Chapter

Quantum Neural Machine Learning: Theory and Experiments

Carlos Pedro dos Santos Gonçalves

Abstract

Cloud-based access to quantum computers opens up the way for the empirical implementation of quantum artificial neural networks and for the future integration of quantum computation in different devices, using the cloud to access a quantum computer. The current work experimentally implements quantum artificial neural networks on IBM's quantum computers, accessed via cloud. Examples are provided for the XOR Boolean function representation problem and decision under risk; in the last case, quantum object-oriented programming using IBM's Qiskit Python library is employed to implement a form of quantum neural reinforcement learning applied to a classical decision under risk problem, showing how decision can be integrated into a quantum artificial intelligence system, where an artificial agent learns how to select an optimal action when facing a classical gamble. A final reflection is provided on quantum robotics and a future where robotic systems are connected to quantum computers via cloud, using quantum neural computation to learn to optimize tasks and act accordingly.

Keywords: quantum artificial neural networks, quantum neural reinforcement learning, quantum object-oriented programming, decision under risk

1. Introduction

Research on quantum neural machine learning has, until recently, mostly been a theoretical effort, anticipating a future where quantum computers would become available and sufficiently advanced to support quantum neural machine learning [1–5]. However, we now have quantum computers that are capable of implementing quantum artificial neural networks (QUANNs) experimentally, and one is able to access these computers via cloud. This brings QUANNs from the purely theoretical realm to the experimental realm, setting up the new stage for the expansion of quantum connectionism. In the current chapter, we address this issue, by implementing different QUANNs on IBM's quantum computers using the IBM Q Experience cloud-based access.

The chapter is divided into three sections. In Section 2, we address the basic properties of quantum neural computation, the connection with the quantum circuit computation model, and how different interpretations of quantum mechanics may address the basic computational dynamics involved.

In Section 3, we discuss how the IBM quantum computers can be considered QUANNs, illustrating with an example of a QUANN applied to the problem of the XOR Boolean function computation, implemented experimentally on two of IBM's devices (Section 3.1); afterward (Section 3.2), we turn to the experimental implementation of quantum robotics and quantum decision with a more complex form of quantum neural computation in the form of a variant of quantum neural reinforcement learning (QNRL), applied to a problem of decision under risk, where the agent must learn the optimal action that leads to the highest expected reward in a classical gamble.

The problem is first addressed in terms of the fundamental equations which employ quantum adaptive computation, namely quantum adaptive gates; then, we implement it experimentally on IBM's quantum computers and, afterward, we address the main Python code that was used to run the algorithm on these computers, thus, introducing quantum object-oriented programming (QOOP) and reflecting on its relevance for research on quantum artificial intelligence.

While, in Section 3.1, the main goal is to illustrate the implementation of QUANNs in a case where QUANNs exhibit a greater efficiency over classical ANNs, in Section 3.2, our main goal is not to address the speed-up of quantum algorithms over classical ones or even the greater efficiency of quantum algorithms over classical ones, but rather to provide for a reflection on the first steps for a possible future where quantum computation is incorporated in different (classical) robotic systems by way of the internet of things and cloud-based access to quantum devices, and the role that quantum adaptive computation may play in such a future.

In particular, in Section 3.2, we illustrate how a QUANN can become adaptive with respect to a problem that is given to it, in this case, a decision problem under risk, therefore, allowing us to address how QOOP can be employed to simulate an artificial agent, with a QUANN as its cognitive architecture, that must make a decision when presented a problem of classical decision under risk; therefore, our main goal in Section 3.2, from a computer science standpoint, is to address how a quantum artificially intelligent system decides when faced with a classical decision under risk problem, using QUANNs and QOOP.

In Section 4, we conclude with a chapter review and a reflection on future directions for cloud-based quantum-enabled technologies and QOOP.

2. Quantum neural computation and quantum mechanics

In order to address quantum neural computation, we need to first introduce some notation, which is commonly used in quantum computation, namely, we use the standard Dirac's *bra-ket* notation, where a *ket* vector corresponds to a column vector and the *bra* vector is its conjugate transpose. Defining, then, the fundamental *ket* vectors $|0\rangle$ and $|1\rangle$, respectively, as:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad (1)$$

with the corresponding *bra* vectors $\langle 0|$ and $\langle 1|$ being defined, respectively, as the conjugate transpose of $|0\rangle$ and $|1\rangle$, then, we can represent Pauli's operators as:

$$\hat{\sigma}_1 = |0\rangle\langle 1| + |1\rangle\langle 0| = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad (2)$$

$$\hat{\sigma}_2 = -i|0\rangle\langle 1| + i|1\rangle\langle 0| = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad (3)$$

$$\hat{\sigma}_3 = |0\rangle\langle 0| - |1\rangle\langle 1| = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad (4)$$

The unit operator on the two-dimensional Hilbert space, spanned by the basis $\{|0\rangle, |1\rangle\}$, is denoted by $\hat{1} = |0\rangle\langle 0| + |1\rangle\langle 1|$ which has the form of the identity matrix.

The Walsh-Hadamard transform unitary operator is, in turn, given by:

$$\hat{U}_{WH} = \frac{\hat{\sigma}_1 + \hat{\sigma}_3}{\sqrt{2}} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{5}$$

We also use the usual notation for the ket vectors $|+\rangle = \hat{U}_{WH}|0\rangle$ and $|-\rangle = \hat{U}_{WH}|1\rangle$.

Besides the above notation, we denote the binary alphabet by $\mathrm{A}_2 = \{0, 1\}$ and the set of d-length binary strings by A_2^d, using boldface letters to represent binary strings of length greater than 1.

Using this notation, we are now ready to address some basic general properties of quantum neural computation.

The basic computational unit of a QUANN is a neuron with a two-level firing dynamics that can be described by the neural firing operator [5, 6]:

$$\hat{\nu} = \frac{\hat{1} - \hat{\sigma}_3}{2}\nu \tag{6}$$

where ν is a neural firing frequency expressed in Hertz.

The eigenvectors for this operator are given by:

$$\hat{\nu}|s\rangle = s\nu|s\rangle, s = 0, 1 \tag{7}$$

Therefore, the eigenvector $|0\rangle$ corresponds to a neural activity where the firing frequency is 0 Hz, while the eigenvector $|1\rangle$ corresponds to a neural activity where the firing frequency is 1 Hz. This means that there are two quantized energy levels associated with the artificial neuron, and these energy levels are obtained from the single neuron Hamiltonian, expressed in terms of the neural firing frequency operator as follows [5, 6]:

$$\hat{H} = 2\pi\eta\hat{\nu} \tag{8}$$

Therefore, the eigenvector $|0\rangle$ is associated with a neural firing energy level of 0 Joules, while the eigenvector $|1\rangle$ is associated with a neural firing energy level of $2\pi\eta\nu$ Joules.

For a neural network with d neurons, the neural firing activity can be addressed in terms of a neural field in the network, with the firing frequency field operators such that the k-th neuron neural firing operator $\hat{\nu}_k$ obeys the eigenvalue equation:

$$\hat{\nu}_k|s_1 s_2 \ldots s_d\rangle = s_k\nu|s_1 s_2 \ldots s_d\rangle \tag{9}$$

and any pair of neural firing operators commute; that is, for $k, l = 1, 2, \ldots, d$, $[\hat{\nu}_k, \hat{\nu}_l] = 0$. Thus, the total neural firing frequency operator is given by:

$$\hat{\nu}_{Tot} = \sum_{k=1}^{d} \hat{\nu}_k \tag{10}$$

which leads to the eigenvalue spectrum for the neural network:

$$\hat{\nu}_{Tot}|s_1 s_2 \ldots s_d\rangle = \sum_{k=1}^{d} s_k\nu|s_1 s_2 \ldots s_d\rangle \tag{11}$$

The general computational dynamics of a d-neuron QUANN can be addressed, in the quantum circuit model, by a computational chain of unitary operators, where the networked computation is implemented by conditional unitary operators that follow the structure of the neural links [4–6], which means that not all conditional unitary operators can be implemented in the neural network, but only those that respect the network's topology and processing direction.

