Solving the Qubit Transmission Problem with Biologically Inspired Neural Networks and Reinforcement Learning

Ariel Fiuri[2], Martín Domínguez[2], Sofía Perón Santana[1,2], and Omar Osenda[1,2]

[1] Instituto de Física Enrique Gaviola (CONICET-UNC)
[2] Facultad de Matemática, Astronomía, Física y Computación (UNC)
{ariel.fiuri,martin.dominguez,omar.osenda}@unc.edu.ar
sofia.peron@mi.unc.edu.ar

Abstract. Over the last few years, machine learning methods have used deep neural network architectures to tackle complex problems. In this paper, we applied biologically inspired artificial neural machine learning to solve a specific task from quantum physics, i.e., quantum state transfer in a one-dimensional spin chain. The mission is to find transfer schemes with high transfer fidelities. We use different methods to learn a policy able to yields a good solution. As a first approach, we combine some architectures for Fiuri Artificial Neural Networks (FANN) with Reinforcement Learning (RL) and optimization techniques to train models, and as a second approach, we find solutions with different RL tabular methods. We use transfer fidelity as a metric to make a detailed comparison of the obtained results, combining different architectures and optimization methods for the FANN approach and different algorithms for the tabular one. We obtained very competitive results compared with the solutions provided in the literature.

Keywords: Artificial Neural Network · Reinforcement Learning · Optimization · Neural Dynamics · Biologically Inspired · Quantum Transmission State.

1 Introduction

The outstanding control accuracy needed to implement quantum information processing in actual implementations is one of the practical obstacles to advancing in the application of quantum technologies to everyday problems.

Even in the absence of external perturbations, the controlled time evolution of quantum systems is not always achievable. For a closed quantum system, *i.e.*. without perturbations, the Hamiltonian operator dictates the time-evolution of the system, accordingly with the postulates of Quantum Mechanics. Quantum information processing involves three simple steps.

First, prepare the initial state. Then, let the Hamiltonian drive the time evolution of the system until the quantum state contains the information with the answer to the problem of interest. The final step usually requires measuring the final state. In most cases, obtaining an accurate answer requires repeating the whole procedure. Sometimes, restricting the problem to study simplifies the task at hand. That is the case in the transmission (or transfer) of quantum states. In this context, an array of simple physical systems interacting with each other acts like a transmission channel. The preferred geometry for this channel is a linear array or chain. A chain of qubits is an array of the simplest quantum mechanical systems. From the mathematical point of view, there is no simpler system than one whose quantum state belongs to a two-dimensional complex vector space, as is the case with a qubit.

So, the system of interest is a chain of equally spaced qubits. The chain has a Hamiltonian independent of time. Besides, we consider that a given qubit interacts only with its two closest neighbors. For this setting, the only way to successfully transfer a quantum state from one extreme of the chain to the other is by controlling external actions applied over the qubits, except for a handful of cases where perfect state transfer results from the time evolution given by the time-independent Hamiltonian. The simplest protocol of transmission only considers the transmission of an arbitrary one-qubit state. In this case, the quantum state of the qubit chain is a vector with N complex components, and the Hamiltonian is an NxN Hermitian matrix, for a chain with N qubits. So, to control the time evolution of the chain, we should consider the number and type of actions applied. Guided by the actions implementable in actual physical systems, we will consider a reduced number of actions.

There are numerous studies about controlled quantum state transfer using different methods to design the controlling actions. Older studies employ different implementations of Optimal Control Theory (OCT), based on the Krotov method. More recently, many studies exploit the flexibility of Machine Learning methods to produce promising control sequences. OCT is a well-understood framework, with a pleasing physical interpretation. Regrettably, OCT is very demanding in computing time when the number of actuators is large.

Machine learning methods, such as the genetic algorithm or Reinforcement Learning, possess better scalability when dealing with a considerable number of control actions. In particular, deep reinforcement learning applied to the design of control policies in distinct problems has shown great promise. Nevertheless, some issues remain concerning, and further investigation is needed. The architecture of the underlying neural networks, training policy, and usefulness of the trained model are some of the issues that need clarification.

