A note on Quantum Neural Networks

MSc. Complex Systems and Networks, AUTh

Σωτήριον έτος 2018

Σωτήριον έτος 2018

**Balas George**

**Kofterou Maria**

**Supervisor: Antoniou Ioannis**

Contents

[Abstract 3](#_Toc517900996)

[1. Introduction 4](#_Toc517900997)

[1) Interpretation of the activation function as measurement 5](#_Toc517900998)

[2) Interacting Quantum Dots 6](#_Toc517900999)

[3) Quantum Neural Circuits 6](#_Toc517901000)

[4) Quantum Associative memory models 7](#_Toc517901001)

[5) Quantum Perceptrons 8](#_Toc517901002)

[2. Quantum Perceptron 10](#_Toc517901003)

[Autonomous Quantum Perceptron 11](#_Toc517901004)

[Quantum Perceptron with Unitary Weights 13](#_Toc517901005)

[Example: The XOR function 15](#_Toc517901006)

[3. Quantum Feedforward Neural Network 17](#_Toc517901007)

[Feed forward QNN using ancilla qubits 17](#_Toc517901008)

[Feedforward QNN with Layer Transition Matrix 19](#_Toc517901009)

[4. Quantum Associative Memory Models 22](#_Toc517901010)

[5. Quantum Learning[14] 28](#_Toc517901011)

[6. Conclusion 30](#_Toc517901012)

[Appendix 31](#_Toc517901013)

[Singular Value Decomposition of a non- square matrix 31](#_Toc517901014)

[Hopfield Neural Network with Associative Memory 31](#_Toc517901015)

[Quantum World Interpretations[7] 32](#_Toc517901016)

[Modern version of double-slit experiment[13] 33](#_Toc517901017)

[Bibliography 35](#_Toc517901018)

# Abstract

This essay constitutes a humble approach on the development of Quantum Neural Networks. Facing the beginning of what seems to be the era of Quantum technology revolution and realizing the tremendous potentials that arise, while living in the years of machine learning, and neural networks dominance in the field of artificial intelligence, pattern recognition, big data and information process as well as many other domains that require computational efficiency, we realized the powerful combination of these two scientific fields.

In the following pages we shall explore the basic approaches towards the definition, creation and realization of a Quantum Neural Network that combines the parallel computational dynamics of a Neural Network with the revolutionary concepts of superposition, interface and entanglement provided by quantum theory. A brief summary of the basic approaches from 1995 when the idea was conceived, till the great steps made today towards the realization of some of them, are included. The basic problems that arise from the contradictive fundaments of the two theories are discussed and several clear and concrete algorithms that made it to simulations are presented.

# 1. Introduction

With the process and innovations in Quantum Mechanics stamping on the beginning of 21st century, and the development of Neural Networks revolutionizing Machine Learning techniques and being established as the most powerful computational models used so far, the idea of a Quantum Neural Network, that is in a loose definition the set of models, systems or devices that combine Quantum Theory features with Neural network properties arises.

From Kak’s[1] first idea of a “quantum neural computation” in 1995 till today however, a strict definition of Quantum Neural Networks (QNN) has not been established. There are several different approaches during these years, some of them paying more respect to the resemblance of a QNN to a classical NN, while others using the neural networks in a more loose sense, and various proposals on how these ideas could be applied, but the most realistic and concrete steps towards a complete and applicable QNN started to form in the last 5 years.

Besides loose definition and the decoherent approaches though, it is a common belief that a QNN could overcome the many difficulties of a NN, such as absence of concrete algorithms and rules to specify optimal architectures and parameters, limited memory capacity and time-consuming training. Combining the parallel NN’s computation and learning, therefore generalization potentials, with quantum computational dynamics based on superposition, it will create new unprecedented abilities in the Quantum Computing, unexpected possibilities in information processing, pattern recognition, associative memory with exponential capacity and solutions in classically intractable problems.

Before briefly explore the basic approaches of QNN’s that lead us to today, when people invest in the “QNN revolution era”, we should first face the **three major problems** faced when one dreams to create a model with both the non linear and dissipative dynamics of a NN and the unitary, linear dynamics of Quantum Computing.

1) A fundamental component of a NN is the activation functions of each neuron. Activation functions are almost exclusively non linear, and are used as the attractors that map the weighted input of a neuron to the output of this neuron. This nonlinearity comes to total contrast with the need for linearity a Quantum system requests, as linear superposition plays an important role. A Quantum system is a probabilistic, not a deterministic system and in order for probabilities to be preserved linearity, and even more unitarity is a principle.

2) The dissipative nature of the deterministic Neural Network, where once you move to the next iteration and the weight values are updated, there is no way to turn back, is incompatible with the time symmetry unitary operators ensure.

3) Neural networks’ training is based on the difference between NN’s output and the target value (supervised learning). The detection of this difference requires a measurement. A measurement’s effect on a Quantum system however would cause the system’s superposition to collapse, basically transforming it to a classical system.

In the first , at least to our knowledge, complete systematic review on QNN , from Schuld, Sinayskiy and Petruccione[2], in an effort to cope with the unestablished and decoherent definition of QNN, the authors introduce **three basic requirements** for a meaningful model:

1) The initial state of the quantum system should be able to encode any binary string of length N, the input, and produce a stable output by encoding the one among all possible output stated, which is closest to the input by some distance measure.

2) The QNN should reflect a minimum comparison to the basic computational mechanisms of a NN.

3) In order to be quantum, the system should evolve with respect to quantum effects such as superposition, entanglement and interference as well as be consistent with the Quantum principles.

Since the basic problems and requirements of a QNN are now stated, a brief summary of the different approaches towards QNN’s shall be done.

## 1) Interpretation of the activation function as measurement

As mentioned above, Kak attempted the first approach of a QNN, by interpreting the weight matrix as an operator, with eigenvectors the (states of binary encodings of the) input and unit eigenvalue. He proposed the update of the network to be a quantum measurement that selects the eigenstates of the system. However he had no point to make on the inability of a nonlinear activation function to be incorporated in the quantum system.

