Artificial Neural Network Implementation Using Many-Valued Quantum Computing

Anas N. Al-Rabadi and George G. Lendaris
Systems Science Ph.D. Program at Portland State University
[alrabadi@ece.pdx.edu, lendaris@sysc.pdx.edu]

Abstract

Neural network (NN) implementation using the general scheme of many-valued (MV) quantum computing (QC) is presented in this paper. The proposed method uses the many-valued orthonormal computational basis states to implement such computations. Physical implementation of NNQC is performed by controlling the potential to yield specific wavefunction as a result of solving Shrodinger's equation that governs the dynamics of QC.

1 Introduction

The notion of implementing neural networks using quantum computing has been suggested by various authors, e.g., [9][10][11][14]. To date, the work reported in such references has been confined to the case of using binary-valued quantum methods, namely, via the quantum bit, known as the qubit [5][7][13]. Researchers have succeeded in implementing many-valued logic gates using many-valued quantum systems (e.g., [12]). The present paper investigates implementing neural networks using many-valued quantum computing, based on previous work of one of the authors [1][2][3][4].

Motivations for pursuing the possibility of implementing NNs using quantum computing (QC) would include items such as (1) **power**: the fact that, theoretically, the internal computations in QC systems consume no power [7][13] (power is only consumed by the writing-into or reading-from quantum operators); (2) **size**: the current trends related to more dense hardware implementations are heading towards 1 Angstrom, at which quantum mechanical effects have to be accounted for [7][13]; and (3) **speed**: if the properties of superposition and entanglement of quantum mechanics can be usefully employed in the neural network context, significant computational speed enhancements can be expected [5][7][13].

The main contributions of this paper are:

(1) Quantum implementation of NNs using many-valued QC. This is achieved via the use of discrete-grid weight space and making an assignment of points on the grid to individual components of a many-valued orthonormal set of quantum basis states.

(2) Show the underlying mathematical methodology and formalisms for such many-valued QC of NNs.

(3) Propose a "reverse engineering" way to develop look-up tables (LUT) for potential functions associated with specified many-valued logic functions to be performed.

The remainder of this paper is organized as follows: Section 2 discusses related background in neural networks; Section 3 presents the fundamentals of QC; Section 4 introduces many-valued quantum computing (MVQC) of NNs; Section 5 introduces NN implementation using MVQC; and Section 6 provides conclusions and discusses some planned future work.

2 Neural Networks

The importance of neural networks in application is their ability to learn to perform functions in a problem domain, based on interacting with data from that domain [8]. A key role in the process is performed by the training set (a collection of input-output pairs from the problem domain). The training set can be said to provide problem domain "constraints."

The role of learning in the classical domain can be implemented in the quantum domain by the dynamics of a physical system governed by the Schrodinger equation (SE) [6]. The role of training set in classical NN learning can be implemented in the quantum domain by the potential function V (also considered as "constraints"). Thus, there is motivation to establish a mechanism for converting the training set (collection of input-output pairs) from the problem domain into an appropriate V function in the quantum domain. An approach for this is presented in Section 5.

3 Quantum Computing

QC is a method of computation that uses a dynamic process governed by the Schrodinger Equation (SE) [6][7][13]. The one-dimensional time-dependent SE (TDSE) takes the following general form [6]:

$$-\frac{(h/2\pi)^2}{2m}\frac{\partial^2|\psi\rangle}{\partial x^2}+V|\psi\rangle = i(h/2\pi)\frac{\partial|\psi\rangle}{\partial t} \qquad (1)$$

$$or \quad H|\psi\rangle = i(h/2\pi)\frac{\partial|\psi\rangle}{\partial t} \qquad (2)$$

where h is Planck's constant ($6.626\cdot 10^{-34}$ J·S), V(x,t) is the potential, m is particle's mass, *i* is the imaginary number, $|\psi(x,t)>$ is the quantum state, H is the Hamiltonian operator ($H = -[(h/2\pi)^2/2m]\nabla^2 + V$), and ∇^2 is the Laplacian operator.