Biologically inspired artificial neural networks have revolutionized the field of machine learning and neuroscience, providing models capable of learning and adapting efficiently. Inspired by the functioning of the animal nervous systems behavior, these networks mimic neuronal connections and processes to tackle complex problems. In the context of quantum physics, where intrinsic nonlinearity and high dimensionality pose significant challenges, biologically inspired artificial neural networks emerge as promising tools for function approximation, model optimization, and predicting quantum behaviors. In this frame, the FANN were successfully used to solve RL related learning situations [3]. We extend that idea using them to solve quantum state transfer in a one-dimensional spin chain problem. Our study focused on two key aspects: the architecture of the neural network and the learning mechanism. Our main objective was to address the problem of understanding the relative capabilities of biological inspiration as far as using it to efficiently solve specific problems unrelated to RL. Throughout this process, we successfully tackled the problem based on accumulated experience using a small network. Our main contributions is confirm the Fiuri neural dynamics model and the FANN (i.e. biologically inspired networks) as a promising and competitive alternative to state of art methods.

2 Background

2.1 Qbits

Here, we present the control scheme used to achieve high-quality quantum state transfer, the mathematical expression of the Hamiltonian, the evolution operator, and the necessary details to formulate the control problem as a learning problem.

The time dependent quantum state, $\psi(t)$, obeys the Schrödinger Equation

$$i\frac{d\psi(t)}{dt} = H\psi(t), \tag{1}$$

where H is the Hamiltonian of the system. For a time-independent Hamiltonian, the solution of Eq. (1) is trivial, $\psi(t) = \exp(iHt)\psi(0) = U(t)\psi(0)$, where U is the evolution operator, $\psi(0)$ is the initial quantum state, and the exponential depends on the spectral decomposition of H. Even for time-dependent Hamiltonians, the solution of Eq. (1) is given by $\psi(t) = \tilde{U}(t)\psi(0)$, but the evolution operator has a more convoluted expression than the simple exponential. It is important to note that since the Hamiltonian is a Hermitian operator, the evolution operator is unitary, so the time evolution preserves the normalization of the vector state $\psi(t)$, which is consistent with the usual probabilistic interpretation associated with it.

When the parameters that define the Hamiltonian depend on time, but in a piecewise fashion, the quantum state at time t is given by

$$\begin{aligned}\psi(t) &= \exp(iH_n\tau_n)\exp(iH_{n-1}\tau_{n-1})\ldots\exp(iH_2\tau_2)\exp(iH_1\tau_1)\psi(0) \\ &= U_nU_{n-1}\ldots U_2U_1\psi(0) = U\psi(0),\end{aligned} \tag{2}$$

where H_k is the appropriate time-independent Hamiltonian valid on the interval τ_k and $t = \sum_{k=1}^{n}\tau_k$.

As is customary in quantum mechanics, the Hamiltonian of a system of qubits is expressed in terms of the Pauli matrices since they form, together with the identity matrix, a basis set for all the Hermitian matrices operating on $\mathbb{C}^2$. So, a rather general Hamiltonian for a chain of N qubits valid at the τ_j interval is given by

$$H_j = \sum_{i=1}^{N-1}(\sigma_i^x\sigma_{i+1}^x + \sigma_i^y\sigma_{i+1}^y) + \sum_{k=1}^{N}B_{k,j}\sigma_k^z = H_{XX} + \sum_{k=1}^{N}B_{k,j}\sigma_k^z, \tag{3}$$