Formally, then, an N-length computational chain that propagates forward in a quantum neural computation circuit is comprised of a sequential product of unitary operators:

$$\hat{C} = \hat{U}_N \hat{U}_{N-1} \ldots \hat{U}_1 \tag{12}$$

The sequence is read from right to left and such that $\hat{U}_1$ is the first operator to be applied and $\hat{U}_N$ is the last. This is the forward sequence proceeding from the beginning to the end of the computation. The reverse chain, which propagates backward in the computational circuit, is, then, given by the conjugate transpose of the forward chain:

$$\hat{C}^{\dagger} = \hat{U}_1^{\dagger} \ldots \hat{U}_{N-1}^{\dagger} \hat{U}_N^{\dagger} \tag{13}$$

Formally, given a general initial density operator, representing the initial neural field dynamics of the QUANN, expressed as follows:

$$\hat{\rho}_{in} = \sum_{\mathbf{r},\, \mathbf{s} \in \mathrm{A}_2^d} \rho_{\mathbf{r},\, \mathbf{s}} |\mathbf{r}\rangle\langle\mathbf{s}| \tag{14}$$

the quantum computation can be addressed in terms of the propagation:

$$\begin{aligned}\hat{\rho}_{out} &= \hat{C}\hat{\rho}\hat{C}^{\dagger} = \\ &= \sum_{\mathbf{r}',\, \mathbf{s}' \in \mathrm{A}_2^d} \left(\sum_{\mathbf{r},\, \mathbf{s} \in \mathrm{A}_2^d} \rho_{\mathbf{r},\, \mathbf{s}} \left\langle \mathbf{r}' \middle| \hat{U}_N \hat{U}_{N-1} \ldots \hat{U}_1 \middle| \mathbf{r} \right\rangle \left| \mathbf{r}' \right\rangle \left\langle \mathbf{s}' \right| \left\langle \mathbf{s} \middle| \hat{U}_1^{\dagger} \ldots \hat{U}_{N-1}^{\dagger} \hat{U}_N^{\dagger} \middle| \mathbf{s}' \right\rangle \right)\end{aligned} \tag{15}$$

The firing patterns, in Eq. (15), $\mathbf{r}$ and $\mathbf{s}$ correspond to input neural firing patterns for the QUANN, while the firing patterns $\mathbf{r}'$ and $\mathbf{s}'$ correspond to output neural firing patterns; in this way, the quantum neural computation is propagating in both directions of the computational chain, so that we have the propagation from the input to the output (from the beginning to the end of the computational chain), which corresponds to the amplitude$\langle \mathbf{r}' | \hat{U}_N \hat{U}_{N-1} \ldots \hat{U}_1 | \mathbf{r} \rangle$, and the propagation from the output to the input (from the end to the beginning of the computational chain), which corresponds to the amplitude $\left\langle \mathbf{s} \middle| \hat{U}_1^{\dagger} \ldots \hat{U}_{N-1}^{\dagger} \hat{U}_N^{\dagger} \middle| \mathbf{s}' \right\rangle$.

For the cases where there is a mismatch between the final output firing dynamics, that is, when $\mathbf{r}' \neq \mathbf{s}'$, the QUANN does not reach a well-defined output; from a computational perspective, we can state that the network does not reach a final solution, since the output computed in the forward direction of the computational circuit, $\langle \mathbf{r}' | \hat{U}_N \hat{U}_{N-1} \ldots \hat{U}_1 | \mathbf{r} \rangle$, does not match the output computed in the reverse direction of the computational circuit, $\left\langle \mathbf{s} \middle| \hat{U}_1^{\dagger} \ldots \hat{U}_{N-1}^{\dagger} \hat{U}_N^{\dagger} \middle| \mathbf{s}' \right\rangle$.

However, when $\mathbf{r}' = \mathbf{s}'$, the output computed in the forward and backward directions matches; this leads to the diagonal components of the final density operator that, for each $\mathbf{s}' \in \mathrm{A}_2^d$, are given by:

$$\left\langle \mathbf{s}' | \hat{\rho}_{out} | \mathbf{s}' \right\rangle |\mathbf{s}'\rangle \left\langle \mathbf{s}' \right| = \left(\sum_{\mathbf{r},\, \mathbf{s} \in \mathrm{A}_2^d} \rho_{\mathbf{r},\mathbf{s}} \left\langle \mathbf{s}' \middle| \hat{U}_N \hat{U}_{N-1} \ldots \hat{U}_1 \middle| \mathbf{r} \right\rangle \left\langle \mathbf{s} \middle| \hat{U}_1^{\dagger} \ldots \hat{U}_{N-1}^{\dagger} \hat{U}_N^{\dagger} \middle| \mathbf{s}' \right\rangle \right) |\mathbf{s}'\rangle \left\langle \mathbf{s}' \right| \tag{16}$$

This means that the neural field computes each alternative final firing pattern $\mathbf{s}' \in \mathrm{A}_2^d$ with a projective intensity given by the weighted sum over each pair of alternative initial firing patterns propagated in both directions of the computational chain:

$$\left\langle \mathbf{s}' | \hat{\rho}_{out} | \mathbf{s}' \right\rangle = \sum_{\mathbf{r},\, \mathbf{s} \in \mathrm{A}_2^d} \rho_{\mathbf{r},\mathbf{s}} \left\langle \mathbf{s}' \middle| \hat{U}_N \hat{U}_{N-1} \ldots \hat{U}_1 \middle| \mathbf{r} \right\rangle \left\langle \mathbf{s} \middle| \hat{U}_1^{\dagger} \ldots \hat{U}_{N-1}^{\dagger} \hat{U}_N^{\dagger} \middle| \mathbf{s}' \right\rangle \tag{17}$$

From a computer science standpoint, this two-directional propagation, which is a basic result of the quantum circuit-based computation (generalizable to any type of quantum computer), exhibits a form of forward propagation and backward propagation, where the forward and backward amplitudes can be, from a computer science standpoint, addressed in terms of a probe and response dynamics, respectively; returning to Eq. (15), each amplitude $\left\langle \mathbf{r}' \middle| \hat{U}_N \hat{U}_{N-1} \ldots \hat{U}_1 \middle| \mathbf{r} \right\rangle$ can be addressed as a probing computational dynamics from the beginning to the end of the computational circuit that links the initial (input) firing pattern $\mathbf{r}$ to the final probed (output) firing pattern $\mathbf{r}'$, and the reverse amplitude $\left\langle \mathbf{s} \middle| \hat{U}_1^{\dagger} \ldots \hat{U}_{N-1}^{\dagger} \hat{U}_N^{\dagger} \middle| \mathbf{s}' \right\rangle$ can be addressed as a response that comes from the end of the computational circuit to the beginning, a response that links the output firing pattern $\mathbf{s}'$ to the initial input firing pattern $\mathbf{s}$.

When the two output firing patterns do not match, $\mathbf{r}' \neq \mathbf{s}'$, we have a mismatch between the probe and the response, and when the two firing patterns match, $\mathbf{r}' = \mathbf{s}'$, an *echo* is produced with an intensity given by the sum in Eq. (17); the computation is, then, like the search for the solution to a computational problem, where each probed alternative final output gets a response with a specific intensity.

These dynamics are simultaneous, that is, the QUANN processes in both the forward and backward directions simultaneously to arrive at the final result.

The above fundamental computational dynamics is characteristic of quantum mechanics, and not limited to QUANNs or quantum computation, nor is it dependent on one's interpretation of quantum mechanics. It arises when one considers the structure of a general density operator for a quantum system [6].

Indeed, as an example, let us consider a general density operator for a quantized observable $\hat{O}$ on some quantum system, which, for the purpose of illustration we consider to have a discrete, not necessarily finite, non-degenerate eigenvalue spectrum, so that the *ket* vectors $|m\rangle$, for $m = 0, 1, 2\ldots$, satisfying $\hat{O}|m\rangle = o_m|m\rangle$, span the basis for a Hilbert space associated with the quantum system with respect to the observable; then, the general dynamics for the quantum system, with respect to the observable, can be represented as a density operator on the system's Hilbert space:

$$\hat{\rho} = \sum_{m,\, n} \rho_{m,n} |m\rangle\langle n| \tag{18}$$

The off-diagonal components of such an operator are such that there is no matching between the corresponding eigenvalues, only in the diagonal do we find a matching between the eigenvalues. The *ket* vector can, in this case, be considered as a probing vector, while the *bra* vector can be considered as a response vector.

In this way, only when a probed alternative eigenvalue finds a matching response eigenvalue do we have an *echo* for an alternative eigenvalue that can be actualized, and the probability for this actualization coincides with the diagonal density value $\rho_{m,m}$ which corresponds to the *echo intensity*. This is a basic result from quantum mechanics that extends to any observable, including observables with both discrete as well as continuous spectra.

It is important to stress that this *echo* dynamics is not specific to QUANNs, but is present in any quantum system; any density operator characterizing a quantum system exhibits, in the formalism, this main dynamics, so the *echo* dynamics is a characteristic of the physics of quantum systems and accounts for Born's probability rule in quantum mechanics—that is, the probability of an alternative eigenvalue to be observed is equal to the corresponding diagonal component of a density operator.

Therefore, embedded within quantum mechanics' formalism, we find an account of Born's probability rule. Furthermore, given a Hamiltonian operator for the quantum system $\hat{H}_S$, and a time lapse of Δt, quantum mechanics defines the unitary propagation of a density operator at time t_0 as:

$$\hat{\rho}(t_0 + \Delta t) = e^{-\frac{i}{\eta}\hat{H}_S\Delta t}\hat{\rho}(t_0)e^{\frac{i}{\eta}\hat{H}_S\Delta t} \tag{19}$$

In the case of the illustrative general example, given in Eq. (18), we get:

$$\hat{\rho}(t_0 + \Delta t) = \sum_{m,n}\left(\sum_{k,l}\rho_{k,l}(t_0)\langle m|e^{-\frac{i}{\eta}\hat{H}_S\Delta t}|k\rangle\langle l|e^{\frac{i}{\eta}\hat{H}_S\Delta t}|n\rangle\right)|m\rangle\langle n| \tag{20}$$

where $\langle m|\exp\left(-i/\eta\hat{H}_S\Delta t\right)|k\rangle$ is a forward in time propagating amplitude from the *k-th* initial eigenvalue to the *m-th* final eigenvalue and $\langle l|\exp\left(i/\eta\hat{H}_S\Delta t\right)|n\rangle$ is a backward in time propagating amplitude from the *n-th* final eigenvalue to the *l-th* initial eigenvalue[1], and this basic dynamics is a general result that stems from Schrödinger's unitary evolution.