While the above holds for all *physical* systems, in the quantum *computing* (QC) context, the *time-independent* SE (TISE) is normally used [6]:

$$\nabla^2\psi = \frac{2m}{(h/2\pi)^2}(V-E)\psi \tag{3}$$

where the solution $|\psi>$ is an expansion over orthogonal basis states $|\phi_i>$ defined in Hilbert space **H** as follows:

$$|\psi\rangle = \sum_i c_i|\phi_i\rangle \tag{4}$$

where the coefficients c_i are called *probability amplitudes*, and $|c_i|^2$ is the probability that the quantum state $|\psi>$ will collapse into the (eigen) state $|\phi_i>$. The probability is equal to the inner product $|<\phi_i|\psi>|^2$, with the unitary condition $\sum|c_i|^2 = 1$. In QC, a linear and unitary operator $\Im$ is used to transform an input vector of quantum bits (qubits) into an output vector of qubits [7][13].

In two-valued QC, a qubit is a vector of bits defined as follows [13]:

$$qubit\_0 \equiv |0\rangle = \begin{bmatrix}1\\0\end{bmatrix}, qubit\_1 \equiv |1\rangle = \begin{bmatrix}0\\1\end{bmatrix} \tag{5}$$

A two-valued quantum state $|\psi>$ is a superposition of quantum basis states $|\phi_i>$, such as those defined in Equation (5). Thus, for the orthonormal computational basis states {|0>,|1>}, one has the following quantum state:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{6}$$

where $\alpha\alpha^* = |\alpha|^2 = p_0 \equiv$ the probability of having state $|\psi>$ in state $|0>$, $\beta\beta^* = |\beta|^2 = p_1 \equiv$ the probability of having state $|\psi>$ in state $|1>$, and $|\alpha|^2 + |\beta|^2 = 1$. The calculation in QC for multiple systems (e.g., the equivalent of a register) follow the tensor product ($\otimes$) [1][2][3][4][13][14]. For example, given two states $|\psi_1>$ and $|\psi_2>$ one has the following QC:

$$\begin{aligned}|\psi_1\psi_2\rangle &= |\psi_1\rangle \otimes |\psi_2\rangle\\ &= (\alpha_1|0\rangle + \beta_1|1\rangle)\otimes(\alpha_2|0\rangle + \beta_2|1\rangle)\\ &= \alpha_1\alpha_2|00\rangle + \alpha_1\beta_2|01\rangle + \beta_1\alpha_2|10\rangle + \beta_1\beta_2|11\rangle\end{aligned} \tag{7}$$

A physical system, describable by the following equation [13]:

$$|\psi\rangle = c_1|Spinup\rangle + c_2|Spindown\rangle \tag{8}$$

(e.g., the hydrogen atom), can be used to physically implement a two-valued QC. Another common alternative form of equation (8) is:

$$|\psi\rangle = c_1\left|+\frac{1}{2}\right\rangle + c_2\left|-\frac{1}{2}\right\rangle \tag{9}$$

Many-valued QC can also be accomplished [1][4][12]. For the three-valued QC, the qubit becomes a 3-dimensional vector, and in general, for many-valued QC (MVQC) the qubit is of dimension "many". For example, one has for 3-state QC (in Hilbert space **H**) the following qubits:

$$qubit\_0 \equiv |0\rangle = \begin{bmatrix}1\\0\\0\end{bmatrix}, qubit\_1 \equiv |1\rangle = \begin{bmatrix}0\\1\\0\end{bmatrix}, qubit\_2 \equiv |2\rangle = \begin{bmatrix}0\\0\\1\end{bmatrix} \tag{10}$$

A three-valued quantum state is a superposition of three quantum orthonormal basis states (vectors). Thus, for the orthonormal computational basis states {|0>,|1>,|2>}, one has the following quantum state:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle + \gamma|2\rangle$$

where $\alpha\alpha^* = |\alpha|^2 = p_0 \equiv$ the probability of having state $|\psi>$ in state $|0>$, $\beta\beta^* = |\beta|^2 = p_1 \equiv$ the probability of having state $|\psi>$ in state $|1>$, $\gamma\gamma^* = |\gamma|^2 = p_2 \equiv$ the probability of having state $|\psi>$ in state $|2>$, and $|\alpha|^2 + |\beta|^2 + |\gamma|^2 = 1$. The calculation in QC for many-valued multiple systems follow the tensor product in a manner similar to the one demonstrated for two-valued QC [1][2][4].