Shortly after, Meneer and Nayanam in a technical report in the University of Exeter, inspired by Everett’s many-worlds interpretation of the quantum world , visionised a QNN as a superposition of classical MLP’s , each one trained to store only one pattern, and the quantum state of the QNN shall be given by the weight vector. In order to retrieve a pattern, the measurement will cause a collapse of the superposition to the one state that corresponds to the component (MLP) that stored the patterns which bear the maximum resemblance to the one that is to be classified. This idea, developed further in the scope of another approach, Quantum Associative memory models and will be explicitly discussed.

Under the same scope, Mitja Perus[3] compared the neuron’s input-to-output function of weighted sums, without an activation function or a threshold, with Green’s function that describes the Time-evolution of a quantum state.

 (1.1)

He also noted that the projection operator, G, utilized in the equation shows an analogy to the Hebbian learning rule. He also imagined the pattern recall from the system as a collapse of the wave function describing the system, that is the superposition. His idea formed the baseline of the Quantum Dots approach, presented next.

## 2) Interacting Quantum Dots

The first complete proposal of a QNN came in 2000 from Elizabeth Behrman and her coworkers [4], who developed Perus idea. It is a Time-array Neural Network of 1 quron that consists of 1 qubit propagated. Green’s function formulated by Feynman path integral forms the input-to-output function of the “guron” (the quantum neuron) and the weights are engineered by the system’s interaction with the environment and updated based on a common backpropagation with gradient descent rule. The physical implementation proposed for this model is a quantum dot interacting with photons. The main problem of this approach is, once again, the nonlinearity that arises from the nonlinear kinetic energy of the Potential in input-to-output function and the description of system’s time evolution through an exponential function. The researchers failed to propose a solution to this problem.

## 3) Quantum Neural Circuits

This approach focuses on the Quantum Computing perspective and among all approaches is the one that has the minimum connection to the NN’s basic concepts. Gupta and Zia [5], inspired by Deutsch algorithm, view the construction of Quantum Circuits from a Neural Network perspective,(or vise versa) and proposed a QNN as a quantum circuit where each quantum operation is executed by U, a unitary operator that mimics the weighted input of a neuron, enables entanglement. U is followed by the application of a non-linear operator D, which they named dissipative operator that maps the state amplitudes, corresponding to a threshold, to either 0 or 1.It s a “constructive” operator, depending only on amplitudes and not phases, that results in the transformation of general states in a single stable state. D acts like the activation function of a classical perceptron neuron, determining whether the neuron is active or not.

 (1.2) where,

A and A’ are the probability amplitudes before and after the application of D, d is the threshold, m is the number of qubits D is applied on – and therefore ket consists of m 0- and c is a constant that encodes 1 in binary system.

Due to D’s non-linearity, the gate cannot be realized in an isolated system where only unitary gates are permitted. The authors adopted a different approach, stating non-linearity in quantum systems it is not forbidden, as if we take into account a quantum system that interacts with the environment, the effects can be measured by the projection of the combined system’s state vector A into the subspace of the quantum system. They stated this results to a nonlinear evolution equation for A, implying that an Open Quantum System is an approach that fits better to their model.

However this idea found no solid ground for implementation, as authors failed to define D for any input state besides |00..0> , choosing to discard all other input states, sending their qubits to a “sink gate”. They also did not discuss the case were D is applied to a state that equals to the threshold. They also did not provide any example of a quantum system that could realize their model.

## 4) Quantum Associative memory models

The most reflective proposal of this approach is probably Martinez and Ventura’s [6] and later on Ventrura and Ezhov’s [7] ideas, based on Everett’s many-worlds universe interpretation (see appendix) once again, to incorporate quantum computing algorithms, specifically Grover’s algorithm, in an effort to create a quantum model that simulates the property of associative memory a Hoppfield NN (see appendix) has. The main advantage of a Quantum Associative memory compared to Hoppfield’s classical associative memory, is that theoretically 2N patterns require n+1 qubits to be stored (a 2n+1 qubit system), instead of the classic memory capacity of n/4ln(n). Also, it will be demonstrated later how quantum associative memory does not suffer from spurious and false memories adding noise to the pattern recognition process. This approach will be discussed in detail later on, but a brief outline is:

1) A quantum circuit with associative memory is created with the train inputs, binary encoded as qubits.

2) The goal is when a new input that corresponds to a sample to be classified, is inserted in the quantum circuit, the output of the circuit to be a state corresponding to the one pattern, among all in the memory, that is closest to the input according to Hamming distance.

3) A superposition of all memorized states (each one for a pattern) is created, let’s say:

 (1.3),

P the number of different patterns , the state corresponding to p-th pattern expressed in the computational basis of N-dimensional Hilbert space, N the number of qubits needed for the binary encoding of the patterns and .

4) A computational quantum algorithm, like Grover’s search algorithm, is utilized to find the most suitable for our input sample state, among all states of |M>.

5) Finally the final measurement retrieves the desired output, which is the appropriate pattern’s state, with the highest probability.

\*As the result is probabilistic and therefore the measurement destroys superposition |M>, the construction of a number of duplicates of |M> is required.

## 5) Quantum Perceptrons

A perceptron neuron, based on binary McCulloch-Pitts neurons, is the most basic NN model, and the fact that neuron’s output reduces the input in two stages, “active and resting” comes to a perfect analogy with the qubit’s binary nature. Therefore, the idea of a quantum perceptron, a “quron” , came quite naturally from Altaisky [8], who first introduced this idea. This became the most famous approach of QNN, utilizing components from all the above approaches, to end up to the formation of Quantum Feedforward MLP (Multilayer Perceptron) NNs and being the closest to realization model our research ended up with.