Cramer was, however, the first to fully address the consequences of this dynamics and propose the concept of *echo*, within the context of quantum mechanics, addressing it related to Born's rule, deriving Born's rule from within the quantum formalism.

While Cramer [7] addresses the *echo* in terms of the encounter of a forward-propagating retarded wave (which we addressed above under the probe dynamics, proceeding forward from the beginning to the endpoint of the unitary evolution) and the backward-propagating advanced wave (which we addressed above under the response dynamics, proceeding from the endpoint to the beginning of the unitary evolution), by working with the density operator, instead of the wave function, we get a clearer picture of the corresponding dynamics, which accounts, in the case of any quantum physical system, for both the off-diagonal terms (as *failed echoes*) and the diagonal terms of the density operator (as the *echoes* where the probe was met by a matching response) with the *echo intensity* giving Born's probability rule. This result is generalizable and independent of the interpretation of quantum mechanics that one follows; that is, all interpretations of quantum mechanics agree with the above results.

It is important to clarify what an interpretation of quantum mechanics is and why there are different interpretations of the same theoretical body and equations. It turns out that the main interpretations do not disagree on the formalism,

[1] One may notice the change in the time lapse signal so that the conjugate transposition corresponds to time reversal.

methods, and how the mathematics is built and applied for prediction of experimental results. The interpretations do not stem from any ambiguity or lack of robustness in the formalism and in the application of the formalism, they stem from the fact that not everything is accounted for by the formalism, and that is where the interpretations come in.

To better frame this issue, one must consider the nature of the theory that one is dealing with, what it explains, and what is outside its theoretical scope.

Quantum mechanics is, in fact, a probabilistic theory of the quantized dynamics of fundamental physical fields, fields that work at the level of the building blocks of physical nature. The physical theory and methods that form the basic structure of quantum mechanics developed progressively from empirical observations and statistical findings on fundamentally random outcomes of physical experiments dealing with the quantum level.

This means that physicists found the basic rules for (dynamical) probability assignments that robustly capture the main probabilistic dynamics of quantum fields.

To understand the nature of the theory, it is important to stress that it was born out of laboratory experiments, that it was built out of the statistical patterns found in an observed stochastic dynamics, and that it was aimed at predicting the statistical distributions of that stochastic dynamics. The current formulation of quantum mechanics essentially encompasses a set of rules for obtaining the probabilities associated with the dynamics of quantum systems.

The theory does not state anything beyond that. A point that allowed many physicists to pragmatically take the theory as it is, not dwelling on the *why quantum systems work that way*, that is, to take the theory as a rule book that works, is robustly tested empirically, applying it to problems following what is usually called a *shut up and calculate stance*.

When one starts to ask on the *why quantum systems work that way*, the interpretations enter into play, but they go beyond the physical setting of the theory in the sense that they are related to ontological questions; that is, each interpretation regards the ontological issue of physical reality and why the quantum dynamics follows the *echoes* with probabilities coincident with the *echo intensities*.

In the pragmatic stance, one just takes the formalism as a recipe, calculates the *echo intensities* without dwelling further on it. Any result in quantum mechanics applying the formalism is valid and empirically testable and the formalism has time and again, during twentieth and twenty-first centuries, been shown to be robust in its predictions.

One way out of the ontological questioning would be to assume that we are dealing with human representations, that we cannot speak of a reality independent of human representations and experiments, that is, that the question of what reality really is outside those representations and experiments cannot be answered and, therefore, one just postulates that the field follows the *echoes*. This was the approach of the Copenhagen school, including Bohr and Born, leading to Born's rule that the probabilities are coincident with the *echoes*, a rule that is introduced, usually, in quantum mechanics' classes as a postulate, a very detailed description of this can be found in [7].

Contrasting with the Copenhagen school are the ontological schools, so called because they assume a reality independent of human representations and experiments.

Quantum mechanics itself does not state anything about this, so there is room for proposals; Cramer [7], for instance, considers these interpretations as actually new physical theories that go beyond the strict formalism and introduce new conjecture that cannot be tested under the formalism itself. The ontological interpretations that include the Bohmian and Everettian lines are all consistent with the formalism, that is, they agree with the formalism and mathematical methods of quantum mechanics and, therefore, cannot be tested using just the formalism.

In the case of Cramer, his proposed transactional interpretation (TI) of quantum mechanics [7] considers a probabilistic selection in terms of a *quantum handshake* (Cramer's transaction), where there is a sequential hierarchical selection for a quantum handshake linking the beginning and endpoint of the quantum dynamics, where each alternative is evaluated probabilistically for the formation of a quantum handshake or not; if no handshake is selected for a given alternative, the quantum dynamics proceeds to the next alternative. In each case, the probability for a quantum handshake is equal to the *echo* intensity, thus deriving Born's rule from within the formalism, instead of assuming it as a postulate.

Everett [8] assumed that all alternatives for a quantum system are actualized simultaneously in different cosmic branches. This led to the many worlds interpretation (MWI). MWI's proposal is, thus, that reality is multidimensional and the formalism is considered to be describing such a multidimensional reality that is a single Cosmos with many worlds (many branching lines). This conjecture cannot be tested empirically; it is consistent with the formalism and agrees with the predictions of quantum mechanics. Namely, the statistical measure associated with repeated experiments made on quantum systems tends to coincide with the *echo* intensities since the *echo intensities* coincide with the existence intensity of each world, recovering a statistical measure upon repeated experiments, as argued by Everett in [8] regarding Born's rule.

Bohm initially worked on the pilot wave model for quantum mechanics but just as a first approximation. Indeed, in [9], the author addressed the pilot wave model as a first approximation but then criticized it, in particular, in regard to the assumption of a particle being separate from the field; even more, in [9], Bohm defended that, at a lower level, the particle does not move as a permanently existing entity, but is formed in a random way by suitable concentrations of the field's energy. Furthermore, he considered that any quantum field was characterized by a nonlocal dynamics, and that the equations of quantum mechanics were just an approximation, an average that emerged at the quantum level, proposing the concept of quantum force and hypothesizing the existence of a subquantum level, so that both the quantum and subquantum levels play a fundamental role in the field's dynamics.

Gonçalves in [6] addressed the relation between the *echo* and Bohm's proposal recovering the Bohmian concept of quantum force [9, 10].

In this interpretation, the *echo* is associated with a dynamics of a quantum field for the evaluation of each alternative; the probing and response dynamics, thus, play a fundamental role, allowing a quantum field, any quantum field, to compute each alternative in parallel, leading to an *echo* associated with each alternative.

As argued in [6], the intensity (modulated) *echoes* would, thus, have a functional role as signalizers of an order to be risen (in the QUANN case, this *order* corresponds to a specific quantum neural firing pattern); the field's quantum and subquantum levels would, then, work in tandem, mobilizing the forces needed to make rise one specific alternative, and the resulting field lines of force, therefore, coincide, in their intensities, with the *echo intensities*.

This quantum computational dynamics, present in quantum mechanics' formalism, works as a basic form of *quantum "learning" dynamics*, where the quantum field "learns" about each alternative in the probe (forward propagating) and response (back propagating) dynamics and, then, the field's lines of force are formed along the *echoes* resulting from the encounters of matching probe and response vectors, with a force intensity that matches the corresponding *echo*'s intensity; the field then follows one of these lines of force with a probability that coincides with the *echo* *intensity*, so that the following of a given line of force is similar to a bifurcation dynamics where the field will follow, stochastically, one of the branches with a probability that coincides with the force intensity associated with each branch [6].

There is a consequence that comes from assuming the Bohmian framework, namely, from the Bohm's conjecture that a subquantum level randomness averages out at the quantum level, but may lead to small deviations from the theoretical probabilities [9, 10]; if such a conjecture holds, then deviations in quantum physical experiments with actual quantum computers may always take place, such that, even if we were to reduce the interaction with the environment to zero (or close to zero), we could still have deviations due to subquantum level fluctuations, so that the field would tend to follow the lines of force with probabilities that would hold on average but with some deviations that might occur in each case.

While Bohm's proposal is potentially testable, at the present stage of scientific and technological development, we have not yet found a way to test the subquantum proposal regarding quantum physical systems, and, in particular, to test, empirically, the possibility that deviations from the main lines of force that agree with a theory's prediction are not due to environmental noise and, rather, to subquantum level fluctuations.

All main interpretations, as reviewed above, agree with quantum mechanics' general predictions, even Bohm, who considers that the predictions will hold empirically on average, therefore, the interpretations do not have, at present, a direct consequence on the results of technological implementation of quantum computers, as long as one is not dealing with fundamental ontological issues regarding the computational nature of quantum fields, but rather with the technological application of quantum algorithms, one is free to choose any interpretation since it is consistent with the main formalism and results.

We consider, nonetheless, that future research directions on Bohm's conjectural line may prove fruitful both at a theoretical and technological level, concerning the issue of quantum errors. This point, however, goes beyond the current chapter's scope. The results that follow, as of any work using the formalism of quantum mechanics, hold for any interpretation of the theory. However, having made that point, we will return to Bohm's conjecture regarding some of the results obtained in the next section, regarding the issue of quantum computing errors.

3. Implementing quantum artificial neural networks on IBM's quantum computers

The development of quantum computing devices has opened up the possibility of transitioning from the purely theoretical approach to QUANNs to an experimental implementation of these networks. A particular example is IBM's quantum processors, available via cloud, under IBM Q Experience, using *superconducting transmon quantum processing units*.