where $B_{k,j}$ is the control applied over the k-th qubit, σ_i^x, σ_i^y, and σ_i^z, are the Pauli matrices associated to the i-th qubit. In many-body physics, the Hamiltonian in Eq. (3) is called the XX Hamiltonian with site-dependent transversal magnetic field. We do not assume a specific physical setting associated with the Hamiltonian, but we assume that the system has tunable Z one-qubit gates, using the common terminology in quantum computation. The Hamiltonian in Eq. (3) operates in the space $\mathbb{C}^{2^N} = \bigotimes_{k=1}^{N} \mathbb{C}^2$. Nevertheless, since we aim to study the transmission of arbitrary one-qubit quantum states, we use the Hamiltonian matrix that results from reducing the Hamiltonian to the one-excitation subspace, spanned by the computational basis $|\mathbf{i}\rangle = |00\ldots1\ldots\rangle$, for $i = 1, 2, \ldots, N$, where the ket $|\mathbf{i}\rangle$ denotes a vector with N components, with only one component different from zero, equal to the unity located in the i-th component. Using this notation, it is straightforward to show that the Hamiltonian matrix on the one-excitation subspace is a tridiagonal one. The elements on the diagonal are the magnetic fields. All the other non-null elements are equal to unity.

We will use as the **actions** driving the evolution of the qubit chain the control "fields" $B_{l,k}$, with the proviso that for each time interval only one of the fields will be non-zero.

The average fidelity of transmission quantifies the quality of the transfer protocol and is given by $f_{av} = \frac{1}{2} + \frac{1}{3}|P| + \frac{1}{6}|P|^2$, where the probability of transmission is

$$|P|^2 = |\langle \mathbf{N}|U(t)|\mathbf{1}\rangle|^2 . \tag{4}$$

Numerous methods allow for selecting sequences of fields $B_{j,k}$ which result in values of P close to unity. We aim to produce sequences compatible with fast transmission. For the units chosen for the Hamiltonian in Eq. 3, the sequences and methods employed produce arrival times T such that $N/2 \leq T \leq N$. For the arrival time, we understand the time that is smaller than N at which the transmission probability attains its maximum value. The number of time intervals τ_k is a hyperparameter of the particular method used.

2.2 Reinforcement learning

Reinforcement Learning (**RL**) is an automatic learning technique in which an agent learns to perform a task through a succession of trial-and-error interactions with a changing environment, usually modeled as a Markov decision process. To train the agent in solving a problem, the environment provides us with a reward and a detailed description of the problem. Every time we execute an action on the environment, we obtain a reward and a descriptor of the State again. A desired solution maximizes the rewards obtained in a given number of interactions [12]. An episode refers to the sequence of interactions with the environment that begins from its initial state. In each episode, the rewards obtained in each interaction accumulate to calculate a total reward for that episode.

To solve the problem of **QBits** using **RL**, it was necessary to model and implement an environment responsible for providing some type of state description (**state**), a set of actions that can be applied to it ($\mathcal{A}$), a reward corresponding to the action executed ($\mathbf{r}$) and a sign of completion (to know when the episode concludes). We define four different reward functions with the aim of exploring better solutions.

In this work, we address the problem of **QBits** through two **RL** general approaches: the first consists of tabular methods, and the second uses an extended **FANN** methodology presented in [3].

Tabular **RL Q-learning** is one of the most popular reinforcement learning algorithms used in automatic learning and artificial intelligence. Allows an agent to learn the optimal actions that must be taken in a given environment to maximize accumulated rewards over time. A key concept in **Q-Learning** is that of the values Q (Value-action function), responsible for storing estimates of the expected accumulated reward that the agent can obtain when executing a specific action in a given state. These values are then used to generate a policy of selection of actions, as presented in [12]: $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$ where: s and s' are the current and next states, a and a' are actions, r is the received reward, α is the learning rate and γ is the discount factor.

The agent chooses actions based on an exploration strategy, such as *Epsilon-Greedy*, to balance the exploration of new actions and the exploitation of actions already known as good. Over time, the values converge towards optimal value-action, guiding the agent to make optimal decisions. In this work, we refer to this model as **QN**.

Double Q-Learning is an extension of the standard **Q-Learning** algorithm, designed to address overestimation bias that can occur in traditional **Q-Learning**. This bias occurs because the maximum operator in the **Q-Learning** update tends to overestimate the value of the actions, especially when the values are low or approximate. **Double Q-learning** maintains two separate estimates of Q values instead of one. These are commonly denoted as Q_1 and Q_2.