This approach will also be furtherly discussed in this essay, and numerous algorithms which made it to simulation and offer a concrete frame of realization, will be presented. The skeleton ideas of this approach are:

The **quantum input-to-output function**: (1.4) , where

F is an arbitrary quantum gate operator, are also operators corresponding to the weights and acting on quron’s input states |xi> and m is the number if input qubits.

The model is trained with the quantum equivalent to **learning rule**:  (1.5) , where

|d> is the target state, h is the learning rate and |y(t)> the state of the guron.

Of course this approach in its basis suffers from the usual problem of nonlinearity, and w operators are not unitary, and the total probability of the system is not preserved, but as we shall see in later years these problems overcome.

# 2. Quantum Perceptron

The idea of a quantum Perceptron was first introduced in 2001 by Altaisky, who combined the quantum information processing principles with the fundamental equations of classical Perceptrons.[8] Altaisky formed quantum analogues for the output of a Perceptron neuron, the learning rule as well as the cost function and he also proposed how physical implementation of such a system could be achieved.

Consider the output of a classical perceptron:

 (2.1),

F being the activation function, wj the weights and (x1,x2,…..,xn) the input , a pattern vector.

Then a quantum perceptron is:

 (2.2) ,

Where the input pattern vector is binary encoded and each coordinate is described by a qubit state |xi>, wj are 2X2 operators acting on the orthocanonical basis of the 2 dimensional Hilbert space of the states, and F is an operator that acts like an activation function, which Altaisky was unable to define. He also defined the learning rule and objective function considering the simplified cause where F=I, the identical matrix.

As a quantum equivalent to the classical perceptron’s learning rule:

 (2.3) , h the learning step,

is formed as:

 (2.4)

And the Objective function, corresponding to Mean Square Error, derived from the above equations:

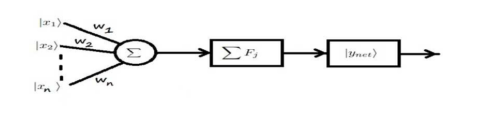
 (2.5)

Altaisky proposed the input and output qubits in this QNN model to be implemented by optical modes with different polarizations, while the weights and F operator by beam splitters and phase shifters.

Unfortunately Altaisky’s idea was incomplete, as besides failing to define the activation operator F, the learning rule he proposed also fails to satisfy unitarity, therefore phase rotations and light attenuation may be observed, disturbing the total probability of the system.

## Autonomous Quantum Perceptron

Inspired by the simplicity of the above approach, in 2013 Sagheer and Zidan[9] described an autonomous Quantum Perceptron NN, autonomous corresponding to the fact that the whole QNN consists of 1 quron only. They also provided several simple examples to illustrate the effectiveness of their algorithm.



Picture 1 The proposed autonomous Quantum Perceptron

The model:

1 quron with n qubit inputs |xi>, corresponding to the binary encoding of a pattern, n wi weight operators applied, one to each input qubit, followed by the Fi unique activation operators and |ynet> the output state of the system.

The algorithm:

**Step 1:** Set Fi=I, the identical matrix, all activation operators, choose randomly the initial weight operators and select a learning step h from [0,1].

**Step 2:** Calculate **the weighted sum** of the input qubits and create a superposition of all of them:

 (2.6) , where

aj, bj, probability amplitudes of the weighted sum for the j-th pattern in the train set. Repeat that for all given patterns.

**Step 3:** Compare the calculated output states |yi> for the patterns of each class with all the other calculated output states for the patterns of other classes.

**Case A:** If each |yi> for any class does not equal the same value for any |yj> in any other class, go to Step 5. Else, go to step 4.

**Case B:** If the value of any |yi> =0, go to step 4. Else go to step 5.

**Step 4:** Update the weight operators, set the iteration counter k=k+1 and return to Step 2.

**Weight update rule:**  (2.7)

**Step 5:** Calculate the **activation operator Fj** for each weighted sum and return to step 2.

Once |yi> is calculated for all available patterns, Fi is defines as:

 (2.8) , where

i=1,….,m , m the number of unique activation operators (repeated operators are discarded), and θ, φ are real valued angles, calculated by:

 (2.9) , where

[ai  bi]T is the weighted sum |yi> qubit and [adi bdi]T the target qubit.

It is obvious now that F operators are applied to map the weighted sum qubit to the target qubit -first row of F- and make it normalized (the sum of the square of amplitudes equal to 1) –second row of F-.

\*By construction the unique Fi operators, and therefore θi, φi , we actually collect information about the problem in hand.

**Step 6:** Recalculate the output state as a sum of the weighted inputs again, after the application of the activation operators defined on previous step.

 (2.10)

**Step 7:** One qubit among all|yi> will be the response state of the Quantum Perceptron , base on the response function:

 (2.11), where

C=[1,1…1]T, D is the vector of the target qubits, L a function that retains the smallest value and makes it 1 and nullify the remain values and  the Hadamard product.

Remarks:

Although the authors underlined the computational power of their model as well as the fact that sometimes the model does not even require the whole train set in order to be trained for a 100% accurate classification (illustrated when the model was used for the XOR problem), and besides the great advantage compared to classical Perceptron of being applicable on non-linear problems (once again illustrated by the XOR example), there remain to major problems in this proposal; the activation operator depends on the angle values, which depend on the weight operators. There is no confirmation provided that either F or W operators are unitary, jeopardizing the violation of basic quantum principles. Also, the iterative nature of training, if overcome, would lead to even greater computational efficiency. In 2015 the above problems exceeded, by the following modification of the Autonomous Quantum Perceptron we just presented.

## Quantum Perceptron with Unitary Weights

Realizing both the power and the gaps in Sagheer and Zidan’s model, Seow, Behrman and Steck[10] worked together to modify it and introduced an alternative Autonomous Quantum Perceptron with unitary weight operators and no iterational learning, utilizing the Singular Value Decomposition (SVD) theorem, a generalization of the Eigendecomposition, and The Moore- Penrose pseudoinverse. They vision is that such a model can become the “building block”[[1]](#footnote-2) for a more complex QNN. Due to the vast similarity to the previous algorithm, a brief summary of this alternated algorithm will be presented and the result will be demonstrated through an example.

