangle \otimes |j\rangle$.

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

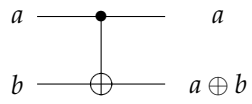
A quantum operator $\mathbf{U}$ over n qubits is a $2^n \times 2^n$ complex unitary matrix. For instance, some main operators over 1 qubit are the identity I, not $\mathbf{X}$ and Hadamard $\mathbf{H}$ operators, described in Equation (2) and Equation (3). The combinational representation of unitary operators is called quantum circuit.

$$\mathbf{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{matrix} \mathbf{I}|0\rangle = |0\rangle \\ \mathbf{I}|1\rangle = |1\rangle \end{matrix} \quad \mathbf{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{matrix} \mathbf{X}|0\rangle = |1\rangle \\ \mathbf{X}|1\rangle = |0\rangle \end{matrix} \tag{2}$$

$$\mathbf{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{matrix} \mathbf{H}|0\rangle = 1/\sqrt{2}(|0\rangle + |1\rangle) \\ \mathbf{H}|1\rangle = 1/\sqrt{2}(|0\rangle - |1\rangle) \end{matrix} \tag{3}$$

The identity operator I outputs the input; flip operator X behaves as the classical NOT on the computational basis; Hadamard transformation H generates superposition of states. The CNOT operator has 2 input qubits and 2 output qubits and flips the second qubit if the first one is 1 as show in Figure 1.

Fig. 1. CNOT operator



Parallelism is an important characteristic of quantum computing. If one has an operator over $n+1$ qubits $\mathbf{U}_f$ such that $\mathbf{U}_f|x_i,0\rangle = |x_i, f(x_i)\rangle$ for each x_i in computational basis, so one can calculate all values simultaneously applying $\mathbf{U}_f$ in the state described in Equation (4), that will result in a state described in Equation (5).

$$|\psi\rangle = \sum_{i=0}^{n-1} \alpha_i |x_i, 0\rangle \tag{4}$$

$$\mathbf{U}_f|\psi\rangle = \sum_{i=0}^{n-1} \alpha_i |x_i, f(x_i)\rangle \tag{5}$$

In a classic computing, a search about unordered arrays of m elements takes, in the worst case, m queries. On average, one finds the desired element in $0.5m$ queries. This process can be better quadratically faster in a quantum computing using the Grover's algorithm [10]. Grover's algorithm outperforms any classical algorithm and solves in $O(\sqrt{N})$ quantum steps. The iteration number T is calculated as in Equation (6), where M is the number of answers in the search space:

$$T = \left\lfloor \frac{\pi}{4}\sqrt{\frac{N}{M}} \right\rfloor \tag{6}$$

The Algorithm 1 shows this routine and M is a squared 2^n-dimensional matrix whose each entry is $1/2^n$.

Algorithm 1: Grover's algorithm

1 Initialise the system $|\psi\rangle$=$|0\rangle_n$
2 Apply the Hadamard Transform $|\psi\rangle$=$H^{\otimes n}|\psi\rangle$
3 Apply the phase inversion operation $U_f(I \otimes H)$
4 Apply the inversion about the mean operation $-I + 2M$
5 Repeat steps 3 and 4, $T = O(\sqrt{2^n})$ times.
6 Measure the state $|\psi\rangle$

III. Classic Perceptron and Neural Network

A classic Perceptron is a mathematical structure to represent one operation inspired by a biologic neuron. The result of a Perceptron is 1, or HIGH, if the intern product between the input vector and the intern weight vector is more than a known threshold value, and is 0, or LOW, otherwise.