The update of the Q values, after executing one action and observing the reward and the following state at each step of the episode, is carried out by randomly selecting one of the two functions Q (for example, q_1 or q_2) and using the other to determine the optimal action in the following state. The update rule for the selected Q function is as follows: $Q_i(s,a) \leftarrow Q_i(s,a) + \alpha\left[r + \gamma Q_j\left(s', \arg\max_{a'} Q_i(s',a')\right) - Q_i(s,a)\right]$ where: i and j are 1 or 2, with $i \neq j$. The idea is that Q_i is updated using the estimate from Q_j for the next state's maximum action, reducing overestimation bias. Like **QN**, the Q-values converge to the optimal action-values, guiding the agent to make optimal decisions. In this work, we refer to this model as **DoubleQN**.

Monte Carlo is the last tabular method used in this work (called us as **MC**). It is a direct approach to reinforcement learning, used to estimate the value of states or the state-action pairs from the average of complete episodes. It is especially useful in environments with spaces of finite and discrete actions, where a table can store these estimates. The agent follows a policy to generate episodes of experience. Then, for each state visited (or status-action), the return G_T is calculated as the sum of discounted rewards from that moment until the episode ends: $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots$

The value estimate for each state (or state-action pair) is updated by averaging all observed returns: $V(s) \leftarrow$ average of all G_t where s was visited.

2.3 FANN

Fiuri Artificial Neural Networks (**FANN**) are biologically inspired artificial neural networks as presented in [3], but in this work, we extended that method to some kind of fully connected architecture, which allows us to have multiple intermediate layers. We added a new connection behavior, trying to simulate the modulation mechanism in the neuron information transmission (that version of the model includes an **M** in its name). A **FANN** is a compound of neurons with a discrete dynamic, which is in charge of describing its behavior. The inter-neuronal connections also have a well-defined behavior depending on their type (excitatory, inhibitory, or electrical). That dynamics (shown in table 1) describes how the neuron reacts when stimuli are coming from its dendritic connections.

Table 1. Neural Dynamics. In ***Fiuri***, E_n and O_n represent the input and output states of neuron n, respectively, once it was influenced. If S_n represents the stimulus coming through the dendritic connections ($j's$) due to the currents I_{in}^{j} (7), then the expressions (5) and (6) define the dynamics of n. T_n and d_n are neuronal parameters that have to be learned and represent the firing threshold and the decay factor (due to not enough stimulus). The currents I_{in}^{i} are defined by (8) for each connection type. ω_j is the connection capacitance.

$$O_n = \begin{cases} S_n - T_n & \text{if } S_n > T_n \\ 0 & \text{other case} \end{cases} \tag{5}$$

$$E_n = \begin{cases} S_n - T_n & \text{if } S_n > T_n \\ E_n - d_n & \text{if } S_n \leq T_n \text{ and } S_n = E_n \\ S_n & \text{other case} \end{cases} \tag{6}$$

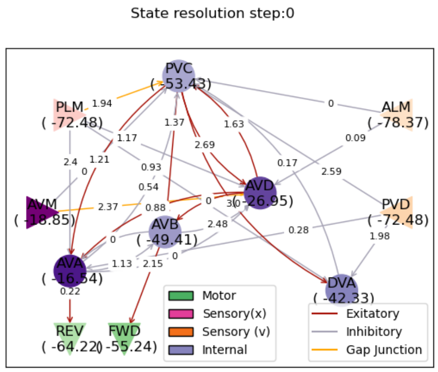
$$S_n = E_n + \sum_{j=1}^{m} I_{in}^{j} \tag{7}$$

$$I_{in}^{j} = \begin{cases} \omega_j * O_j & \text{if } O_j \geq E_n \text{ y gap junct.} \\ -\omega_j * O_j & \text{if } O_j < E_n \text{ y gap junct.} \\ \omega_j * O_j & \text{chemical excitatory} \\ -\omega_j * O_j & \text{chemical inhibitory} \end{cases} \tag{8}$$