The term *transmon* stands for *transmission-line shunted plasma oscillation*. A *transmon qubit* [11, 12] is an attempt at a technological implementation of a *qubit* for quantum computation, using superconductivity and Josephson junctions, gaining in charge noise insensitivity [11, 12]. The control, coupling, and measurement are implemented by means of microwave resonators and circuit quantum electrodynamics.

IBM has different *transmon*-based quantum computers in different locations around the world and provides access to these computers via cloud; this availability allows researchers to implement quantum experiments on actual quantum computers via cloud using IBM Q Experience, opening also the way for programmers to run algorithms on quantum computers by using the Python library Qiskit, which allows for the programmer to build quantum circuits in the Python code and manage the cloud-based access for simulation and experiments. The examples addressed in the present section all used Qiskit and two devices were employed: the "IBM Q 5 Tenerife" (ibmqx4)[2] and the "IBM Q 16 Melbourne"[3] (ibmq_16_melbourne).

The "IBM Q 5 Tenerife" device is a 5 *qubit* device with quantum registers labeled from Q0 to Q4, and the connectivity is, according to IBM, provided by two coplanar waveguide (CPW) resonators with resonances around 6.6 GHz (coupling Q2, Q3, and Q4) and 7.0 GHz (coupling Q0, Q1, and Q2).

The "IBM Q 16 Melbourne" device is a 14 *qubit* device with a connectivity that is, in turn, provided by a total of 22 CPW bus resonators each one connecting two quantum registers. For both the Tenerife and Melbourne devices, each quantum register also has a dedicated CPW readout resonator attached for control and readout.

From a computational model standpoint, we can treat the network connections and resulting quantum computing framework, provided by these physical devices, as a form of QUANN, where the conditional neural gates must obey the quantum device's basic topology in what regards the possible quantum controlled gates.

This is so because the quantum registers are linked in specific topologies that limit how conditional quantum operations are implemented; this is a main characteristic of QUANNs, namely, the conditional unitary gates implemented in neural computational circuits are dependent upon the topology and links between the different artificial neurons.

For the simplest algorithms, we can use just a few registers and connections, which means that each quantum device can simulate different QUANNs, within the restrictions of their respective topologies.

For a QUANN using all the quantum registers in the device, the types of algorithms are limited by the device structure, which can only implement a specific neural network topology and link direction.

In **Figures 1** and **2**, we, respectively, show the connectivity structure of the "IBM Q 5 Tenerife" and the "IBM Q 16 Melbourne" devices.

Having introduced the two devices, we now exemplify the theoretical and experimental implementation of a QUANN, on these devices, for a basic problem: the XOR Boolean function representation. This is a relevant example in the artificial neural network (ANN) literature, since the classical feedforward ANN needs a hidden layer to solve this problem, while its quantum counterpart does not [4].

Namely, a three-neuron QUANN with two input neurons feeding forward to a single output neuron is capable of representing the XOR function, while, in the classical case, we need an additional hidden layer comprised of two neurons. This is a feature of QUANNs that is generalizable to other Boolean functions as discussed in [4] regarding the computational efficiency of QUANNs over classical ANNs.

The reason for the greater efficiency is linked to entanglement, namely, the output neuron's firing dynamics can become entangled with the input layer's firing dynamics by way of the implementation of conditional NOT (CNOT) gates,

[2] The relevant elements on this processor, including the quantum circuit structure, can be consulted at the Qiskit backend website: https://github.com/Qiskit/qiskit-backend-information/tree/master/backends/tenerife/V1 (consulted in 21/10/2018)

[3] The relevant elements on this processor, including the quantum circuit structure, can be consulted at the Qiskit backend website: https://github.com/Qiskit/qiskit-backend-information/blob/master/backends/melbourne/V1/README.md (consulted in 21/10/2018).

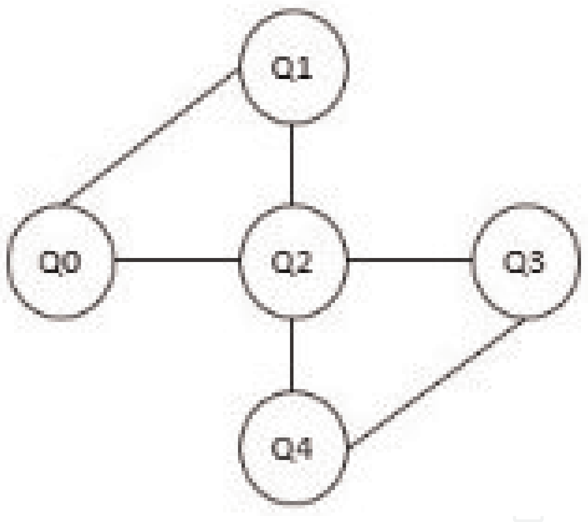

Figure 1.
IBM Q 5 Tenerife (ibmqx4) connectivity structure.

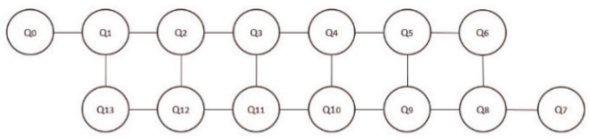

Figure 2.
IBM Q 16 Melbourne (ibmq_16_melbourne) connectivity structure.

providing for an example of the importance of entanglement in the efficiency of quantum computation over classical computation, a point that was object of detailed discussion in [4] regarding the relevance of entanglement for quantum neural computational efficiency.

3.1 The XOR representation problem

The XOR Boolean function representation problem is such that we want an output neuron to fire when the input neurons' firing patterns are reversed and to remain nonfiring when the input neurons' firing patterns are aligned. This means that the neural network's output follows the XOR truth table with the output neuron firing when the XOR function evaluates to "True" and not firing otherwise.

In this case, as shown in [4], the XOR function representation problem can be solved by a standard quantum feedforward neural network with no hidden layer, by taking advantage of quantum entanglement dynamics.

Formally, the quantum circuit, in the forward direction, can be represented by the following chain:

$$\hat{C} = \hat{U}_3 \hat{U}_2 \hat{U}_1 \tag{21}$$

with the gates, respectively, given by:

$$\hat{U}_1 = \hat{U}_{WH} \otimes \hat{U}_{WH} \otimes \hat{1} \tag{22}$$

$$\hat{U}_2 = |0\rangle\langle 0| \otimes \hat{1} \otimes \hat{1} + |1\rangle\langle 1| \otimes \hat{1} \otimes \hat{\sigma}_1 \tag{23}$$

$$\hat{U}_3 = \hat{1} \otimes |0\rangle\langle 0| \otimes \hat{1} + \hat{1} \otimes |1\rangle\langle 1| \otimes \hat{\sigma}_1 \tag{24}$$

We begin with all three neurons in a nonfiring dynamics; then, the propagation from input to output (in the forward direction of the computational circuit) yields:

$$\begin{aligned}
\left\langle s_1s_2s_3\middle|\hat{C}\middle|000\right\rangle &= \langle s_1s_2s_3|\hat{U}_3\,\hat{U}_2\hat{U}_1|000\rangle = \\
&= \langle s_1s_2s_3|\hat{U}_3\hat{U}_2\left(\frac{|000\rangle+|010\rangle+|100\rangle+|110\rangle}{2}\right) = \\
&= \langle s_1s_2s_3|\hat{U}_3\left(\frac{|000\rangle+|010\rangle+|101\rangle+|111\rangle}{2}\right) = \\
&= \langle s_1s_2s_3|\left(\frac{|000\rangle+|011\rangle+|101\rangle+|110\rangle}{2}\right) = \\
&= \frac{\delta_{s_1,0}\delta_{s_2,0}\delta_{s_3,0}}{2}+\frac{\delta_{s_1,0}\delta_{s_2,1}\delta_{s_3,1}}{2}+\frac{\delta_{s_1,1}\delta_{s_2,0}\delta_{s_3,1}}{2}+\frac{\delta_{s_1,1}\delta_{s_2,1}\delta_{s_3,0}}{2} \qquad (25)
\end{aligned}$$

The result in Eq. (25) means that the only probed final alternatives are those where the XOR rule $s_3 = s_1 \oplus s_2$ holds; that is, these are the only alternatives where there is a nonzero amplitude.