A physical system comprising trapped ions under multiple-laser excitations can be used to reliably implement MVQC [12]. A physical system in which an atom (particle) is exposed to a specific potential field (function) can also be used to implement MVQC (two-valued being a special case) [13]. In such an implementation, the (resulting) distinct energy states are used as the orthonormal basis states. The latter is illustrated in Example 1 below.

Example 1. We assume the following *constraints*: (1) spring potential $V(x) = (1/2)\,kx^2$, where m is a particle, $k = m\omega^2$ is spring constant, and ω is angular frequency ($= 2\pi\cdot$frequency), and (2) boundary conditions. Also, assuming the solution of Equation (3) for these constraints is of the following form (i.e., the Gaussian function),

$$\psi(x) = Ce^{-\alpha\frac{x^2}{2}}$$

where $\alpha = m\omega/(h/2\pi)$. The general solution for the wavefunction $|\psi>$ (for a spring potential) is [6]:

$$C = \left[\frac{\alpha}{\pi}\right]^{1/4}\frac{1}{\sqrt{2^n n!}}H_n(\sqrt{\alpha}x)$$

where $H_n(x)$ are the Hermite polynomials. This solution leads to the sequence of evenly spaced energy levels (eigenvalues) E_n characterized by a quantum number (n) as follows:

$$E_n = (n+\frac{1}{2})(h/2\pi)\omega$$

The distribution of the energy states (eigenvalues) and their associated probabilities are shown in Figure 1.

4 Many-Valued Quantum Implementation of Neural Networks: Methodology and Notation

A classic shortcoming of single-neuron neural networks (e.g., perceptron, Adeline) is their inability to implement the XOR function [8]. In [14] on the other hand, a

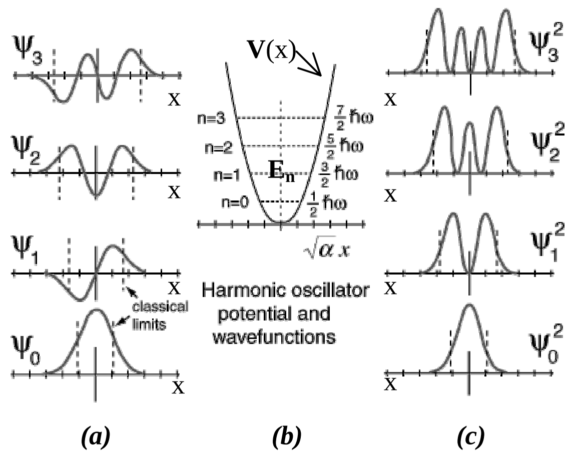


Figure 1. Harmonic oscillator potential and wavefunctions: (a) wavefunctions for various energy levels (subscripts), (b) spring potential V(x) and the associated energy levels E_n, and (c) probabilities for measuring particle (m) in each energy state (E_n).

single quantum neuron is shown to be capable of solving this not-linearly separable (NLS) function. In this section, we develop a methodology and formalisms for dealing with many-valued logic functions using many-valued quantum neurons. The notion of linearly separable and not-linearly separable mappings in the two-valued context generalizes to the many-valued case.

We describe the notion of associating a quantum state to a point in the weight space of a neural network (NN) using a two-weight NN as an example (extension to any number of weights is straightforward).

- Assume each weight can take a finite (discrete) set of values.
- Form a 2-dimensional grid of all possible combinations of weight value pairs (2-tuples).
- Assign each of the grid points (i.e., each 2-tuple) to be a quantum basis state (in quantum state space).
- If each of the two weights (w_1 and w_2) can take *m* values, then there will be m^2 quantum basis states, each with dimension m^2 (to yield an orthonormal basis set).
- Let

$$\vec{w}_{ij} = \begin{bmatrix} w_{1i} \\ w_{2j} \end{bmatrix}, \qquad i, j = 0,1,2,\ldots,m-1 \tag{11}$$

  represent the m^2 points in the 2-dimensional weight space, where the *i,j* are position indices for the vector w_{ij} and the components of w_{ij} are weight values at the corresponding positions (*i,j*).
- Then define

$$\vec{w}^{2,m} = \begin{bmatrix} \vec{w}_{00}{}^T \\ \vec{w}_{01}{}^T \\ \ldots \\ \vec{w}_{0,m-1}{}^T \\ \vec{w}_{10}{}^T \\ \ldots \\ \vec{w}_{m-1,m-1}{}^T \end{bmatrix} \tag{12}$$

where the superscript refers to 2 dimensions, with m (discrete) values in each dimension. To reference a subset of all these possibilities, an appropriate subscript may be provided.