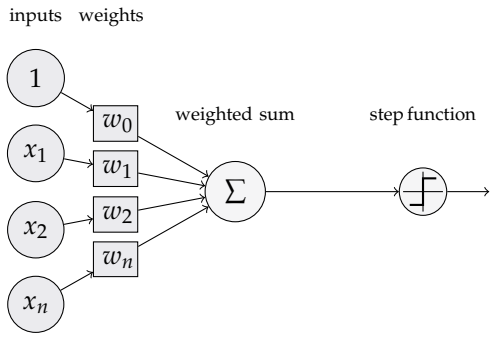


Fig. 2. Classic Perceptron

The first model proposed of Perceptron had an activation function, which processes the intern product, a step function. Other models, like Adaline, incorporated nonlinear function and increased its computational power, described in (7). It is used to classification problems and as a function approximator.

$$y = f(\sum_{i=0} x_i * w_i) \tag{7}$$

The Neural Networks are a set of Perceptrons that are connected to each other. An output of one is the entry to the other perceptron. In other words, each output is one of the inputs of another. A neural network can have more than one layer of perceptrons, increasing its computational power to solve nonlinear problems. Currently, they are used in temporal series forecast, in classification problems, pattern recognition and many other real problems.

IV. Quantum Neuron Models

The main problem of a quantum model to incorporate a neuronal function is the need of a nonlinear function that allows to compute nonlinear problems, as the classic XOR boolean operator. Quantum computing is not directly compatible with that operations, because its gates are linear, unitary and invertible. Differents solutions are given to solve this problem and one explains four of them below. Each one has a set of characteristics to solve this problem. Depending on the method chosen to build a quantum neuron maybe one needs a non-unitary quantum gate to operate it, during or after the training of the neuron.

A. AQP

The model AQP Quantum Perceptron was proposed in [7] by Altaysky. This model interacts over only one qubit of input. It outputs also a state of only one qubit.

$$|y\rangle = \hat{F} \sum_{j=1}^{n} \hat{w}_j |x\rangle_j \tag{8}$$

It is described by the Equation (8), where $\hat{F}$ can have the role of the function of activation, w_j is a 2x2 matrix

acting and $|x\rangle_j$ one of the n inputs $|x\rangle_1, ..., |x\rangle_n$. For this method, Altaiksy also proposed a learning rule that adapts the matrix of weights to a desired output, through the learning rule described in (9), where w_j is a matrix of weight, η is a real number between 0 and 1, $|d\rangle$ is the desired qubit, $|y(t)\rangle$ is the output of the operator $\hat{F}$ in Equation (8) and x_j the qubit of input.

$$\hat{w_j}(t+1) = \hat{w_j} + \eta(|d\rangle - |y(t)\rangle)\langle x_j| \tag{9}$$

This methods also implies a possible non unitary gate during or in the end of the training steps.

An example of that does not preserve unitary operators in Equation (9) can be demonstrated in one iteration of the iterative quantum learning rule. Set j=1, the weight $\hat{w_1}(t) = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$, $\eta = 0.2$, the desired output $|d\rangle = |0\rangle$, considering the input $|x_j\rangle = |1\rangle$, the network output is $-|1\rangle$, by the Equation (8). Then,

$$\begin{aligned}
\hat{w_j}(t+1) &= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} + 0.2(|0\rangle + |1\rangle)\langle 1| \\
&= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} + 0.2(|0\rangle\langle 1| + |1\rangle\langle 1|) \\
&= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} + \begin{pmatrix} 0 & 0.2 \\ 0 & 0.2 \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0.2 \\ 0 & -0.8 \end{pmatrix}
\end{aligned}$$

where one sees that $\hat{w_j}(t+1)$ is non-unitary. This implies that properties of the quantum computing are not preserved. The machine where this model will be used must consider some mode to work with nonlinearity.

B. ZQP

Zhou [3] proposed one representation of neuron given by a matrix of weights where it is updated from a training set during a training process. This algorithm tries to be an analogous method in relation to the classic one, but without the nonlinear function. The product of the matrix of weight with the input generates the quantum state output. The output is described by the Equation (10).

$$O_i = \sum_j w_{ij}\phi_j \tag{10}$$

where j = 1, 2, 3, ... , 2^n, n is a number of qubits of inputs, and w the matrix of weight 2^n x 2^n.

A quantum state of n qubits is presented to the Perceptron and there is an intern product with the weight matrix. The adaptation of the weights can turn this one nonlinear, during the training step presented below.

Its weight update function is described by the Equation (11).

$$w_{ij}^{t+1} = w_{ij}^t + \eta(|O\rangle_i - |\psi\rangle_i)|\phi\rangle_j \tag{11}$$

where w_{ij} is one position of the matrix of weights, η is a traditional learning constant that is fitting to the size of the training step, ψ_i is a i-th position of the output of the neuron, $|O_i\rangle$ the i-th qubit in the desired quantum state in the training set presented and $|\phi_j\rangle$ the j-th qubit of the input.