The algorithm:

**Step 1:** Set F=I in (2.6), fix the learning rate and define the target states |yi>.

**Step 2:** Calculate the **Moore-Penrose pseudoinverse** of each input qubit |xi>, that is:

(2.12)

**Step 3:** Calculate the **weight operators** as the tensor product of the pseudoinverse of each input and the corresponding target:

(2.13)

**Step 4:** Sum the weight operators of all inputs, with respect to their indexing, to create operator W.

**Case A:** If W is not unitary, move to step 5.

**Case B:** Else move to step 7.

**Step 5:** Use SVD to decompose W into three unitary matrices (see appendix), so that

, U and V unitary and Σ a diagonal matrix with the singular values of W.

**Step 6:** Replace Σ with Σnew by replacing with 1 the singular values of the diagonal and leave everything else 0.

**Step 7:** Introduce a **measurement function**:

 (2.14),

and calculate model’s output |yout> as:

 (2.15)

## Example: The XOR function

Note that XOR problem is not linear, thereafter is incomputable by a simple perceptron.

**Class 1:** P1={|x1>=|0>, |x2>=|0>, |d1>=|0>} or |x1>=|00>

P2={|x1>=|1>, |x2>=|1>, |d2>=|0>} or |x2>=|11>

**Class 2:** P3={|x1>=|0>, |x2>=|1>, |d3>=|1>} or |x3>=|01>

P4={|x1>=|1>, |x2>=|1>, |d2>=|1>} or |x4>=|10>

1st Model

Assume a random initial weight operator 

Using (2.6) to calculate the weighted sums we obtain: 

As |y3>=|y4> and both these weighted sums belong to the same class, only three activation operators F will be formed.

We compute the real valued angles φ and θ for all 3 different |y> and then we create:



Using (2.10) the supersposition output is calculated. And no weight update is needed as the Quantum Perceptron is trained in just 1 iteration, and without using all of the training patterns!

Model 2

In the first step we calculate the pseudoinverse for each of the 4 input patterns and obtain:



and using (2.13) we calculate the weight operators. Summing them all together we have:



Since W is obviously non unitary, we use SVD and find:

. Using (2.14) the proper output is returned is a closed form (that means 1 corresponds to |1> and 0 to |0>)

# 3. Quantum Feedforward Neural Network

## Feed forward QNN using ancilla qubits

While autonomous perceptrons started to formulate back in 2013, the last couple of years the first ideas about a complete feed-forward QNN started to form. Cao, Guerreschi and Aspuru [11] tried to introduce a new thinking of propagating in quantum neural networks by adding ancilla qubits as hidden layer. Each ancilla qubit will be like an encoder of information of the input layer to pass it to the output layer. In order to perform that they propose of a qubit rotation based on Pauli gates (X, Y, Z)

. (3.1)

For the activation method they use:

with  and , (3.2)

where is the number of ancilla qubits being used.

Following the implementation of Control Phase (3.2) in ancilla qubit, we channel the information stored in the ancilla qubit via a Control Phase Y to the output quron. Due to quantum randomness the user must be sure for the right output in every qubit rotation. To overcome this obstacle Repeat Until Success (RUS) method is introduced.

In this method ancilla qubit’s state is measured and a decision is made accordingly. If it is the rotation to output qubit was successfully applied, whether it is, then we have applied a rotation, so we correct it by applying the rotation and repeat the process until we measure

Picture 2 RUS process

For a more generalized case for input and k ancilla qubits the whole process method is described by the following the picture:

**, scales the weighted input to [0, π/2].

In a neural network the goal is to minimize the error of the output layer according to the known attributes of the dataset , .



In the first iteration we randomly choose the weights. Using backpropagation (based on Nelder-Mead algorithm) the weights are updated based on the cost function, and in the end of training the training accuracy is calculated, as defined below.

(3.3)

where, the states of the qubits in dataset and the states of the output layer, and m the length of dataset and output layer. the best accuracy.

Another way to find accuracy is by



## Feedforward QNN with Layer Transition Matrix

Chen and Wang[12] have tried to combine the supervised learning theory and quantum mechanics reaching in a really nice outcome. They introduced the name of quron for quantum neuron and the quantum state of it being

(3.5)

With these states we can compute the probabilities of each state as

and

respectively, with .

By this way we can simulate the non-linearity of a neural network keeping the linearity of the quantum system with a sigmoid activation function.

For inputs in the state of , their superposition state will be

(3.6)

Then the activation function of will be

(3.7)

With and the binary expansion vector of number .



In order to reproduce the probability of a quron to be in state according to the state of previous layer state , they propose the **Layer Transition Matrix** (LTM), resulting to with dimensions with the number of qubits in layer . So the probability of quron in layer 2 to be is with and (3.8)

To generalize it for output layer being in state in binary expansion vector of state of training sample

*(3.9)*

Backpropagation

When we try to backpropagate in order to alter the weights of the neural network we take the partial derivatives of the loss function to every weight. Our loss function of the output layer of the sample will be

(3.10)

Where and

(3.11)

For the hidden layer with weights

(3.12)

Generalizing for layers and the quron of the layer

(3.13)

With representing the LTM from the input layer to layer .

For classical simulation, due to computational cost they propose to sample the qurons of the hidden layer according to the contribution to the final output, only sampling states are needed.

A special case for QPNN is by adding one control layer, only qubits are needed to use them as QRAM that will store different quantum gates defined as output transition matrix, OTG, (the LTM of the output layer) .Based on this analysis they computed the conditional probability of entire QPNN as , where represents the th network, is the probability of the control layer in the state , is the output matrix of the th network, and is the OTG of the th network.