For example, by letting each weight take three values from the set {a,b,c} where a, b, c are any discrete real values (i.e., m = 3) then one would have nine grid points {00,01,02,10,11,12,20,21,22}, and Equation (12) becomes:

$$\vec{w}^{2,3} = \begin{bmatrix} w_{00}{}^T \\ w_{01}{}^T \\ w_{02}{}^T \\ w_{10}{}^T \\ w_{11}{}^T \\ w_{12}{}^T \\ w_{20}{}^T \\ w_{21}{}^T \\ w_{22}{}^T \end{bmatrix} = \begin{bmatrix} aa \\ ab \\ ac \\ ba \\ bb \\ bc \\ ca \\ cb \\ cc \end{bmatrix}$$

Based on Equation (10), the following nine ternary orthonormal computational basis states for the MV quantum space are obtained [1][2][4][13]:

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{bmatrix}1\\0\\0\end{bmatrix} \otimes \begin{bmatrix}1\\0\\0\end{bmatrix} = [1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$$

$$|01\rangle = |0\rangle \otimes |1\rangle = \begin{bmatrix}1\\0\\0\end{bmatrix} \otimes \begin{bmatrix}0\\1\\0\end{bmatrix} = [0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$$

$$|02\rangle = |0\rangle \otimes |2\rangle = \begin{bmatrix}1\\0\\0\end{bmatrix} \otimes \begin{bmatrix}0\\0\\1\end{bmatrix} = [0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0]^T$$

$$|10\rangle = |1\rangle \otimes |0\rangle = \begin{bmatrix}0\\1\\0\end{bmatrix} \otimes \begin{bmatrix}1\\0\\0\end{bmatrix} = [0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0]^T$$

$$|11\rangle = |1\rangle \otimes |1\rangle = \begin{bmatrix}0\\1\\0\end{bmatrix} \otimes \begin{bmatrix}0\\1\\0\end{bmatrix} = [0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0]^T$$

$$|12\rangle = |1\rangle \otimes |0\rangle = \begin{bmatrix}0\\1\\0\end{bmatrix} \otimes \begin{bmatrix}0\\0\\1\end{bmatrix} = [0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]^T$$

$$|20\rangle = |1\rangle \otimes |0\rangle = \begin{bmatrix}0\\0\\1\end{bmatrix} \otimes \begin{bmatrix}1\\0\\0\end{bmatrix} = [0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0]^T$$

$$|21\rangle = |2\rangle \otimes |1\rangle = \begin{bmatrix}0\\0\\1\end{bmatrix} \otimes \begin{bmatrix}0\\1\\0\end{bmatrix} = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0]^T$$

$$|22\rangle = |2\rangle \otimes |2\rangle = \begin{bmatrix}0\\0\\1\end{bmatrix} \otimes \begin{bmatrix}0\\0\\1\end{bmatrix} = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1]^T$$

Note that this has resulted in an orthonormal basis set. One can perform MVQC by making the following assignments between the weight space and the quantum space:

MV Weight Space **MV Quantum Space**

$$\vec{w}^{2,3} = \begin{bmatrix} aa \\ ab \\ ac \\ ba \\ bb \\ bc \\ ca \\ cb \\ cc \end{bmatrix} \quad \begin{array}{c} \longrightarrow \\ \longleftarrow \end{array} \quad \left|\vec{w}^{2,3}\right\rangle = \begin{bmatrix} |00\rangle \\ |01\rangle \\ |02\rangle \\ |10\rangle \\ |11\rangle \\ |12\rangle \\ |20\rangle \\ |21\rangle \\ |22\rangle \end{bmatrix} \qquad (13)$$