This architecture bounds the size of the output qubit to be the same of the input. If one desires only one qubit as answer, one solution of that is to use some of the qubit in the output state as the answer of the perceptron. Other problem is that for n qubits of inputs, the matrix of weights necessarily has n^2 entries.

It is important to mention that in training example in [3], the author has not used the correct Equation (11) to training, what is a mistake. The $|\phi\rangle_j$ is changed to $|\phi\rangle_i$ in his example and the calculus of some weight matrix is not what Zhou describes. If one corrects the use of position of the $|\psi\rangle$ during the training, the example of the paper becomes correct.

The updated matrix of the proposed method is also not necessarily unitary during or after the training step. An example is showed below:

Supposing the initial weight matrix:

$$W^0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Two pairs of input-output:

$$\left\{ \left(\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right), \left(\begin{pmatrix} 0 \\ 1 \end{pmatrix}, \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} \right) \right\}$$

Then:

$|\psi\rangle = W^0|\phi\rangle = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ where $|\psi\rangle$ is the output of the first input in the training set.

The weight upload is:

$$\begin{aligned}
w_{00}^{t+1} &= w_{00}^t + \eta(|O\rangle_0 - |\psi\rangle_0)|\phi\rangle_0 \\
&= w_{00}^1 = 1 + 1\left(\tfrac{1}{\sqrt{2}} - 1\right)1 \\
&= \tfrac{1}{\sqrt{2}}
\end{aligned}$$

In the same way:

$$\begin{aligned}
w_{01}^{t+1} &= w_{01}^t + \eta(|O\rangle_0 - |\psi\rangle_0)|\phi\rangle_1 \\
&= w_{01}^1 = 0 + 1\left(\tfrac{1}{\sqrt{2}} - 1\right)0 \\
&= 0
\end{aligned}$$

$$w_{10}^{t+1} = w_{10}^t + \eta(|O\rangle_1 - |\psi\rangle_1)|\phi\rangle_0$$

$$= w_{10}^1 = 0 + 1\left(\frac{1}{\sqrt{2}} - 0\right)1 = \frac{1}{\sqrt{2}}$$

$$\begin{aligned} w_{11}^{t+1} &= w_{11}^t + \eta(|O\rangle_1 - |\psi\rangle_1)|\phi\rangle_1 \\ &= w_{11}^1 = 1 + 1\left(\frac{1}{\sqrt{2}} - 0\right)0 = 1 \end{aligned}$$

So, the first uploading of weight matrix will finish in:

$$W^1 = \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & 1 \end{pmatrix}$$

where it is easy to check it is not a unitary matrix. Continuing the procedure of update the weights in all training set, it will be necessary to apply this matrix W^1 again as a quantum operator but it is not an unitary matrix. The access in the each position of weight matrix in the update must be considerate when this model is running, because it is not a quantum operation.

C. QNNW

In [5], Panella tells that the models proposed by Altaisky in [7] and others in [6] and [8] are not convincing because they do not use an efficient training in the Perceptron, with massive parallelism procedure. Panella also said that they do not solve basic aspects of neural computing such as non-linearity, implying the explanation as this nonlinearity works. There are two ways to solve non-linearity in QNNW, elucidated by Panella: (a) or a dissipative nonlinear gates are used or (b) modifications are made in the network preliminarily in order to result in a unitary gate. In Panella solution, the second way is used. The non-linearity is used only in the training step and this mechanism will not imply in a non-linear network after the training like the AQP [7]and ZQP [3] neurons. Panella also states that one efficient training happens only when a superposition of states is used.

This proposed model by Panella intents to solve this non-linearity problem with a training efficient.