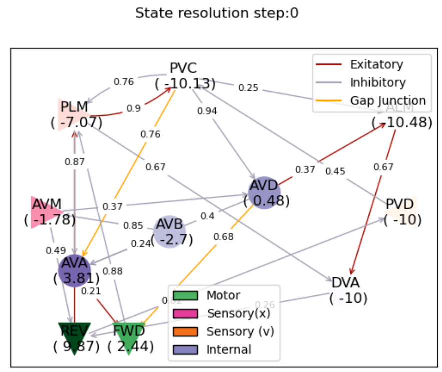
We use the well-studied **Tap Withdrawal Circuit** (**TWC**) architecture that models a small portion of the *C-elegance* connectome [13, 4]. This small network consists of eleven neurons, with four sensory, two motor, and five internal neurons. We also use a variant of that circuit named **Shuffle Architecture** (**SA**) obtained from the same neurons as **TWC** but with all its connections randomly shuffled. The table 2 shows a graphical view for both of them.

Then we use the **TWC** as [3] in its two versions (including Shuffle Architecture **SATWC**) but also our extension to fully connected architecture named **FC** (and **SAFC** for the shuffle architecture version). We can think of **FC** as layers where we can stimulate the neurons belonging to the input layer. Each neuron of this layer is connected to all neurons belonging to the first internal layer. Each internal layer has a neuron connected to the next internal layer, and so on till the last internal layer is connected in the same way to the output layer. The left image in table 3 shows this; notice is similar to deep learning fully connected networks.

Table 2. Architecture examples: Numbers on neurons are the electrical voltage at the moment the picture was taken, and numbers on connections are their capacitance.

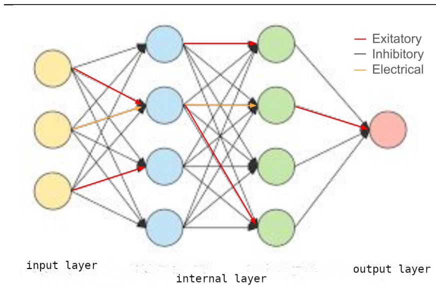


Tap Withdrawal Circuit Architecture.

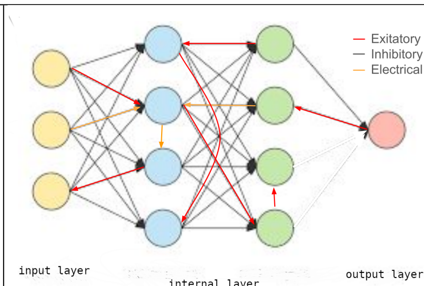


Example of **Shuffle Architecture**.

Table 3. Architecture examples. Notice SAFC allows loops.



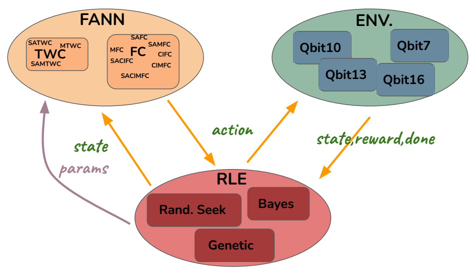
FC Architecture.



Example of **SAFC**.

SA can break this order due to the connections being randomly shuffled, even allowing loops as shown in the right figure. That kind of network maintains the bio inspiration due to use the Fiuri neuron dynamic model, different connection types, the proportion of connections for each type (inhibitory, excitatory, and electrical), and the new modulation mechanism. The figure 1 attempts to help with a schematic view.

Fig. 1. Schematic view.



All these architectures learn due to the optimization procedure on the ANN internal parameters, so we also maintain the three mechanisms used in [3] (Random Seek, Genetic, and Bayesian) RL in this method simply acts as a "circuit evaluator" due to only accumulating the obtained step rewards on each episode, storing the ANN that has the best total reward.