Likewise, back propagation from the output to the input yields the same result, that is, the only responses come from outputs where the XOR rule $s_3 = s_1 \oplus s_2$ holds, as the following derivation shows:

$$\begin{aligned}
\left\langle 000\middle|\hat{C}^{\dagger}\middle|s_1s_2s_3\right\rangle &= \langle 000|\hat{U}_1{}^{\dagger}\,\hat{U}_2{}^{\dagger}\hat{U}_3{}^{\dagger}|s_1s_2s_3\rangle = \\
&= \delta_{s_2,0}\langle 000|\hat{U}_1{}^{\dagger}\hat{U}_2{}^{\dagger}|s_1s_2s_3\rangle + \delta_{s_2,1}\langle 000|\hat{U}_1{}^{\dagger}\hat{U}_2{}^{\dagger}|s_1s_21-s_3\rangle = \\
&= \delta_{s_1,0}\delta_{s_2,0}\langle 000|\hat{U}_1{}^{\dagger}|s_1s_2s_3\rangle + \delta_{s_1,0}\delta_{s_2,1}\langle 000|\hat{U}_1{}^{\dagger}|s_1s_21-s_3\rangle + \\
&+\delta_{s_1,1}\delta_{s_2,0}\langle 000|\hat{U}_1{}^{\dagger}|s_1s_21-s_3\rangle + \delta_{s_1,1}\delta_{s_2,1}\langle 000|\hat{U}_1{}^{\dagger}|s_1s_2s_3\rangle = \\
&= \frac{\delta_{s_1,0}\delta_{s_2,0}\delta_{s_3,0}}{2}+\frac{\delta_{s_1,0}\delta_{s_2,1}\delta_{s_3,1}}{2}+\frac{\delta_{s_1,1}\delta_{s_2,0}\delta_{s_3,1}}{2}+\frac{\delta_{s_1,1}\delta_{s_2,1}\delta_{s_3,0}}{2} \qquad (26)
\end{aligned}$$

Replacing Eqs. (25) and (26) in the general Eq. (17) yields, for this quantum circuit, the *echo intensities*:

$$\langle s_1s_2s_3|\hat{\rho}_{out}|s_1s_2s_3\rangle = \left\langle s_1s_2s_3\middle|\hat{C}\middle|000\right\rangle\left\langle 000\middle|\hat{C}^{\dagger}\middle|s_1s_2s_3\right\rangle = \frac{\delta_{s_3,s_1\oplus s_2}}{4} \qquad (27)$$

That is, the forward and back propagation is such that the *echoes* are only formed for the cases where the rule $s_3 = s_1 \oplus s_2$ holds, leading to a ¼ probability associated with each alternative firing pattern of the first two neurons.

The **Figure 3** shows the theoretical results from the above equations, the simulation in the IBM quantum assembly language (QASM) simulator and the experimental implementation on the Tenerife (ibmqx4) and Melbourne (ibmq_16_melbourne) devices.

The QASM simulation expresses, as expected, the basic random results from the repeated experiments, which is associated with the fundamental stochastic dynamics underlying quantum processing; however, the simulator results agree with the theoretical results, so that the basic XOR computation holds, that is, in each case, the output neuron exhibits the firing pattern that is consistent with the XOR rule.

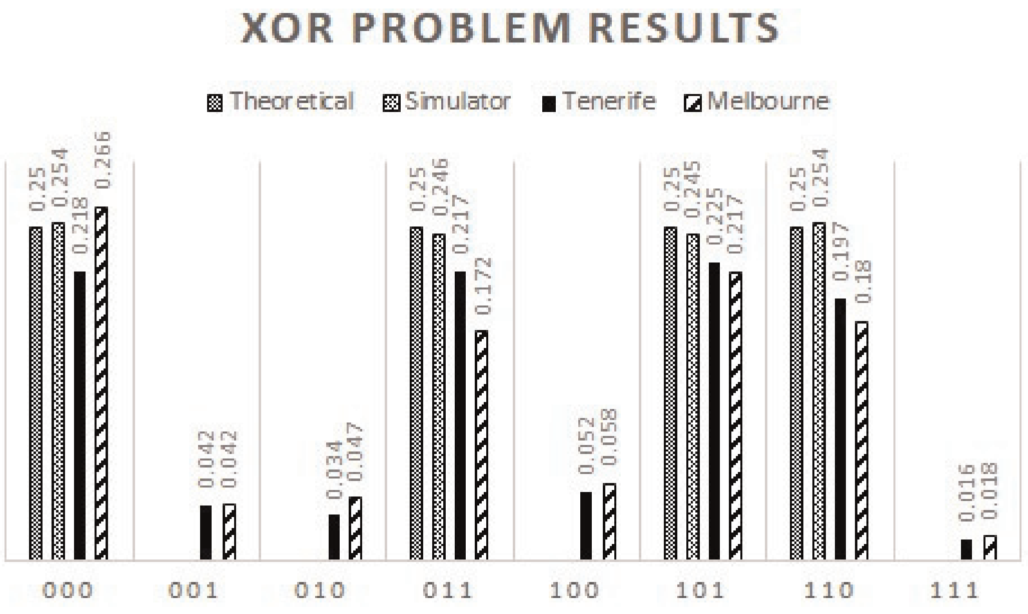

Figure 3.
Theoretical and experimental implementation of the XOR representation problem on the QASM simulator, Tenerife and Melbourne devices, with 8192 shots.

In the case of experiments, the XOR rule is predominant, that is, the dominant frequencies are those consistent with the circuit; there are, however, also a few residual cases that deviate from the XOR rule, all with low relative frequencies. These deviations are to be expected on the actual physical devices. For the Tenerife device, the relative frequency of cases that follow the XOR rule is 0.857; for the Melbourne device, this relative frequency is 0.835.

One of the main problems in physical implementation of quantum computation is the presence of errors. Indeed, the equations are derived for an isolated circuit so that the only *echoes* are those matching the quantum circuit; therefore, in an isolated QUANN, the stochastic results from repeated trials tend, in a frequentist approach, to the actual probabilities with zero frequencies associated with the alternatives for which no *echo* is produced. This is a basic property of quantum mechanics as predicted by the theory, and explains that the QASM simulator gets a zero measure for those alternatives for which there is no *echo*.

Of course, if Bohm's conjecture regarding the subquantum dynamics [9, 10] is right, then, even for a sufficiently isolated circuit, small deviations coming from the subquantum level may be present and lead to *echoes* that do not correspond to those of the main computing circuit. In any other interpretation that does not assume a subquantum dynamics and that takes the formalism to be exact, then, such deviations, for an isolated system, are considered physically impossible.

While we cannot rule out Bohm's subquantum hypothesis, we cannot also confirm it for now, since one never has a completely isolated circuit, and both conjectural lines (Bohmian and others) agree that some deviations on physical devices will always be present due to the environment.

The differences between the two conjectural lines, for quantum computer science, are worth considering regarding quantum error correction; however, for now, in regard to the technological issue of quantum error correction, we cannot yet make use of Bohm's conjecture that the quantum probabilities are average quantities and that subquantum fluctuations may introduce small deviations that average out at the quantum level to lead to the main experimental agreement with the theory.

Having provided, through the XOR problem, an example of how quantum neural computation can be run experimentally on IBM's quantum devices, we now address artificial intelligence (AI) applications; we are interested in the theoretical and experimental implementation of a form of reinforcement learning using QUANNs, namely the quantum neural reinforcement learning (QNRL) and its connection to quantum robotics and quantum adaptive computation.

3.2 Quantum neural reinforcement learning, robotics, and quantum adaptive computation

Quantum robotics involves the need for the development of quantum adaptive algorithms that allow the robot to process alternatives and select appropriate actions using quantum rules [6, 13–15], that is, to incorporate decisions in quantum AI. In this context, there are two major types of artificial agents that one may consider:

- classical agents that implement classical actions but whose cognitive substrate is quantum computational;

- quantum agents that implement quantum operations on a quantum target.

The first type of agent is addressed as a classical robot dealing with problems at a classical level but whose computational substrate is run via cloud access on a quantum computer, thus, pointing toward a possible future where quantum computation is incorporated on different robotic systems by way of the internet of things and cloud-based access to quantum devices.

The second type of agents corresponds to *quantum software robots* (*quantum bots*) that are implemented within a quantum computer and can be used for the adaptive management of target quantum registers and for the purpose of more complex adaptive computation [6, 13–15].

This second type of agents forms the basis for AI solutions aiming at intelligent quantum computing systems with application in quantum internet technologies and, also, possible quantum adaptive error correction.

This latter point (quantum adaptive error correction) must draw specifically on the empirical implementation in physical devices, since it is this implementation that may ultimately test the best adaptive algorithms for quantum error correction. A basic direction, in this case, regards *echo* strengthening, in order to diminish the *echoes* coming from alternatives that do not fall in an intended computation.

We do not address this last point here, but rather illustrate the implementation of the first type of agent in the context of an adaptive computation of a classical gamble, namely, optimal action selection in a classical gambling problem through quantum neural reinforcement learning (QNRL).

In this case, the artificial agent is dealing with a classical problem and implementing its decision processing on a QUANN, namely, the agent has an action set described by 2^d binary strings; following an evolutionary computation framework, we use d-length genetic codes to address actions, so that the actions' codes are comprised of d loci, each with two alleles, 0 and 1.

Now, given each alternative action, the agent is offered a classical gamble on a measurable space $(\Omega, \wp)$ where $\wp$ is a sigma-algebra of subsets of Ω and $\Omega = \{w_0, w_1, \dots, w_{N-1}\}$ is the set of rewards for the gamble, which we consider, in this example, to be discrete, although the results also apply to continuous reward spaces and (classical) probability distributions.

Now, for each action genetic code $\mathbf{s} \in A_2^d$ there is a corresponding gamble probability measure $P_\mathbf{s}$ that is offered to the agent, so that the conditional expected value for the reward w can be calculated as:

$$E[w|\mathbf{s}] = \sum_{n=0}^{N-1} w_n P_{\mathbf{s}}[w_n] \tag{28}$$

The goal for the agent is to select the action that maximizes this conditional expected reward, that is:

$$\mathbf{s}^* = \arg\max_{\mathbf{s}} E[w|\mathbf{s}] \tag{29}$$

To solve the optimization problem in Eq. (29), we use a variant of QNRL, which applies modular networked learning [16], in the sense that, instead of a single neural network for a single problem, we expand the cognitive architecture and work with a modular system of neural networks.