Using the notation of Equation (7), the above may be written as follows [1][2][4][13]:

$$\begin{aligned} |\psi_1\psi_2\rangle &= |\psi_1\rangle \otimes |\psi_2\rangle \\ &= (\alpha_1|0\rangle + \beta_1|1\rangle + \gamma_1|2\rangle) \otimes (\alpha_2|0\rangle + \beta_2|1\rangle + \gamma_2|2\rangle) \\ &= \alpha_1\alpha_2|00\rangle + \alpha_1\beta_2|01\rangle + \alpha_1\gamma_2|02\rangle + \beta_1\alpha_2|10\rangle + \\ &\quad \beta_1\beta_2|11\rangle + \beta_1\gamma_2|12\rangle + \gamma_1\alpha_2|20\rangle + \gamma_1\beta_2|21\rangle + \gamma_1\gamma_2|22\rangle \end{aligned} \qquad (14)$$

Note that each component of the tensor product is associated with a product of two probabilities.

The coefficients of the quantum basis functions in Equation (4) (probabilities) are the system parameters, obtained by solving the waveequation *with the specified potential function V applied*. We note that different V's will (normally) result in different solutions (i.e., different probabilities) for each of the quantum basis states. Upon measurement of an observable variable in a physical quantum implementation, by definition, the highest probability state is the most likely one to occur. In the context of neural networks (NNs) with an assignment such as the one given in Equation (13), each basis state corresponds to a particular combination of weight values. These weight values determine the mapping performed (e.g., logic function) by the NN. See Figure 2.

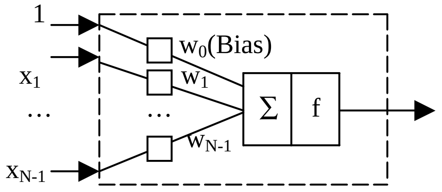


Figure 2. A simple neuron.

where f is the activation (transfer) function, and

$$y = f\left(\sum_{i=0}^{N-1} w_i x_i\right)$$
$$= f\,(w_0 \cdot 1 + w_1x_1 + w_2x_2 + \ldots + w_{N-1}x_{N-1}) \qquad (15)$$

and f can be an appropriate mapping such as threshold function, sigmoid, etc [8].

The manner in which the MVQC is implemented for neural computing (NC) is illustrated in Figure 3.

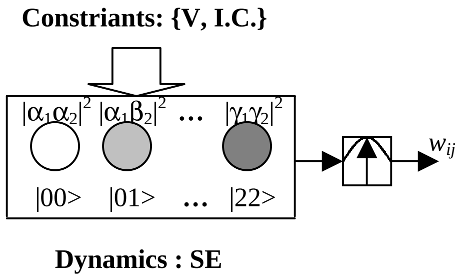
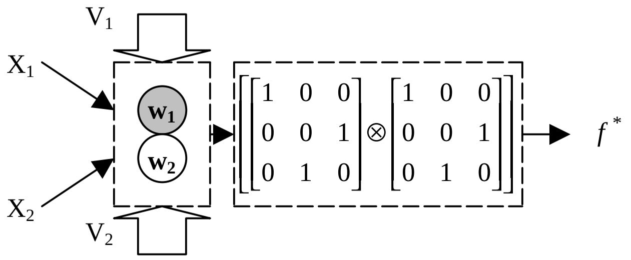


Figure 3. MVQC scheme to implement a NN using a ternary 2-qubit QC system. (All possible quantum states are shown in different colors.)

Example 2. For a quantum neuron, let the following unitary ternary quantum operator **A** [1][2][4] perform a function analogous to the activation function (AF) and summing junction (SJ) in classical artificial neurons.

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Let us denote a ternary 2-weight quantum neuron as in Figure 4:



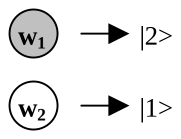
$$\left[\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}\right]$$

Figure 4. Ternary quantum neuron.