The main added value of the QPNN which they proposed is that we are able to retrieve the samples from output matrix and control layer, not only from the dataset, but every possible sample that fits the desired output and the layout of the NN. By simulating it in MATLAB they reached 2.38% test error accuracy that was only comparable with really complex FFNN that have really big computational complexity

# 4. Quantum Associative Memory Models

Among all approaches of a QNN, this one exploits the theories of quantum world with the most imaginative way. Dived by a Hopfiled Neural Network, with its special associative memory, the attractors and Network’s Energy Equation (see appendix), and under the umbrella of Everett’s many-world interpretation millennium was launched by two individual proposals of a QNN with respect to an HNN, in fact as a network of HNNs as shall discussed.

The basic idea, as Ventrura and Ezhov[7]conceived it, is instead of having a HNN trained to recognize P different classes to have P such networks, one trained to recognize one and only one class, and all of them together forming a quantum system, in superposition of all these distinct HNN’s! Narayanan and Menneer [13] discussed several architectures of such a system, using several MLPs as the components of a QNN with associative memory. Now that the main idea is formed, let’s dive deeper!

The main problem of a single HNN used to store several patterns, is the interference that often happens among those patterns in its memory, when summing their correlations following Hebb’s rule:

 ,sip the i-th neuron’s state on p-th pattern

The second major problem is the limited memory capacity of such a NN. Well, both these problems can be overcome with a QNN that stores all patterns in a superposition, so that each pattern can be considered to exist in a different universe! This superpositional state is what offers to the QNN the parallel processing that distinct a classical NN, as the interaction of |ψ> with the environment is performed in parallel. Also this way an exponential memory capacity is gained, entanglement guarantees QNN’s stability and the problem of corrupted memories is elegantly overcome. So how this QNN will work?

Memorization:

By memorization we mean the process of storing the P train patterns to the memory of the QNN. The goal is the creation of a coherent state of m basis states representing m patterns as proposed by Ventura and Martinez in their previous work [6]. In case of m patterns, whose binary encoding is of length n (therefore use n qubits for their representation in the system), 2n+1 qubits are required for storage and a polynomial number of m\*n elementary operators (applied on 1,2 or 3 qubits) is needed. The 2n+1 qubits are utilized as follows; the first n are used to store the patterns, therefore to form the component network of the QNN’s associative memory and the remaining n+1 qubits are ancillary, used for “bookkeeping” during the storage iteration and restored to |0> after the termination of each iteration.

The key operator during this process is Sp, an operator that forms a set of conditional Hadamard-like transforms that will help to set the QNN in a coherent state. Consider the case of 2 qubits required to encode a pattern, then:

 , p an index {1,2,…,m} indicating the pattern. Thereafter, there is a different operator associated to each pattern we want to store. A different S operator is applied to the input state of the system (a basis state of n qubits that represents the pattern) in every iteration, adding one pattern at a time to the coherent state of QNN.

Along to this operator the full algorithm, developed to initialize the amplitude distribution of a quantum state and shall not be fully presented here, because it exceeds this essays purpose, combines the S operator to CNOT and CSWAP (Fredkin) gates to obtain the result. The result of this algorithm, after O(mn) steps, is a coherent superposition of states corresponding to the m basis states, the m patterns’ states, with equal possibility amplitudes for each state.

Recall-completion:

In order to recall the state of the pattern that corresponds better (has the closest resemblance) to a new input pattern the QNN is required to classify, Grover’s quantum search algorithm is used. In the quantum computational sense, recalling a state means measure the system and cause it to collapse in that basis state that corresponds to the pattern we are searching for.

The basic idea is to invert the phase of the sought out basis state and then invert all the basis states about the average amplitude of all the states, repeating  (binary encoding of length n for each pattern) times. These repetitions will increase the amplitude of our desired basis state to near 1, while decreasing the remaining states’ amplitudes accordingly. An outline of the procedure is:

1) Prepare the system in |0n> state.

2) Apply Hadamard gate  to each input quibit, to initialize all possible states to the same amplitude.

3) Apply , an identity matrix except for idd=-1, d the desired state, to inverse its phase.

4) Finally, apply , to inverse about average.

5) Repeat  times and observe the system.

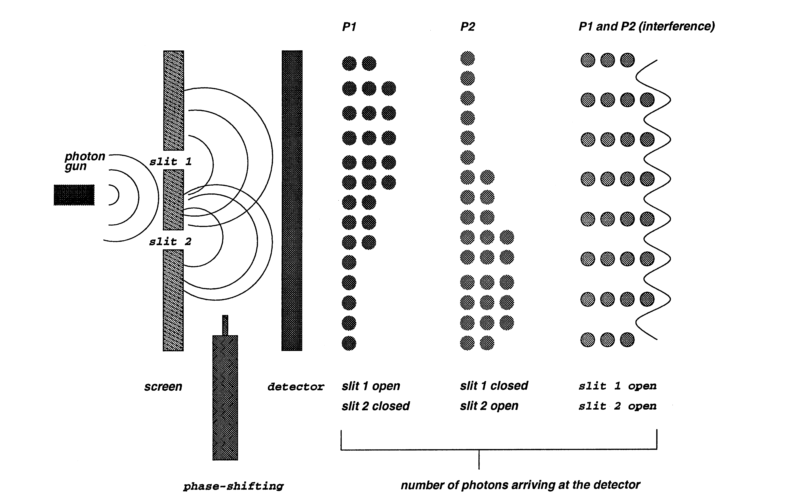
To implement the quantum associative memory, the above two algorithms will be combined.