Modular networked learning (MNL) was addressed in [16] and applied to financial market prediction, where, instead of a single problem and a single target, one uses an expanded cognitive architecture to work on multiple targets with a module assigned to each target and possible links between the modules used to map links between subproblems of a more complex problem.

For modular neural networks, the resulting cognitive architecture resembles an artificial brain with specialized "brain regions" devoted to different tasks and connections between different neural modules corresponding to connections between different brain regions. In the present case, the agent's "artificial brain" (as shown in **Figure 4**) is comprised of three "brain regions" connected with each other for a specific functionality, where the first module (first brain region) corresponds to the action exploration region, the second module (second brain region) corresponds to the reward processing region, and the third module to the decision region.

The connections between the modules follow the hierarchical process associated with the necessary quantum reinforcement learning for each action, **Figure 4** expresses this relation. The reinforcement learning, in this case, is a form of quantum search, implemented on the above modular structure, that proceeds in two stages: the exploration stage and the exploitation stage.

In the exploration stage, the agent's first brain region, taking advantage of quantum superposition, explores with equal weights, in parallel, each alternative initial action and the second brain region processes the conditional expected rewards; this last processing is based on optimizing quantum circuits [6], where

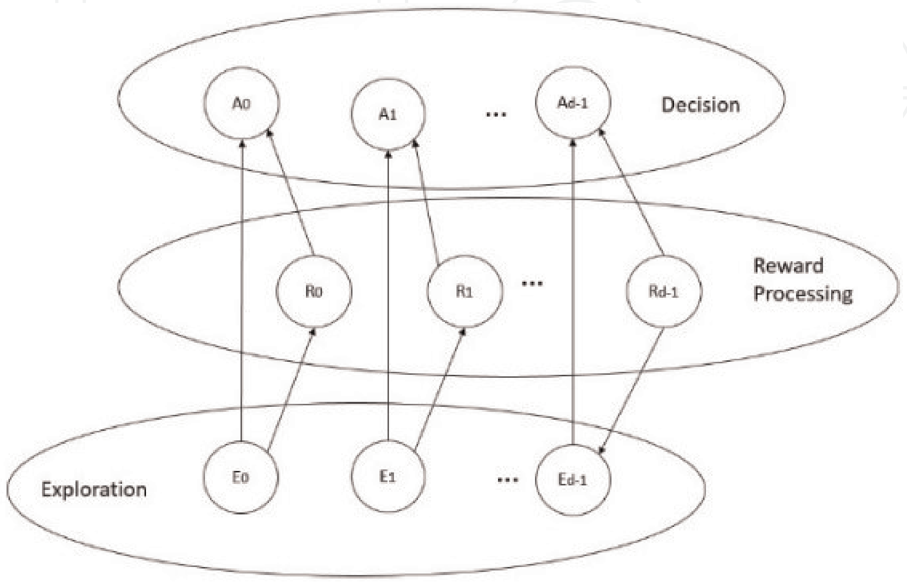

Figure 4.
Modular structure for the reward maximization problem.

the unitary operator for the second brain region incorporates the optimization itself.

The second brain region will work as a form of oracle in the remaining adaptive computation and allows for the agent's artificial brain to implement an optimal expected reward-seeking dynamics.

Now, in the second phase of the exploration stage, the synaptic connections from the first to the second brain region are activated, leading to a quantum entangled dynamics between the two brain regions, where the first region acts as the control (input layer) and the second as the target (output layer).

Thus, at the end of the exploration stage, the first two brain regions exhibit an entangled dynamics. This is a basic point of quantum strategic cognition, in the sense that the processing of the alternative courses of action is not localized in a specific neuron or neurons, but rather it leads to quantum correlations between different brain regions; these connections allow the artificial brain to efficiently select the best course of action, from the evaluation of the alternatives and rewards.

In the exploitation stage, the synaptic connections from the first brain region (the action exploration region) to the third brain region (the decision region) are activated first, so that the decision region is first processing the explored alternative actions, becoming entangled with the action exploration region; then, the synaptic connections between the reward processing region and the decision region are activated for the conditional expected reward processing by the decision module. In this way, the decision module makes the transition for the optimal action, consulting the "oracle" (which is the reward processing module) only once.

The artificial brain thus takes advantage of quantum entanglement in order to adaptively output the optimal action. Formalizing this dynamics, the artificial brain is initialized in a nonfiring probe and response dynamics so that the initial density is:

$$\hat{\rho}_0 = |0\rangle\langle 0|^{\otimes 3d} \tag{30}$$

Now, we denote by $\mathbf{s}_k^*$ the *k-th* bit in the string $\mathbf{s}^*$, and use the following notation for the maximization in Eq. (29) evaluated at the *k-th* bit:

$$\mathbf{s}_k^* = \underset{\mathbf{s},\,k}{\arg\max}\, E[w|\mathbf{s}] \tag{31}$$

Using this notation, the first phase of the exploration stage is given by the unitary operator:

$$\hat{U}_1 = \\ = \hat{U}_{WH}{}^{\otimes d} \otimes_{k=1}^{d} \left(\cos\left(\frac{\underset{\mathbf{s},\,k}{\arg\max}\, E[w|\mathbf{s}]}{2} \pi \right) \hat{1} - i \sin\left(\frac{\underset{\mathbf{s},\,k}{\arg\max}\, E[w|\mathbf{s}]}{2} \pi \right) \hat{\sigma}_2 \right) \otimes \hat{1}^{\otimes d} \tag{32}$$

The operator incorporates the optimization dynamics into the conditional quantum gates' parameters themselves. Since we have:

$$\left(\cos\left(\frac{\underset{\mathbf{s},\,k}{\arg\max}\, E[w|\mathbf{s}]}{2} \pi \right) \hat{1} - i \sin\left(\frac{\underset{\mathbf{s},\,k}{\arg\max}\, E[w|\mathbf{s}]}{2} \pi \right) \hat{\sigma}_2 \right) |0\rangle \tag{33}$$

$$= \cos\left(\frac{\underset{\mathbf{s},\,k}{\arg\max} E[w|\mathbf{s}]}{2}\pi\right)|0\rangle + \sin\left(\frac{\underset{\mathbf{s},\,k}{\arg\max} E[w|\mathbf{s}]}{2}\pi\right)|1\rangle = |\mathbf{s}_k^*\rangle$$

after the first phase of the of exploration stage, the resulting density is given by:

$$\hat{\rho}_1 = \hat{U}_1\hat{\rho}_0\hat{U}_1^{\dagger} = |+\rangle\langle+|^{\otimes d} \otimes |\mathbf{s}^*\rangle\langle\mathbf{s}^*| \otimes |0\rangle\langle 0|^{\otimes d} \quad (34)$$

Thus, the neural field is probing, for the first brain region, each alternative neural pattern (each alternative action) with equal weight, the response dynamics also comes, for the first brain region, from each alternative neural pattern with equal weight, which means that the *echoes* for the first brain region are independent from the *echoes* for the remaining brain regions and show an equal intensity associated with each alternative neural pattern.

On the other hand, for the second brain region, the neural field exhibits a reward-seeking dynamics that is adaptive with respect to the optimal action; that is, the probing dynamics is directed toward the optimal action and the response dynamics also comes from the optimal action, so that, due to the adaptive unitary propagation, the second brain region is projecting over the optimum value, and this is the only *echo* that it gets.

The third brain region still has a projective dynamics toward the nonfiring neural activity.

Now, for the second phase of the exploration stage, we have the operator:

$$\hat{U}_2 = \left[\sum_{\mathbf{s}\in A_2^d} |\mathbf{s}\rangle\langle\mathbf{s}| \otimes_{k=1}^{d} \left((1-s_k)\hat{1} + s_k\hat{\sigma}_1\right)\right] \otimes \hat{1}^{\otimes d} \quad (35)$$

which leads to the density after the second phase of the exploration stage:

$$\hat{\rho}_2 = \hat{U}_2\hat{\rho}_1\hat{U}_2^{\dagger} = \sum_{\mathbf{r},\,\mathbf{s}\in A_2^d} \frac{|\mathbf{r}\rangle\langle\mathbf{s}| \otimes_{k=1}^{d} |\mathbf{s}_k^* \oplus r_k\rangle\langle\mathbf{s}_k^* \oplus s_k|}{2^d} \otimes |0\rangle\langle 0|^{\otimes d} \quad (36)$$

Thus, after the second phase, the first and second brain regions exhibit an entangled probe and response dynamics, where the neural field, for second brain region, is effectively computing both the rewards and the explored actions.

Next comes the exploitation stage with the neural processing for the decision module (the third brain region).

The first step of the exploitation stage is the processing of the initially explored actions, by way of the operator:

$$\hat{U}_3 = \sum_{\mathbf{s}\in A_2^d} |\mathbf{s}\rangle\langle\mathbf{s}| \otimes \hat{1}^{\otimes d} \otimes_{k=1}^{d} \left((1-s_k)\hat{1} + s_k\hat{\sigma}_1\right) \quad (37)$$

which leads to the density:

$$\hat{\rho}_3 = \hat{U}_3\hat{\rho}_2\hat{U}_3^{\dagger} = \sum_{\mathbf{r},\,\mathbf{s}\in A_2^d} \frac{|\mathbf{r}\rangle\langle\mathbf{s}| \otimes_{k=1}^{d} |\mathbf{s}_k^* \oplus r_k\rangle\langle\mathbf{s}_k^* \oplus s_k| \otimes |\mathbf{r}\rangle\langle\mathbf{s}|}{2^d} \quad (38)$$

That is, the probe and response dynamics for the third brain region are correlated and coincident with the probe and response dynamics for the first brain region, so that the third brain region is effectively computing the initially explored actions.