Panella uses superposition of states to apply an exhaustive search of the optimal net through the nonlinear operators. The main idea is to represent the neuron as a string of qubits which is equivalent to a problem to solve. The general case $|\Psi\rangle$ for all the possible QNNs is represented in Equation (12), where n is the number of qubits. One can represent all the possible QNNs by the $N = 2^n$ states in superposition.

$$|\Psi\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N} \left|\Psi_1^{(k)} \Psi_2^{(k)} ... \Psi_n^{(k)}\right\rangle \tag{12}$$

where $\left|\Psi^{(k)}\right\rangle$ is the kth QNN, $k = 1, ..., N$ and $\left|\Psi_j^{(k)}\right\rangle$, j=1,...,n, the j-th qubit of its representation. If one has N_0 inputs and in one N_1 neurons layer, one will need $N_0 * N_1$ blocks of F_0, which computes the synapses. The synapses of the one neuron happen in two steps. The first one by the sum of two complex numbers of magnitude 1 and considering as its output the argument of the sum. $N_1 * N_0$ bias, θ, are considered in this step in each block of F_0. The calculus of this step (involving entries and bias) is showed with details in [5]. In the second step, the result of each F_0 block is added with the bias ξ. This sum is applied by F_1 operator, where the non-linear function is implemented and it is explained with details in [5]. This function does is not a quadratic matrix and does the role of the nonlinear function of the neuron. In the other words, its output of F_1 is the output of the neuron. The successive layers are similar to the first one.

For an example, a neuron with three inputs and two neurons in the first layer, it would have $3 \cdot 2 = 6$ blocks of F_0, then $3 \cdot 2$ also θ, ($\theta_1^{(1)}, \theta_2^{(1)}, \theta_3^{(1)}, \theta_4^{(1)}, \theta_5^{(1)}, \theta_6^{(1)}$) and two ξ, ($\xi_1^{(1)}, \xi_2^{(1)}$), as shown in Figure (3).

The efficient of a neural network given a training set is calculated as a performance $\Delta^{(p)}$, by a one block F_2, described with details in [5]. The F_2 function can be considered as a boolean function of another function F_3. F_3 is implemented supported by [9]. When an input is a $|\Psi\rangle$ that represents all QNNW in superposition, then the output of F_3 is the superposition of $\left||\Psi\rangle, \Delta^{(p)}\right\rangle$. In this state, each QNN will be entangled with its performance, for the example, $\{X^{(p)}, t^{(p)}\}$, where X is the inputs and t the desired targets. The performance is described for the equation (13), where $u^{(p)}$ is the output of the QNN.

$$\Delta = \sum_{p=1}^{N_p} \Delta^{(p)} = \sum_{p=1}^{N_p} |t^{(p)} - u^{(p)}| \tag{13}$$

In resume, the Δ makes the sum of the difference (the error) of the output and the desired state. This is done through the F_3 gate, but a value δ is considered in the place of Δ, which δ is the average of the $\Delta^{(p)}$. After that, the quantum state will be:

$$\Phi = \frac{1}{\sqrt{(2)}} \sum_{k=1}^{N} \left|\Psi^{(k)}, \delta^{(k)}\right\rangle \tag{14}$$

To determine the optimal QNNW, one needs to find the QNN that has the δ less than or equal to a given threshold. This is done by a non-linear function U_g described with details in [5]. This gate marks the states which have the previous condition (to be less or equal to a given threshould) if it exists. This state is marked in a special qubit $|c\rangle$ with the value $|1\rangle$ when the δ is less or equal to a suitable threshold; and with $|0\rangle$ otherwise. After that, the quantum state will be as showed in Equation (15).

$$\begin{aligned} \Omega = |\Phi, c\rangle = \frac{1}{\sqrt{N}} \sum_{c^k=1} \left|\Psi^{(k)}, \delta^{(k)}, c^{(k)}\right\rangle + \\ \frac{1}{\sqrt{N}} \sum_{c^k=0} \left|\Psi^{(k)}, \delta^{(k)}, c^{(k)}\right\rangle \end{aligned} \tag{15}$$

The final step is an exhaustive search which finds the QNNW which was marked with that flag $|c\rangle = 1$, that it will be the chosen network.