3 Related works

References [8, 9, 2] studied controlled quantum state transfer using optimal control theory, although in different settings. Later studies found that quantum state transfer occurs at times close to the *quantum speed limit*, but early studies found arrival times orders of magnitude larger. Choosing a simple set of actions, Zhang et al. [14] achieved fast transfer times, with a good to poor transmission probability in qubit chains with short to moderate lengths. These authors used Deep Reinforcement Learning with an ϵ-greedy policy. Preliminary results [10, 11] convinced us that exploring other RL strategies or different neural network settings was worthwhile. Improving the results of Zhang et al. using the same approach is indeed quite laborious. In [3], the methods employed in the present work were presented and developed thoroughly.

4 Our approach

4.1 Exploring the solution space on FANNs

As mentioned before, traversing the space of potential solutions (trying different values for internal dynamic parameters) is crucial when training our **FANN** model. In this paper, we employ three well-known methods for exploring the parameter search space: **Random Seek** (**RS**), **Genetic Algorithm** (**GA**) [7], and **Bayesian Optimization** (**BO**) [6, 5]. In all cases, we maintained the same hyperparameter optimization as in [3] remained fixed throughout the experiments.

4.2 Environmet

A critical piece of any RL system is the environment that is in charge of modeling the problem we want to solve. We implemented this piece of code following the problem characteristics introduced before. We considered four versions of the same problem, named **Qbit7**, **Qbit10**, **Qbit13**, and **Qbit16**, which correspond to chain lengths of 7, 10, 13, and 16, respectively. Our environment has as many actions as the chain length (i.e. $\mathcal{A}$), so when we want to solve **Qbit7**, we will need to take into account that we have 7 actions, but if we want to solve **Qbit13**, we then have 13 actions. We assigned numbers $1..n$ to the actions and implemented them in such a way that the effect of executing the action n on the environment is to influence the chain link n.

Another key component is the *reward* the environment provides on each action execution. We define an ideal goal fidelity to be reached on every problem, for **Qbit7** is 0.99, for **Qbit10** is 0.98 for **Qbit13** is 0.95, and for **Qbit16** 0.92. Notice that when the chain length increases, fidelity degradation is expected. We proposed four different reward functions: **site evolution**, **transmission evolution**, **last action auto-complete** and **random action auto-complete**, here is a detail:

Site evolution(SE): aims to measure the position of the excited (up) state along the chain. To do that, we define the mean site as: $\mu = \sum \alpha_i i^2$ where α_i are the coefficients of the system's full quantum state $\psi(t) = \sum_i \alpha_i i$. Since transmission probability does not start to grow until several time steps have gone by, this reward tries to encourage actions that make the excited state move forward along the chain, even in earlier stages where it is still far from the final qubit. The formula also includes the transmission probability, resulting in: $r(t) = (1 + P)\mu$.

Transmission evolution(TE): tries to induce the information wave moving to the last chain link. During each action execution, it maintains the current wave position (i.e., the specific chain link) and its amplitude. It rewards when the wave picks change one chain link forward to the final, or if no position change occurs, then it at least increases its amplitude. It penalizes in other cases. It returns on each step 10 if it was a good action, 50000 if it exceeded the objective, and 0.1 in other cases.

Last action auto-complete(LA): calculates the whole episode action list fidelity, but starting from the current state and auto-completing till the final with the last executed action. To be more clear, suppose that we are trying to solve **Qbit7**, so the episode is done when we execute 35 actions. In that example, suppose we want to execute action n and we have already run m actions on the environment, which is in the state e. This reward function will complete the episode (from state e), executing always n till reaches the *done* signal ($35 - m$ times). This reward function returns the value defined by (9) where o_n is the n-sized chain objective and fid the fidelity.

Random action auto-complete(RA): is like last action auto-complete, but in every step, the action used to complete the action list is random. This reward function also returns the value defined by (9) where o_n is the n-sized chain objective and fid the fidelity.

$$rew = \begin{cases} fid & \text{if } fid \leq 0.8 \\ 100/(1 + e^{(10*(o_{chain} - fid))}) & \text{if } 0.8 \leq fid \leq o_n \\ 50000 + fid & \text{if } o_n \leq fid \end{cases} \tag{9}$$

Another important piece is the state description that the environment provides us. We use as a state descriptor a complex *n_uple* representing the whole quantum chain state for a chain of size n. By last, we use a *done* signal equivalent to 5 times the chain length, which seems to be good enough.