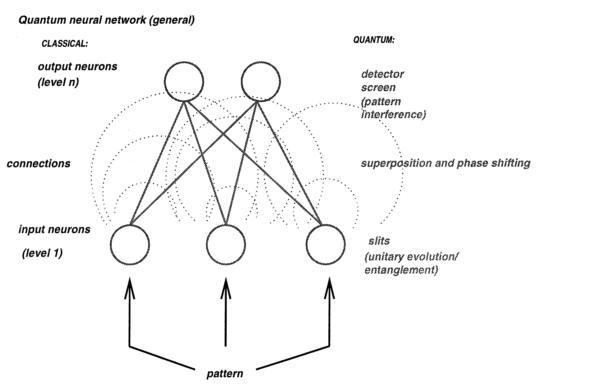
Although no example or a concrete step-by-step algorithm is offered by the authors, not even a graph to explain how they imagined such a QNN’ s structure, they did propose several physical systems that could be used for an implementation; Quantum Dots, Nuclear Magnetic Resonance, ion traps and cavity QED. Since our level of understanding in Quantum Theory required a less abstract presentation though, we resorted to Menneer and Narayanan’s proposal, inspired from Young’s double slit experiment (see appendix) and what it implies if translated in a hardware level. They also were the first that managed to simulate their proposals providing interesting and encouraging results (yet once again not a lot of information about the simulations shared).

The model:

**Main idea:** A QNN modeled as a modern version of “double slit” experiment where the “photon gun” is the input pattern and the slits the input neurons, in a form of entanglement. The waves between the slits created by the input patterns correspond to the connections between input and next layer’s neurons, in superposition, while the output acts like a detector screen and therefore the superposition collapses when arrived in the output layer and a result is given. In our case the output is the state obtained after a measurement. The weight changes can be applied by a phase-shifting operator, such as one of those proposed by Grover.

This can be better explained in the following pictures:



Picture 3 The modern version of double-slit experiment

Picture 4 The corresponding QNN, modeled with respect to the double-slit experiment

The generic architecture can be realized for any number of levels, but without loss of generality we will proceed considering a two layer network (input-hidden). For 2 layers we have as much as 4 quantum components, namely:

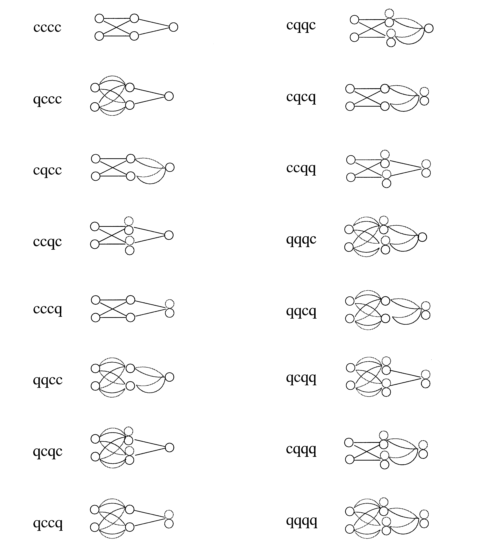
1) input-to-hidden-layer (IH)

2) hidden-bias weights (HB)- the hidden neurons

3) hidden-to-output layer (HO)

4) output-bias weights (OB)-the output neurons

Thereby, combining these possible quantum components with classical NN components one can obtain 24=16 possible architectures of QNN, ranging from completely quantum to completely classical. Adopting the notations [IH, HB, HO, OB] to describe the architecture of a 2 layer QNN of such type, where “c” represents a classical component and “q” a quantum one, we obtain the following 16 architectures:



Picture 5 16 different architectures of a QNN

Training and testing:

1) Given m patterns for training, set m homogenous components and channel each pattern to each one of them. (Imagine each component as a classical MLP)

2) The weights of each component will be updated to learn the pattern that was channeled to it, and only this pattern.[[2]](#footnote-3)

During classical simulation, the weight update happens with classical back propagation, caused by bit-by-bit training through the input pattern. Therefore there are as many links (remember weights are representing links) in every component as the bits of the input pattern, each link representing one bit of the pattern, and each link be entangled with this bit.

3) Train the components either as a superposition of networks or as a set of classical networks which will be set in superposition after their training. The weights from the same component are entangled with each other. (And each weight is entangled with one of the input’s bits)

4) The links of the QNN (links between neuron components of one layer to these of the other) are constructed based on the weights of each component as follows; every link is a superposition of the weights taken from a different component or a different network of components

5) When a test input is presented, the QNN collapses to a set of weights that correspond to the component trained to recognize the pattern closest (by hamming distance) to the test input.

Since the weights of each component are entangled, when one of QNN’s link’s superposition collapses to the state of a weight from this given component, all the other link superpositions in the QNN will collapse in the same component too.

Collapse:

Given a test input, the QNN will search for match by examining the test bit-by-bit with all the training inputs stored in its components.

1) The weights of each component are assigned coefficients, corresponding to their amplitude. The coefficient for a given weight is: ,

c is the component, j the neuron to which the weight is connected, i the neuron the weight comes from, N(i) the activation of neuron I and T(c,i) the training activation for neuron I in component c.

That practically means that if a bit’s match is found, the phase of the appropriate weighted is rotated to make it stand out, when the phases of the remaining weights remains the same.

2) After all test input’s bits are one-by-one tested, the weight coefficients are summed for each component and the measurement causes the system to collapse to the component with the greatest sum. Mathematically speaking:



# 5. Quantum Learning[14]

**Quantum Machine Learning** (QML) is defined as the implementation of quantum software, that is quantum algorithms, to enable the classical machine learning algorithms to perform faster and more accurate, exploiting the advantages of Quantum Computing. This potential of quantum software is called **quantum speedup**. By **quantum algorithm** we mean the stepwise procedure performed on a quantum computer to solve a problem. Since the hardware to create a Quantum Computer and test the efficiency of the QML algorithms compared to classical ones is yet to come, **Quantum Simulators**, the “quantum analogue computers” is the most powerful quantum data analysis technique obtained in our days. Quantum simulators are basically quantum systems whose dynamics can be programmed to match the dynamics of a desired quantum system.