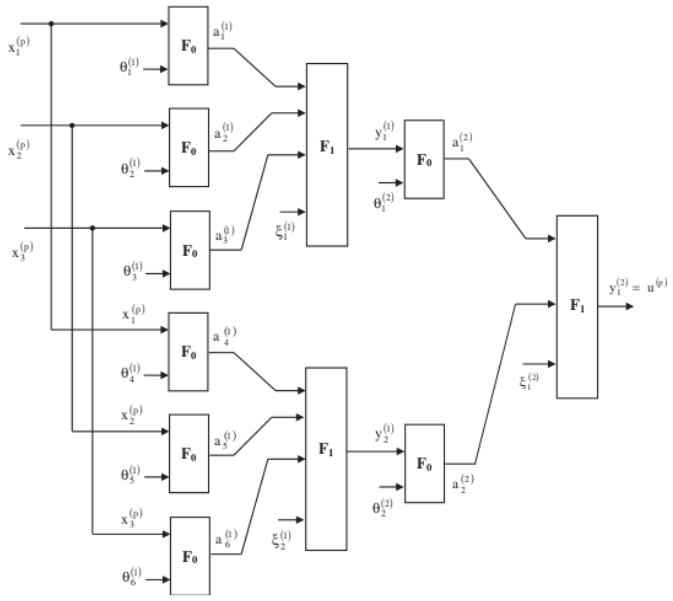


Fig. 3. Scheme of the proposed QNN by Panella in an example of three inputs and two neurons

D. qRAM Neuron

Adenilton [11] defined a quantization of a RAM based neuron described in [4]. This model does not use a non-linear activation function. The neuron works with a set of 2^n selectors, $|s\rangle$, where n is the number of qubits of the inputs $|i\rangle$, one qubit $|o\rangle$ of output and 2^n matrices (or quantum circuit) of controlled-NOT gates. In other words, there are three quantum registers $|s\rangle$, $|i\rangle$ and $|o\rangle$, where the input $|i\rangle$ chooses one qubit of the selector to apply in the controlled-NOT gates with the output qubit.

For example, given an input $|i\rangle$=$|0\rangle$, the neuron will choose the first qubit, $|s_0\rangle$, of the selector; given another one $|i\rangle$=$|1\rangle$, the neuron will select the second qubit, $|s_1\rangle$, of the selector, and so on. After this choice, the neuron applies the chosen selector with the output qubit in a controlled-NOT gate. In superposition state of input, this neuron can choose selectors in superposition, and, probably, to generate an output in superposition. The N neuron is represented by the operator in Equation (16).

$$N = \sum_{i=0}^{2^n-1} |i\rangle_n \langle i|_n A_{s_i o} \tag{16}$$

This model can be better understood with the representation in Figure (4). As one can see, the selectors $|s\rangle$ after the training step can learn a pattern, changing their values. Adenilton also suggests a configuration of a neural network using this model of neuron represented in Figure (5).

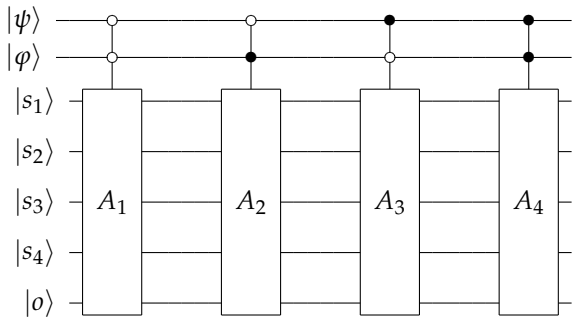


Fig. 4. qRAM node

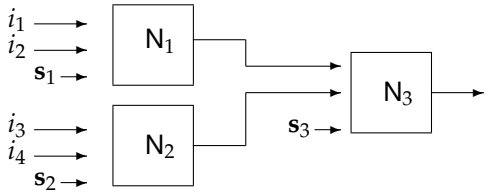


Fig. 5. Two layer qRAM network

Two training algorithms are proposed by Adenilton in [11]. The first one is called by the author as "Naive qRAM Net Learning" and it received this name because it does not explore quantum mechanics properties. This algorithm describes its check and update of the selectors $|s\rangle$, given a training pattern set. It does not make use of quantum superposition and it does a change of register qubit iteratively. More details are found in [11]. The second one, described in Algorithm 2, makes use of superposition and quantum properties. Its work is based in Grover's algorithm [10], that amplifies amplitudes of the qubits given an input $|\psi\rangle$ and its desired states $|d\rangle$.