Now, the second step for the third brain region results from the activation of the synaptic links with the second brain region, leading to the conditional unitary operator:

$$\hat{U}_4 = \sum_{\mathbf{s}\in A_2^d} \hat{1}^{\otimes d} \otimes |\mathbf{s}\rangle\langle\mathbf{s}| \otimes_{k=1}^{d} \left((1-s_k)\hat{1} + s_k\hat{\sigma}_1\right) \tag{39}$$

Under this operator, we get the final density:

$$\hat{\rho}_4 = \hat{U}_4\hat{\rho}_3\hat{U}_4{}^{\dagger} = \\ = \sum_{\mathbf{r},\,\mathbf{s}\in A_2^d} \frac{|\mathbf{r}\rangle\langle\mathbf{s}| \otimes_{k=1}^{d}\left(|\mathbf{s}_k^* \oplus r_k\rangle\langle\mathbf{s}_k^*\oplus s_k| \otimes |r_k \oplus (\mathbf{s}_k^*\oplus r_k)\rangle\langle s_k \oplus (\mathbf{s}_k^*\oplus s_k)|\right)}{2^d} \tag{40}$$

Since we have the Boolean equality $p \oplus (q \oplus p) = q$, this means that the above density can be simplified, so that the neural field's probe and response dynamics for the third brain region projects over the optimal action:

$$\hat{\rho}_4 = \hat{U}_4\hat{\rho}_3\hat{U}_4{}^{\dagger} = \left(\sum_{\mathbf{r},\,\mathbf{s}\in A_2^d} \frac{|\mathbf{r}\rangle\langle\mathbf{s}| \otimes_{k=1}^{d} |\mathbf{s}_k^*\oplus r_k\rangle\langle\mathbf{s}_k^*\oplus s_k|}{2^d}\right) \otimes |\mathbf{s}^*\rangle\langle\mathbf{s}^*| \tag{41}$$

The third brain region's computation takes advantage of the entangled dynamics between the first and second brain regions to learn the optimal action. For the final density, while the first and second brain regions exhibit an entangled probe and response dynamics, the third brain region is always projecting over the optimum.

It is important to stress how QNRL takes advantage of quantum entanglement such that the neural field for the third brain region followed each alternative action and then the reward processing dynamics to find the optimum in all these alternative paths, so that the optimal action is always followed by the agent.

As an example of the above problem, let us consider the case where we the reward set is $\Omega = \{-1, 1\}$, and that there are two possible actions 0 and 1 leading, respectively, to the classical probability measures P_0 and P_1, with $P_0[w = 1] = 0.4$ and $P_1[w = 1] = 0.6$; then, we get the probabilities of selection for each gamble and device shown in **Table 1**, for 8192 repeated experiments.

As expected, the QASM simulator always selects the action 1, which is the best performing action by the conditional expected payoff criterion. The Tenerife device selects the correct action with a proportion of 0.778, while the Melbourne device selects the correct action with a proportion of 0.647. If, instead of the above gamble profile, we had $P_0[w = 1] = 0.6$ and $P_1[w = 1] = 0.4$, the optimal choice would be the action 0; in this case, as shown in **Table 2**, the QASM simulator, again, selects the correct action each time. The Tenerife device, in turn, selects the correct action with a 0.857 frequency and the Melbourne device with a 0.814 frequency.

In **Figure 5**, we show the Melbourne device's results[4] when we have four actions for the same rewards profile, and the probabilities are $P_{00}[w = 1] = 0.6$, $P_{01}[w = 1] = 0.4$, $P_{10}[w = 1] = 0.8$, and $P_{11}[w = 1] = 0.9$, still setting the *rewards* to $\Omega = \{-1, 1\}$.

In this case, if we run the experiment on the QASM backend, with 8192 shots, we get the action encoded by the string 11 with relative frequency equal to 1, which

[4] We can only use the Melbourne device since the Tenerife device does not have the required capacity in terms of number of quantum registers.

Device	Action	
	0	**1**
QASM	0	1
Tenerife	0.222	0.778
Melbourne	0.353	0.647

Table 1.
Results for two alternative actions using the QASM simulator, the Tenerife device (ibmqx4) and the Melbourne device (ibmq_16_melbourne); in each case, 8192 shots were used, with $P_0[w = 1] = 0.4$ and $P_1[w = 1] = 0.6$.

Device	Action	
	0	**1**
QASM	1	0
Tenerife	0.857	0.143
Melbourne	0.814	0.186

Table 2.
Results for two actions using the QASM simulator, the Tenerife device (ibmqx4) and the Melbourne device (ibmq_16_melbourne); in each case, 8192 shots were used, with $P_0[w = 1] = 0.6$ and $P_1[w = 1] = 0.4$.

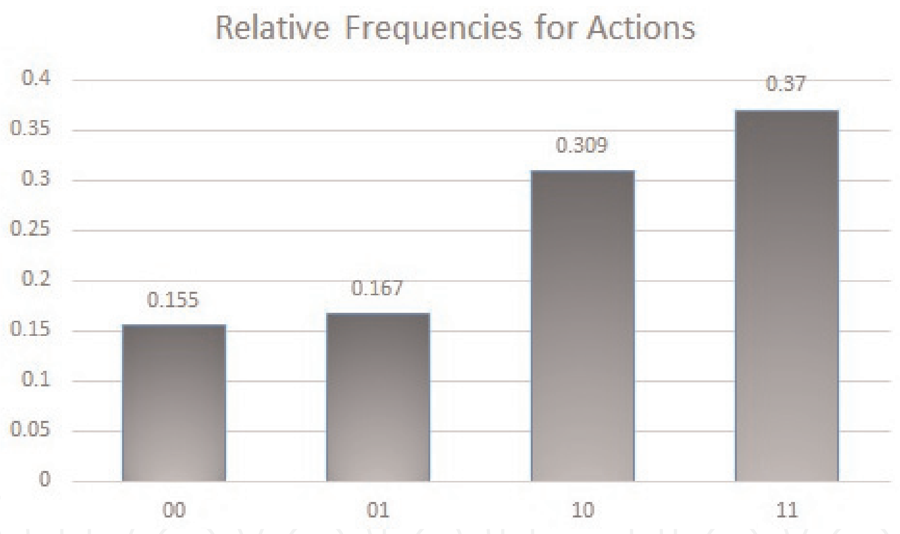

Figure 5.
Results for four actions using the Melbourne device ("ibmq_16_melbourne"), with 8192 shots used, and probability profiles given by: $P_{00}[w = 1] = 0.6$, $P_{01}[w = 1] = 0.4$, $P_{10}[w = 1] = 0.8$ and $P_{11}[w = 1] = 0.9$.

is the optimal action. If we run the experiment with the same number of shots on the Melbourne device, then, as shown in **Figure 5**, the output 11 is still the dominant action, however, with a proportion of 0.370, the second dominant action being non-residual and with a value of 0.309 occurs for the output 10.

Therefore, the first *qubit* tends to be measured with the right pattern with a proportion of 0.679 (0.309 + 0.370); the probability of the second *qubit* being correct given that the first is correct is only about 0.54492 (0.370/0.679). This suggests that the deviation may be due to the entanglement with the environment significantly deviating the second *qubit* from the correct pattern.

The above algorithm was implemented using Qiskit and Python's Object Oriented Programming (OOP); the code, shown in the appendix, exemplifies how OOP can be integrated with quantum computation for implementing quantum AI on any terminal, due to the cloud access to IBM's quantum resources, constituting an example of Quantum Object Oriented Programming (QOOP) using Qiskit.

The code defines the class "Agent" with an attribute that is the quantum neural network; in this case, the attribute will be assigned a quantum circuit with the required quantum and classical registers.

There are two methods that any instance of the class Agent must be able to implement: the first method manages the cloud access to IBM's resources, the second method implements the action selection and the quantum algorithm.

The inputs for the first method are the accounts to be loaded, for the classical computer to be able to access quantum computer via the cloud service, and the backend code, which, by default, is set to the QASM simulator but can be changed to any of the devices. The method returns the backend to be used.

The second method, for the action selection, has a structure that is specific to the problem in question; that is, the agent is offered a set of rewards and probabilities associated with each alternative action, and must choose the action that maximizes the conditional expected reward.

Thus, the probabilities are known to the agent and form part of the gamble that is offered to it; therefore, we are dealing with a decision problem under risk, and wish to address how the agent's QUANN can exhibit an adaptive computation with respect to this problem.

While, in the above equations, the adaptive nature of the quantum neural circuit was introduced in the unitary operator's parameters themselves, the Python code for the method must use the gamble's inputs to make the quantum circuit adaptive; that is, the method must be such that the agent designs its own cognitive architecture (updating its qnnet attribute) and quantum circuit using the inputs to the method, and, then, the agent must implement the cloud-based access to run, in IBM's quantum computers, the corresponding quantum algorithm.

The inputs for the method are, then, given by a list of probability distributions, where each line corresponds to a different probability gamble profile associated with each action, for instance, in the case of **Table 1**, the distributions are given by ([0.6, 0.4],[0.4, 0.6]). In the case of **Tables 1** and **2** and **Figure 5**, the rewards list is $[-1, 1]$.