Then, for instance, for binary inputs $\{x_1,x_2\}$, the MVQC would proceed as follows to produce the following ternary function f^*:

X1	X2	f^*
0	0	0
0	1	2
1	0	0
1	1	2

The quantum weights will be determined by a suitable learning algorithm utilizing the operator **A** (e.g., the algorithm using bipolar quantum Fourier operater in [14]), which is equivalent to solving the TISE with an appropriate potential V. In the notation of Figure 4, an example result would be:



where in the MV quantum space:

$$|w_1 w_2\rangle \to |21\rangle = \begin{bmatrix}0\\0\\1\end{bmatrix} \otimes \begin{bmatrix}0\\1\\0\end{bmatrix} = \begin{bmatrix}0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\end{bmatrix}^T$$

Then the MVQC (in Figure 4) is performed in the following manner: the matrix of inputs $\{x_1,x_2\}$ is transformed, before being processed by the activation function (AF), to a new matrix of inputs by multiplying the set of inputs by the values of the corresponding weights $\{w_1 = 2,\ w_2 = 1\}$ as follows:

$$[x_1\ \ x_2] \to [w_1x_1\ \ w_2x_2]$$

$$\begin{bmatrix}0 & 0\\0 & 1\\1 & 0\\1 & 1\end{bmatrix} \to \begin{bmatrix}0 & 0\\0 & 1\\2 & 0\\2 & 1\end{bmatrix}$$

Encoding the new matrix of inputs in the MV quantum space (**H**) will lead to:

$$\begin{bmatrix}0 & 0\\0 & 1\\2 & 0\\2 & 1\end{bmatrix} \to \begin{bmatrix}|00\rangle\\|01\rangle\\|20\rangle\\|21\rangle\end{bmatrix} = \begin{bmatrix}[1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0]^T\\ [0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0]^T\\ [0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0]^T\\ [0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0]^T\end{bmatrix}$$

By using the 2-qubit ternary operator:

$$[A]\otimes[A] = \left[\begin{bmatrix}1 & 0 & 0\\0 & 0 & 1\\0 & 1 & 0\end{bmatrix} \otimes \begin{bmatrix}1 & 0 & 0\\0 & 0 & 1\\0 & 1 & 0\end{bmatrix}\right] = \begin{bmatrix}\begin{bmatrix}1 & 0 & 0\\0 & 0 & 1\\0 & 1 & 0\end{bmatrix} & & \\ & & \begin{bmatrix}1 & 0 & 0\\0 & 0 & 1\\0 & 1 & 0\end{bmatrix}\\ & \begin{bmatrix}1 & 0 & 0\\0 & 0 & 1\\0 & 1 & 0\end{bmatrix} & \end{bmatrix}$$

the matrix of the output functions will be obtained from the matrix of the weighted inputs as follows:

$$\begin{bmatrix}\begin{bmatrix}1 & 0 & 0\\0 & 0 & 1\\0 & 1 & 0\end{bmatrix} & & \\ & & \begin{bmatrix}1 & 0 & 0\\0 & 0 & 1\\0 & 1 & 0\end{bmatrix}\\ & \begin{bmatrix}1 & 0 & 0\\0 & 0 & 1\\0 & 1 & 0\end{bmatrix} & \end{bmatrix} \begin{bmatrix}[1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0]^T\\ [0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0]^T\\ [0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0]^T\\ [0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0]^T\end{bmatrix} =$$

$$\begin{bmatrix}[1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0]^T\\ [0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0]^T\\ [0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0]^T\\ [0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0]^T\end{bmatrix},$$

but in MV quantum space, the matrix of outputs correspond to the following values:

$$\begin{bmatrix}1\\0\\0\\0\\0\\0\\0\\0\\0\end{bmatrix} \begin{bmatrix}0\\0\\1\\0\\0\\0\\0\\0\\0\end{bmatrix} \begin{bmatrix}0\\0\\0\\1\\0\\0\\0\\0\\0\end{bmatrix} \begin{bmatrix}0\\0\\0\\0\\0\\1\\0\\0\\0\end{bmatrix} = \begin{bmatrix}|00\rangle & |02\rangle & |10\rangle & |12\rangle\end{bmatrix} \to |f_1 f^*\rangle$$

Then by measuring the second output one obtains the function f^*. (Note that in this example another function f_1 is naturally obtained, and thus this adds a possibility of utilizing such additional output in a separate computation.) The generation of other many-valued logic functions, at the output of the quantum neuron in Figure 4, is performed using the same topology and same AF (and SJ) by changing the values of the weights.