Algorithm 2: qRAM - Superposition based Learning

1. Initialize all the qubits in register **s** with the quantum state $\mathsf{H}|0\rangle$.
2. Initialize the register **p**, **o** , **d** with the quantum state $|p\rangle = \sum_{i=1}^{p} |p_i, 0, d_i\rangle$.
3. $|\psi\rangle = \mathsf{N}|\psi\rangle$, where N is a quantum operator representing the action of the neuron.
4. Use a quantum oracle to change the phase of the states where the registers **p**, **o**, **d** $= \sum_{i=1}^{p} |p_i, d_i, d_i\rangle$
5. Apply the operator inverse to the neuron in the state $|\psi\rangle$, $|\psi\rangle = \mathsf{N}^{-1}|\psi\rangle$, to disentangle the state in the register s
6. Apply the inversion about the mean operation (see Algorithm 1) in the register **s**
7. Repeat steps 3, 4, 5 and 6 $T = \frac{\pi}{4}\sqrt{n}$ times, where n is the number of selectors of the networks.
8. Measure the register **s** to obtain the desired parameters.

The Algorithm 2 shows the procedure of initialization

of the registers and the routine that amplifies the amplitudes of the desired states of the $|s\rangle$ based in the desired outputs.

V. Comparative Method

A Quantum Neuron needs to be a universal quantum gate or quantum circuit and, not only that, it must provide an adaptive and powerful learning. This is found in the models described above, each one with its peculiarities.

Then, a simple general structure of Perceptron is presented below. Adenilton and Panella models have used algorithms that implement superposition and explicit quantum parallelism, so this simple model of training is not considered directly by them.

$$|P\rangle = \hat{F}\sum_{j}^{n}\hat{W}_j|\Psi\rangle_j \tag{17}$$

$$|W\rangle_{t+1} = |W\rangle_t + \sum_j |x\rangle_j(|\psi\rangle, |d\rangle, |\phi\rangle)\langle y|_j(|\psi\rangle, |d\rangle, |\phi\rangle) \tag{18}$$

In the equation (17), a Perceptron is defined as applications of weight operators $\hat{W}$ in inputs $|\Psi\rangle$ that can be one qubit or more than one. The AQP model is the one with the restriction that its input is only one qubit and w a 2x2 matrix. In qRAM model, the model is exactly that, if one considers that $|i\rangle_n$ and $\langle i|_n$ build a linear operator W based on extern product. In ZQP model is equal considering $\hat{F}$ as the identity operator I. Panella has proposed that the perceptron is the sum of strings of qubit and the Equation (17) also is able to represent, as the string of qubits being the operator gate W applied in the inputs ψ. In other words, the string of qubits can be generated from the product of W with the inputs.

It is important to refer that the model AQP accesses the amplitudes of the qubit states. In quantum computing, this is done by a measuring where the state is lost and it implies the collapse of the measured quantum state.

In the general learning rule (18), one needs to ensure that the adaptation of weights is function of the input and of the desired output. The qubit $|x\rangle$ and the qubit $|y\rangle$ are quantum states in function of the input of the network and of the desired output. Any operator is formed from the $|i\rangle_a$ and $\langle i|_b$ states [13] and it can generate the operators which are proposed by each model presented. It is necessary also to consider that this procedure of training will converge.

To compare the models, some important topics have been considered. In the Table (I), one sees that the method AQP and ZQP have a matrix learning, but only AQP has a matrix operation in the training. In other words, AQP is the unique to update the own matrix using matrix operations. The other models necessarily need to update qubits or values in the matrix positions, like an assignment, with no process of matrix operation. The column q-learning is set if the model uses the quantum computing properties during the training and all of the methods are signed as able, although the AQP and ZQP model can generate a nonlinear weight matrix or network. Only qRAM and QNNW models have an explicit superposition training. The qRAM has the power to be trained from a classical training.