The other two inputs for the method are the backend used which allows the agent that is instantiated in a classical computer to access via cloud the quantum computer, using the backend code (backend_used) and repeatedly running the algorithm on the respective device for a number of shots (num_shots).

The choose_action method's step zero is the extraction of the expected values and of the corresponding parameters for the adaptive gates, namely, the expected values array associated with each action is extracted by the agent using the Python library NumPy's dot product applied to the distributions and rewards lists.

The number of actions and dimension d that determines the network size are extracted from the length of the expected values array; then, the parameters for the adaptive gates are extracted by applying NumPy's argmax function on the expected values array and then converting the resulting index in binary format (using NumPy's binary_repr). Since the indexes match the lexicographic order of binary strings, the agent, thus, effectively extracts the parameters for the adaptive unitary gates.

Now, the next step is to set up the QUANN, including the three modules, the classical registers for the measurement of the final actions to be chosen and updating the agent's qnnet attribute, assigning it the corresponding Qiskit's quantum circuit object.

The last step implements the QNRL algorithm, following the inter-module links as per the main equations introduced in this section, and defines the quantum measurement for the decision module, executing the algorithm on the backend (taking advantage of the cloud access) and plotting the histogram to extract the main experimental relative frequencies obtained from the repeated experiments (the number of shots).

4. Conclusions

Cloud-based access to quantum computers opens up a major point: the empirical testing of algorithms and the implementation of computer programs in a quantum computational substrate has become feasible.

The IBM Q Experience constitutes an example of how a programmer can use Python programming language and IBM's Python Qiskit package for building programs that use quantum computation, limited only by the specific device resources, namely the number of quantum registers available.

For quantum AI and machine learning, this provides a way to effectively bring the algorithms from the theoretical level to the test level, allowing one to *test drive* different quantum AI frameworks on actual quantum computers. The work developed in the previous sections allowed us to provide several examples of such an implementation, with a few main points standing out:

- We showed how one can address IBM's superconducting *transmon devices* as examples of QUANNs, since, just as in a QUANN, the *devices* can only implement the conditional gates depending on the network topology and the directions of the links, which only allow for specific conditional gates to be implemented; as an example, the Tenerife device is a *bowtie feedforward network*, we cannot turn it into a recurrent network so that the gates have to be implemented following specific directions of the links (this limit can be experienced by any user that accesses the online resources and tries to visually build circuits in IBM Q Experience homepage).

- We exemplified how basic Boolean functions' representation, in this case the XOR function, can be implemented on a (physical) quantum computer using the cloud access to Tenerife and Melbourne devices and compared the experimental results with the theoretical derivation; a relevant point of this is that we only needed three quantum registers and no hidden layer to solve the XOR problem, a point already raised about this function and generalized to other functions in [4], regarding the theoretical efficiency of QUANNs.

- We addressed how a form of quantum adaptive computation, incorporating a reward-seeking behavior and a variant of QNRL, can be implemented, in the context of quantum robotics and AI, on different quantum devices.

The three main points above help strengthen two core arguments: the first is that quantum machine learning can now be tested on actual quantum computers, making it feasible to empirically test the algorithms; the second is that, in the near future, with further advancements in quantum computation and quantum hardware, quantum adaptive computation may be implemented on actual robots with a quantum cognitive architecture that is based on cloud access to a quantum computer.

The present work addresses both core arguments by exemplifying how a form of QNRL can be employed to implement quantum adaptive computation on a physical QUANN with cloud-based access, employing QOOP and addressing, experimentally, a decision under risk problem.

A. Python Code for Quantum Neural Reinforcement Learning Problem.

```
# Import NumPy and Qiskit's main functionalities
import numpy as np
from qiskit import ClassicalRegister, QuantumRegister, QuantumCircuit
from qiskit import execute
from qiskit import IBMQ
from qiskit.tools.visualization import plot_histogram
class Agent:
def __init__(self, qnnet):
self.qnnet = qnnet # agent's Quantum Neural Network
def get_backend(self,
load_accounts = True, # if accounts are to be loaded
backend_code = 'ibmq_qasm_simulator' # backend code
):
# Load IBM account if needed
if load_accounts == True:
IBMQ.load_accounts()
# Get the backend to use in the computation
backend_used = IBMQ.get_backend(backend_code)
# If one is not using the QASM simulator get the backend status
if backend_code! = 'ibmq_qasm_simulator':
print(backend_used.status())
# Return the backend used
return backend_used
def choose_action(self,
distributions, # probability distributions
rewards, # reward system
backend_used, # backed to be used
num_shots): # number of shots to run in quantum computer
# Step 0: get the expected values and unitary parameters:
# Get the expected values
expected_values = np.dot(distributions,rewards)
# Get the number of actions involved
num_actions = len(expected_values)
# Get the base number that we will need for the network size
dim = int.(np.log2(num_actions))
# Get the parameters for the adaptive gate
maxstring = np.binary_repr(np.argmax(expected_values), width = dim)
# Step 1: Setup the Quantum Artificial Neural Network:
# Get the number of quantum registers
q = QuantumRegister(3*dim)
# Get the number of classical registers
c = ClassicalRegister(dim)
# Setup the quantum neural network
self.qnnet = QuantumCircuit(q, c)
# Step 2: Implement the Reinforcement Learning Algorithm:
# Exploration Stage
for i in range(0,dim):
self.qnnet.h(q[i])
for j in range(0,dim):
self.qnnet.u3(float(maxstring[j])*np.pi,0,0,q[dim+j])
for k in range(0,dim):
```

```
self.qnnet.cx(q[k],q[dim+k])
# Exploitation Stage
for l in range(0,dim):
self.qnnet.cx(q[l],q[2*dim+l])
self.qnnet.cx(q[dim+l],q[2*dim+l])
# Quantum Measurement
for m in range(0,dim):
self.qnnet.measure(q[2*dim+m], c[dim-1-m])
# Execute the algorithm on the backend
job_exp = execute(self.qnnet, backend = backend_used, shots = num_shots)
# Plot the histogram
plot_histogram(job_exp.result().get_counts(self.qnnet))
```

Author details

Carlos Pedro dos Santos Gonçalves
University of Lisbon, Institute of Social and Political Sciences, Lisbon, Portugal

*Address all correspondence to: cgoncalves@iscsp.ulisboa.pt

References

[1] Menneer T. Quantum artificial neural networks [thesis]. Exeter: The University of Exeter; 1998

[2] Narayanan A, Meneer T. Quantum artificial neural network architectures and components. Information Sciences. 2000;**128**:231-255. DOI: 10.1016/S0020-0255(00)00055-4

[3] Schuld M, Sinayskiy I, Petruccione F. The quest for a quantum neural network. Quantum Information Processing. 2014;**13**(11):2567-2586. DOI: 10.1007/s11128-014-0809-8

[4] Gonçalves CP. Quantum cybernetics and complex quantum systems science: A quantum connectionist exploration. NeuroQuantology. 2015;**13**(1):35-48. DOI: 10.14704/nq.2015.13.1.804

[5] Gonçalves CP. Quantum neural machine learning: Backpropagation and dynamics. NeuroQuantology. 2017; **15**(1):22-41. DOI: 10.14704/nq.2017.15.1.1008

[6] Gonçalves CP. Quantum robotics, neural networks and the quantum force interpretation. NeuroQuantology. ISSN: 1303 5150

[7] Cramer JG. The Quantum Handshake: Entanglement, Nonlocality and Transactions. Cham: Springer. p. 218. DOI: 978-3-319-24640-6

[8] Everett H. Relative state' formulation of quantum mechanics. Reviews of Modern Physics. 1957;**29**(3):454-462. DOI: doi.org/10.1103/RevModPhys.29.454

[9] Bohm D. Causality and Chance in Modern Physics. London: Routledge; 1997 [1957]. p. 189. ISBN: 0-415-17440-6

[10] Bohm D, Hiley BJ. The Undivided Universe. New York: Routledge; 1995. p. 409. ISBN: 978-0415121859

[11] Koch J, Yu TM, Gambetta J, Houck AA, Schuster DI, Majer J, et al. Charge-insensitive qubit design derived from the Cooper pair box. Physical Review A. 2007;**76**:042319. DOI: 10.1103/PhysRevA.76.042319

[12] Schreier JA, Houck AA, Koch J, Schuster DI, Johnson BR, Chow JM, et al. Suppressing charge noise decoherence in superconducting charge qubits. Physical Review B;**77**:180502(R). DOI: 10.1103/PhysRevB.77.180502

[13] Benioff P. Quantum robots and environments. Physical Review A. 1998; **58**(2):893-904. DOI: 10.1103/PhysRevA.58.893

[14] Benioff P. Some foundational aspects of quantum computers and quantum robots. Superlattices and Microstructures. 1998;**23**(3–4):407-417. DOI: 10.1006/spmi.1997.0519

[15] Dong D-Y, Chen C-L, Zhang C-B, Chen Z-H. Quantum robot: Structure, algorithms and applications. Robotica. 2006;**24**(4):513-521. DOI: 10.1017/S0263574705002596

[16] Gonçalves CP. Financial Risk and Returns Prediction with Modular Networked Learning. arXiv: 1806.05876 [cs.LG]. 2018. Available from: https://arxiv.org/pdf/1806.05876.pdf [Accessed: 28-10-2018]