Other varieties of quantum operators from [1][2][4] could be used as well to perform the functionality of the AF (and SJ) in Figure 4. For instance, one could use the *quantum* Chrestenson operator defined as [1][2][4]:

$$C^{(3)}_{(1)} = \frac{1}{\sqrt{3}}\begin{bmatrix}1 & 1 & 1\\1 & d_1 & d_2\\1 & d_2 & d_1\end{bmatrix}$$

where the superscript indicates the radix, the subscript indicates number of variables, and d_1 and d_2 are complex numbers. The quantum Chrestenson operator used here is the quantum many-valued Fourier operator (which is equivalently called the quantum many-valued Walsh-Hadamard operator), which is the generalization (extension) of the quantum bipolar Fourier operator (which is equivalently called the quantum (two-valued) Walsh-Hadamard operator) [4][14]. Consequently, the (quantum) learning algorithm proposed in [14] for the quantum Walsh operator could be extended to be used for the quantum Chrestenson operator as well.

5 Further NN Implementations Using MVQC

As noted earlier, the quantum analog of a training set in the classical NN context is the potential function V, and the quantum analog for the training process are the dynamics described by the SE. An approach to implement a quantum NN suggested here is as follows (cf. Figure 5a): (1) specify a set of functions, F_i, and train a separate neural network (in the first stages of this work, think in terms of a single-neuron NN, e.g., perceptron) for each function; (2) construct a table that associates the trained NN weight vector for each function F_i; (3) construct a separate wave-function ψ_i in the MV quantum space for each F_i such that its highest probability is at the weight vector in the table, and relatively low at all other weight values as illustrated in Figure 5b; (4) substitute this ψ_i into the TISE and solve for V_i. After the above information has been generated and tabulated (as a look-up table) as indicated in Figure 5c, one could implement a full quantum NN as shown in Figure 6. In this more general case, we specify a single V for an

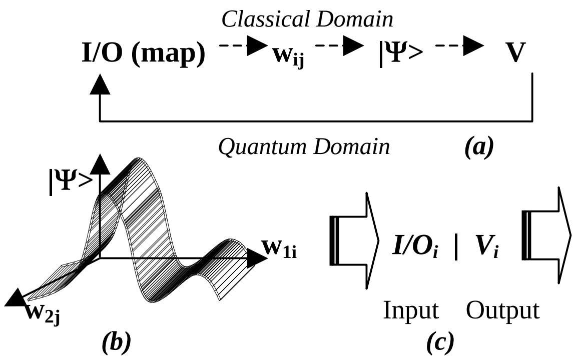


Figure 5. (a) Possible MVQC strategy to implement a NN, (b) MV Quantum-Weight space to obtain |Ψ>, and (c) Look-Up-Table to implement a NN for all logic functions.

entire weight vector going into a single quantum neuron (corresponding to a single neural element of a NN). Such a quantum neuron (QN) is here represented as shown in Figure 6a. A full network would be a collection of such QNs, connected in a specified topology, as in Figure 6b.

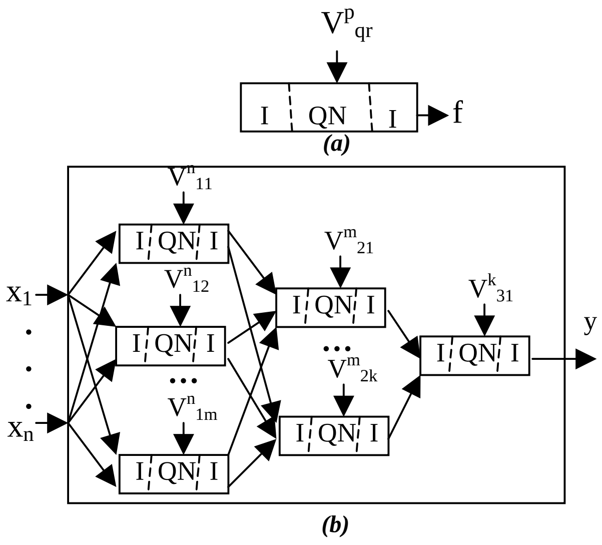


Figure 6. An MVQC implementation of a NN: (a) a quantum neuron (QN), which is a dynamical system governed by TISE constrained by V^p_{qr}, where I is the interface mechanism, superscript p is the number of incoming weights, subscript q is the layer number, and subscript r is the element number in layer q, and (b) a 3-layer NN.