TABLE I. Comparison of quantum neuron techniques

Model	Matrix learning	input/ output	q-learning	classical learning
AQP	X	1 qubit	X**	
qRAM		>=1 qubit	X	X
ZQP	X*	>=1 qubit	X**	
QNNW		>=1 qubit	X	

*Despite ZQP uses a matrix learning, during the training, the update in each position of the matrix is not one matrix operation, it is an individual calculus in each position. **AQP and ZQP do not have an explicit superposition training.

The models presented are not self-adaptative or self-organizing methods, they need tags or supervised learnings to adjust its intern parameters. Then all of them use supervised learning.

About non linearity, the AQP and ZQP models results in possible nonlinear quantum structures, despite the QNNW needs to use the nonlinearity in the training to solve some problems, like check strings of qubits and flip one qubit if necessary. The nonlinearity is not present in the qRAM model.

In the study of quantum computing, using a non-unitary matrix can be understood as the use of dissipative gate that not only changes the phase in the systems as also the amplitude. Although it is not considerate to some physicists as quantum computing, this nonlinear methods have increased in the last researches, as in [12].

In details about complexity, only Panella and Adenilton investigates its complexity and computational cost in their articles.

VI. Conclusion

There are many aspects to consider when an artificial neuron model is created and the choice of the technique to build the neurons will reflect in its features. This is also present in classical models. Many times, a neuron model is good to solve some problem in particular. On this paper, one can analyze some topics about the complexity, the operation of the models and to verify what the operation of each neuron struct provides. This study is limited in terms of a full hardware review but it is introductory to elucidate the variation of each one in operations, size of input and output and about the quantum properties, as parallelism and measuring. As one sees the models present different ways to work and an analysis of the models is necessary if one needs to choose a neuron to work in an application.

This study can be extended in a simulation of real databases of Artificial Intelligence field to compare these techniques in real problems to verify results in each one.

ACKNOWLEDGMENT

This work is supported by research grants from CNPq, CAPES and FACEPE (Brazilian research agencies). The author Fernando is grateful by the inspiration and motivation from the engineer and professor Helio de Oliveira, UFPE.

REFERENCES

[1] Boley, D.; Gini, M.; Gross, R.; Han, E.-H.; Karypis, G.; Kumar, V.; Mobasher, B.; Moore, J.; Hastings, K. *Partitioning-based clustering for web document categorization*, Decis. Support Syst., v. 27, n. 3, p. 329-341, 1999.

[2] Feynman, R.P. *Simulating physics with computers*, Int. J.Theor.Phys. 21(1982)467-488.

[3] R. Zhou, Q. Ding, *Quantum M-P neural network*, International Journal of Theoretical Physics 46 (2007) 3209-3215.

[4] I. Aleksander, *Self-adaptive universal logic circuits*, Electronics Letters 2 (8) (1966) 321-322

[5] M. Panella, G. Martinelli, *Neural networks with quantum architecture and quantum learning*, International Journal of Circuit Theory and Applications 39 (1) (2011) 61-77, doi:10.1002/cta.619.

[6] Lewenstein M. *Quantum perceptrons*. Journal of Modern Optics 1994; 41:2491-2501.

[7] Altaisky MV. *Quantum neural network*. ArXiv: quant.ph/0107012, 5 July 2001.

[8] Ventura D, Martinez T. *An artificial neuron with quantum mechanical properties*, Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms, Norwich, U.K., 1997; 482-485.

[9] Rieffel E, Polak W. *An introduction to quantum computing for non-physicists*, ACM Computing Surveys 2000; 32:300-335.

[10] L.K. Grover, *A fast quantum mechanical algorithm for database search*, in: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, STOC '96, ACM, 1996, pp. 212-219.

[11] Adenilton J. da Silva, Wilson R. de Oliveira, Teresa B. Ludermir *Classical and superposed learning for quantum weightless neural networks* Neurocomputing Journal, 2011, pp. 52-60.

[12] Daniel S. Abrams, Seth Lloyd *Nonlinear quantum mechanics implies polynomial-time solution for NP-complete and #P problems* Phys.Rev.Lett. 81 (1998) 3992-3995

[13] M.A. Nielsen, I.L. Chuang *Quantum Computation and Quantum Information*, Cambridge University Press, 2000.