4.3 Interfaces

The interface in that work concerned with the way the environment state is passed to the learning mechanism and how the learning mechanism's output is interpreted as an action. Our first approach for **FC FANN**s was to feed two input neurons with every chain link state; we passed the real part to the first and the imaginary part to the second one. So to train a **FC** trying to solve a chain with length n, we needed $2n$ input neurons. It was quickly noticed that passing both components to **FANN** was not necessary, and so we tried to pass only the real part of the state (in the same way, we attempted to pass only the imaginary part, with similar results). Therefore, for every chain link, we only have one input neuron. In this way, the results were very similar to the first approach, but the train times were reduced drastically due to we needed n input neurons for the **FC**. A third approach we also tried was to have only 2 input neurons and feed the first with the wave peak amplitude and the second with the peak position (in terms of the chain link). A model with this kind of interface has a **CI** (Condensed Information) word in its name. This last approach reduced the training times once again due to only having a few internal parameters in comparison with others. For **TWC**, we can not select the number of input neurons, so we took the **CI** approach, and the same for tabular methods. Concerning the output neurons, all our **FC** networks have only one output neuron, which is directly matched with an action after a discretization step. In **TWC** nets, we have 2 output neurons that have to compete, and the result is discretized and mapped to an action.

4.4 Modulation mechanism on FANNs

We added a modulation mechanism to the connection dynamics, just with the utterance of mimicking the biological modulation behavior in dendritic connections. Without that mechanism, the dendritic connection influence was defined by (8), so we present now how the neuron dynamics are influenced. The expression (10) shows that the neuron suffered influence when we use the modulation mechanism. Notice a new parameter σ is introduced there, this parameter needs to be learned so models with that mechanism (i.e, with **M** appearing in the name) have more parameters than others. This modulation is based on a sigmoid function and does not affect the electrical connections, which is also a biological inspiration [1].

$$I_{in}^{j} = \begin{cases} \omega_j * O_j & \text{if } O_j \geq E_n \text{ y gap junct.} \\ -\omega_j * O_j & \text{if } O_j < E_n \text{ y gap junct.} \\ \omega_j * O_j/(1+e^{-\sigma * O_j}) & \text{chemical excitatory} \\ -\omega_j * O_j/(1+e^{-\sigma * O_j}) & \text{chemical inhibitory} \end{cases} \tag{10}$$

Table 4. Different architectures for the explored FANNs

Architecture	Description
TWC	Clasical TWC
SATWC	Shuffle Architecture of TWC
MTWC	Clasical TWC with Modulation
MSATWC	Shuffle Architecture of TWC with Modulation
FC	Fully Conected FANN
MFC	Fully Conected FANN with Modulation
CIFC	Fully Conected FANN with Condensed Information
CIMFC	Fully Conected FANN with Condensed Information and Modulation
SAFC	Shuffle Architecture of FC FANN
SAMFC	Shuffle Architecture of FC FANN with Modulation
SACIFC	Shuffle Architecture of FC FANN with Condensed Information
SACIMFC	Shuffle Architecture of FC FANN with Condensed Information and Modulation

4.5 FANN's Architecture Notes

We need to mention some considerations here. In the first place, we tried to maintain the rate between input and other neurons between 0.57 and 0.7, the first because it is observed in **TWC** and the second risen from experience. In a second place and the same sense, we tried to maintain the rate of connections by type concerning **TWC**, defined 55% of connections should be inhibitory, 35% should be excitatory, and approximately. 10% electrical. All these rates are rounded as needed, so they are not exact numbers. With all those considerations presented till moment, the table 4 resumes the different architectures we explored, together with their names, and a brief description. Notice that the size architecture depends on the chain length in some configurations. Table 5 shows detailed info about configuration characteristics and their dynamical parameters.