6 Conclusions and Future Work

In this paper, we proposed a methodology of implementing a NN using a many-valued quantum computation (MVQC). This method uses the encoding of many-valued orthonormal computational basis states in the quantum space to be the weights in a NN. The potential plays the role of a training (I/O) set and the dynamics of the solution of SE to be the training process. Future work will involve (1) simulations for various designs of potential distributions that correspond to specific logic functions; (2) determine MVQC equivalents of supervised, reinforcement, and unsupervised learning strategies; and (3) for storing given number of patterns $\mathbf{S_i}$ in Auto-Associative memory (as in a Hopfield NN) where i = 1, 2, …, N and the pattern vector $\mathbf{S_i}$ is of dimension D. This is done conceptually as follows:

. construct a *state-space grid* (equivalent to weight-space grid discussed earlier). Each point on the grid corresponds to a specific pattern $\mathbf{S_i}$.

. Design a wavefunction ψ_i for each given pattern to be stored. Then solve the TISE for the corresponding potential function V_i.

. For a query that is a "dirty" version of a stored pattern $\mathbf{S_i^*}$, construct a corresponding ψ_i^* and V_i^*, where the designed ψ_i^* corresponds to the query pattern, and V_i^* is obtained by solving the TISE.

. If the original ψ_i was crafted such that probability is maximum at $\mathbf{S_i}$ and gradually decreases for nearby patterns, then the application of V_i^* should yield the quantum state $\mathbf{S_i}$ (the "clean"/complete pattern).

7 References

[1] A. N. Al-Rabadi, "Synthesis and Canonical Representations of Equally Input-Output Binary and Multiple-Valued Galois Quantum Logic: Decision Trees, Decision Diagrams, Quantum Butterflies, Quantum Chrestenson Gate, and Multiple-Valued Bell-Einstein-Podolsky-Rosen Basis States," *Technical Report #2001/007*, ECE Dept., Portland State University, 22 Aug. 2001.

[2] A. Al-Rabadi, L. W. Casperson, M. Perkowski, and X. Song, "Multiple-Valued Quantum Logic," Booklet of the 11th Post-Binary Ultra Large Scale Integration (ULSI)'2002 workshop, pp. 35-45, Boston, Massachusetts, 15 May 2002.

[3] A. N. Al-Rabadi, L. W. Casperson, M. Perkowski, and X. Song, "Canonical Representations for Two-Valued Quantum Computing," International Workshop on Boolean Problems (WBP)'2002, pp. 23-32, Freiberg, Germany, 19-20 Sep. 2002.

[4] A. N. Al-Rabadi, *Novel Methods for Reversible Logic Synthesis and their Application to Quantum Computing*, Ph.D. Dissertation, ECE Dept., PSU, Portland, Oregon, Fall 2002.

[5] D. Deutsch, "Quantum Computational Networks," in *Proc. Royal Soc. of London A*, vol. 425, pp. 73-90. 1989.

[6] P. Dirac, *The Principles of Quantum Mechanics*, first edition, Oxford University Press, 1930.

[7] R. Feynman, *Feynman Lectures on Computation*, Addison Wesley, 1996.

[8] S. Haykin, *Neural Networks*, 2nd Edition, Prentice-Hall, 1999.

[9] S. Kak, "Quantum Neural Computing," *Advances in Imaging and Electron Physics*, vol. 94, pp. 259-313, 1995.

[10] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial Neural Network Methods in Quantum Mechanics," *Comp. Phys. Commun.*, vol. 104, pp. 1-14, 1997.

[11] T. Menneer, *Quantum Artificial Neural Networks*, Ph.D. Dissertation, University of Exeter, UK, May 1998.

[12] A. Muthukrishnan and C. R. Stroud, "Multivalued Logic Gates for Quantum Computation," *Phy. Rev. A*, V. 62, 2000.

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

[14] D. Ventura, *Quantum and Evolutionary Approaches to Computational Learning*, Ph.D. Dissertation, Computer Science Department, Brigham Young University, 1998.