Since the majority of QML algorithms proposed are still far from be implemented, due to hardware issues, cost and complexity, their performance and quantum speedup compared to classical algorithms is currently measured by idealized measures for complexity, namely **query and gate complexity**. Query complexity, following the traditional notion, measures how many queries to the information source are needed for the implementation of the algorithm. Gate complexity has no classical analogue, and refers to the number of elementary gates needed to obtain the desired result.

So far, the basic ML algorithms and models that found a quantum analogue are:

1) **Linear-algrebra problems:** Harrow, Hassidim and Loyd algorithm, developed to solve Ax=b equation using a quantum computer. This algorithm consist a tool for almost everything else on this list.

2) **Quantum Support Vector Machines and Quantum Radial Basis Functions**, seeking to find the optimal hyperplane between two classes of data to ensure the highest probability of classifying these two classes in the one and the other side of this hyperplane. Several approach made it to experimental demonstration in a nuclear magnetic resonance for the classification of NMIST digits.

3) **QPCA** the quantum analogue of PCA, the algorithm that was developed offers a quantum speedup as it requires O[( logN)2] computational and query complexity, compared to O(N2) for the classical PCA, N data’s dimension.

4) **Deep learning by training quantum Boltzmann Machines**, with the ability to create a quantum associative memory and generate quantum states, currently in a design state.

5) **Optimization via qBLAS- based optimization** and the Quantum Approximate Optimization (QAO) algorithm, promising a quantum speedup of O(logN) complexity compared to classical O(N), but only for spare matrices at this point.

The issues arising when going deeper to the realization of these algorithms are numerous; whether they are implemented in classical or quantum data, how to develop and control those quantum systems( Heuristic Search, Genetic algorithms and Stochastic Gradient Descent being proposed) and the biggest question, how to overcome the 4 major problems that the implementation of these algorithms :

1)**The input problem,** referring to the difficulties presenting when quantum algorithms must read classical data, as the cost of reading the data may outperform the quantum speedup of the QML algorithm.

2) **The output problem**; as the quantum system will provide us with a quantum state as an answer, the decoding of this quantum state to bits, and the encoding of our desired states to binary codes at the same time, require the quantum system to learn an exponential number of bits.

3) **The costing problem;** little are still known about how many gates will be needed to perform a QML algorithm, a number that may arise to 1025 as estimated for one proposed realization of HHL. Moreover the construction of interface devices such as qRAM that allow classical information to be encoded in quantum mechanic form is a great challenge for the development of QML, as the cost of a qRAM is to our days feasible.

4) **The benchmarking problem** , as it is often extremely difficult to prove that a QML algorithm will eventually outperform the classical one, at least until realization is an option.

To conclude, half of the above problems can be avoided with the usage of quantum data instead of classical ones, and the development of small quantum computers and larger special- purpose quantum simulators, like annealers, seem to have potential use in QML and data analysis, yet the required quantum hardware is not available in our days.

# 6. Conclusion

Advantages of QNN expected : [7], [10]

1) Exponential Memory capacity

2) Higher performance with lower complexity (less number of neurons, less required training data)

3) Faster learning (due to superposition)

4) Elimination of spurious or erased memory duo the absence of pattern interference

5) Solution to linear inseparable problems with only one layer QNN

6) Absence of wiring (thanks to entanglement property)

7) Higher stability and reliability

8) Processing speed (1010bits/s)

9) Small scale (1011 neurons/mm3)

# Appendix

## Singular Value Decomposition of a non- square matrix

Let A be a mXn matrix.

1) Calculate the eigenvalues and corresponding eigenvectors of W= AAT. (They called *left eigenvectors and eigenvalues* of A)

The eigenvectors of W are the columns of the mXm matrix U, and U is unitary.

Place them in descending order according to their corresponding eigenvalues.

2) Calculate the eigenvalues and corresponding eigenvectors of M=ATA. (They called *right eigenvectors and eigenvalues* of A)

The eigenvectors of M are the rows of the nXn matrix VT, and V is unitary.

Place them in descending order according to their corresponding eigenvalues.

3) The singualar values that fill the diagonal of the mXn matrix Σ are the square roots of either the left or the right eigenvalues of A, in descending order.

All other elements of A are 0.

4) Then, A=UΣVT

## Hopfield Neural Network with Associative Memory

A Hopfield Neural Network (HNN) introduced by John Hopfield in 1982 and is a Recurrent NN of McCulloch-Pitts neurons (their output is -1 or 1) with the following properties: [15]

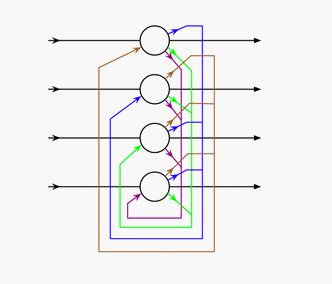
1) It consists of only one layer of neurons and is a feedforward NN, therefore its outputs are redirected to their inputs.

2) Every neuron acts as an input and output, therefore the input and output neurons are of equal number.

3) wij=wji (symmetric synapses)

4) wii=0 (no self-loops)

5) Each neuron’s state converges to a fixed state after a certain number of updates.



Picture 6A Hopfield NN

The most powerful property of a HNN is the Associative Memory, the ability to retrieve the network state ,out of P stored network states XP={x1(p),….,xN(p)} , that is closest to the input pattern in terms of Hamming distance. Its memory capacity is estimated to be C=N/4logN, N the number of different patterns.

Basic Equations of a HNN:

**Weighted input sum** of i neuron: , sj is the state of neuron j

**New state** of a neuron: , θ a threshold

**Hebbs learning rule** for weight update: , sip is the state of I neuron on p-th pattern

\*Neurons that have the same state in the majority of memory patterns will receive a synaptic weight close to 1 while neurons that have high antiparallel correlations in the memory will receive weights close to -1.

**Energy of i neuron:** 

**HNN Energy Function:[[3]](#footnote-4)** 

Network’s energy indicates its level of modification, reaching its minimum when it is stable after an update. This is because each neuron’s energy decreases when it changes sign.

The training patterns represent the attractors of the network, corresponding to network energy’s minima. This causes the training process to stop once a pattern is reached.

This behavior of an HNN may cause a malfunction, in case a local minimum is reached and misunderstood as a pattern, when it’s not. These local minima (called *corrupted or spurious memories*) are interpreted as false memories.

## Quantum World Interpretations[7]

Copenhagen Interpretation

1) Non-unitary operators do exist (per example observation)

2) Non-evolutionary behavior of the system is as important as the evolutionary one

3) Pattern recall is a non-unitary process, as observation or measurement cause the superposition to collapse

4) Decoherence in the quantum system is the analogue of evolution of a HNN’s state to an attractor’s state

Feynman’s Interpretation

1) Use of path integrals to express the probabilities on a system’s state

2) Non- linear probabilities

Everett’s many worlds interpretation

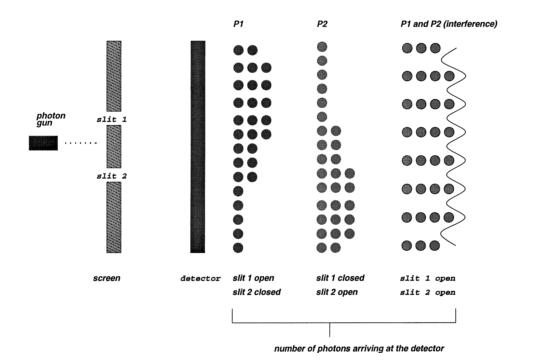
1) Decoherence and Collapse of the quantum system are “illusions”

2) Wave function obeys Schrödinger’s equation all times

3) Measurement’s result is not the collapse of the wave function, but the split of the observer into a number of copies equal to the possible states, each copy observing one and only one of the possible results, unaware of the other possibilities

4) There exist many, mutually unobservable but equally real universes.

## Modern version of double-slit experiment[13]



Picture 7 A modern version of Young's double split experiment

A “photon gun” fires photons individually through a “wall” or screen with two splits, onto a detector.

There are two groups of individual photons corresponding to patterns P1 and P2, whose detection when one or the other split is closed follows a probability distribution expected of discrete particles.

When both slits are open, the pattern obtained follows a typical interference pattern obtained if the photon travelled as a wave from the two slits, and is not the sum of P1 and P2.

Interestingly enough, when both slits are open individual photons do not appear in areas where previously they appeared, and individual photons arrive in number in areas where previously there was little chance of a photon appearing.

This bizarre arrival of photons in places they were not supposed to be indicates that the photons have a double nature, acting like particles when fired from the “photon gun” and then transform to a wave when they travel, becoming particles again once they collapse to the detector.

The “bizarre” double nature actually applies to any particle at the atomic and sub-atomic level, including electrons.

# Bibliography

[1] S. C. Kak, “Quantum Neural Computing,” *Adv. Imaging Electron Phys.*, vol. 94, no. C, pp. 259–313, 1995.

[2] M. Schuld, I. Sinayskiy, and F. Petruccione, “The quest for a Quantum Neural Network,” *Quantum Inf. Process.*, vol. 13, no. 11, pp. 2567–2586, 2014.

[3] M. Perus, “Neural networks as a basis for quantum associative networks,” *Neural Netw. World*, vol. 10, no. 6, pp. 1001–1013, 2000.

[4] E. C. Behrman, L. R. Nash, J. E. Steck, V. G. Chandrashekar, and S. R. Skinner, “Simulations of quantum neural networks,” *Inf. Sci. (Ny).*, vol. 128, no. 3, pp. 257–269, 2000.

[5] S. Gupta and R. K. P. Zia, “Quantum Neural Networks,” *J. Comput. Syst. Sci.*, vol. 63, no. 3, pp. 355–383, Nov. 2001.

[6] D. Ventura and T. Martinez, “Initializing the amplitude distribution of a quantum state,” *Found. Phys. Lett.*, vol. 12, no. 6, pp. 547–559, 1999.

[7] A. A. Ezhov and D. Ventura, “. Quantum Neural Networks,” *Futur. Dir. Intell. Syst. Inf. Sci. Futur. Speech Image Technol. Brain Comput. WWW, Bioinforma.*, p. 213, 2000.

[8] M. V. Altaisky, “Quantum neural network,” no. 1, pp. 1–4, 2001.

[9] A. Sagheer and M. Zidan, “Autonomous Quantum Perceptron Neural Network,” no. December, 2013.

[10] K.-L. Seow, E. Behrman, and J. Steck, “Efficient learning algorithm for quantum perceptron unitary weights,” pp. 1–10, 2015.

[11] Y. Cao, G. G. Guerreschi, and A. Aspuru-Guzik, “Quantum Neuron: an elementary building block for machine learning on quantum computers,” pp. 1–30, 2017.

[12] J. Chen, L. Wang, and E. Charbon, “A quantum-implementable neural network model,” *Quantum Inf. Process.*, vol. 16, no. 10, pp. 1–24, 2017.

[13] A. Narayanan and T. Menneer, “Quantum artificial neural network architectures and components,” *Inf. Sci. (Ny).*, vol. 128, no. 3, pp. 231–255, 2000.

[14] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, “Quantum machine learning,” *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.

[15] S. S. Haykin, *Neural networks and learning machines, 3rd Edition*. 2009.

1. The idea of a quantum neuron that consist a small quantum circuit , as a “building block” for creating a feedforward QNN was fully completed by Cao et all[11], who introduced a QNN based on Hopfield NN and a Hebbian learning backpropagation rule for its training. They provided a solution to the problem of nonlinearity by utilizing such “neuron-circuits” [↑](#footnote-ref-2)
2. Each pattern has its own component therefore no interfere between patterns will occur and no corrupted memories will be created [↑](#footnote-ref-3)
3. This is equivalent to the energy function of an Ising spin-glass model [↑](#footnote-ref-4)