Table 5. FANN's dynamical configuration. For each architecture number of neurons and connection are shown grouped by type. *np* and *cp* refer to total neuron and connection parameters, respectively.

	neurons				connections				dyn param		
	in	out	intern.	total	inhib.	excit.	elctr.	total	np	cp	total
No length dependant											
TWC,	4	2	5	11	15	9	2	26	22	26	48
MTWC	4	2	5	11	15	9	2	26	22	52	74
CIFC	2	1	7	10	12	7	2	21	20	21	41
MCIFC	2	1	7	10	12	7	2	21	20	42	62
Length dependant											
Qbit7											
FC	7	1	9	17	40	25	7	72	34	72	106
MFC	7	1	9	17	40	25	7	72	34	144	178
Qbit10											
FC	10	1	15	26	91	58	16	165	52	165	217
MFC	10	1	15	26	91	58	16	165	52	330	382
Qbit13											
FC	13	1	21	35	162	103	29	294	70	294	364
MFC	13	1	21	35	162	103	29	294	70	588	658
Qbit16											
FC	16	1	27	44	252	161	46	459	88	459	547
MFC	16	1	27	44	252	161	46	459	88	918	1006

4.6 Tabular Methods

Some special considerations are required when using tabular methods. In such techniques, there are no input neurons to receive the chain state directly. Instead, all possible state combinations must be represented in a table (the Q-values table), which stores the expected accumulated reward for each state-action pair. This table is then used to guide the action selection policy.
Consequently, we need to store a value for each pair (s, a), where s is the environment state and a is the action to be executed. The set of actions is known to be finite; specifically, we consider n actions, where n is the length of the chain.

As mentioned earlier, we adopt the **CI** approach for tabular methods. In this representation, the state is defined as a pair (p, a), where $p \in \mathbb{I}$ is an integer denoting the position of the wave peak (with $0 \leq p \leq n-1$), and $a \in \mathbb{R}$ is a real number representing the wave peak amplitude.

Since a is a continuous value, it cannot be used directly in a tabular representation. Therefore, we discretize the amplitude to define the environment states as elements of the Cartesian product $\mathbb{P} \times \mathbb{A}$, where $\mathbb{P}$ is the set of possible positions and $\mathbb{A}$ is the set of discretized amplitude levels. We chose to define 15 discrete amplitude levels and map the actual amplitude from the environment to the nearest among these 15 levels.

To clarify, if we are solving the problem for a chain of length n, the resulting Q-value table will have a size of $n \times 15 \times n$.

4.7 Experimental Setup

In this section, we present the experiments conducted to evaluate the performance of various combinations applied to solve the four problem instances defined as **Qbit7**, **Qbit10**, **Qbit13**, and **Qbit16**. A total of 3,600 models were trained, corresponding to 900 models per problem instance.

For each problem, we explored the four reward versions introduced earlier. Thus, for each combination of problem *Qbitn* and reward function r, we trained 225 models.

This number results from the following configuration space: we trained 12 neural architectures (listed in Table 4), each with one of three optimization algorithms—**RS** (Random Search), **GA** (Genetic Algorithm), and **BO** (Bayesian Optimization)—resulting in 36 different **FANN** configurations. In addition, we trained three tabular methods, each with three different step sizes, adding 9 tabular configurations. This gives a total of 45 configurations per $(Qbitn, r)$ pair.

Each configuration was trained and evaluated five times to account for variability, resulting in $45 \times 5 = 225$ models per pair. Therefore, across the 4 problem instances and 4 reward functions, a total of $4 \times 4 \times 225 = 3,600$ models were trained and analyzed.

FANN models incorporate an early detection mechanism (Early Stopping), so in all cases, a large number of optimization steps was set. However, the specific number of steps varies according to the optimization algorithm used: for **RS**, 500,000 steps were used; for **GA**, 5,000; and for **BO**, 10,000. All **FANN** models were trained using the number of steps corresponding to the optimizer used.

Instead, tabular methods do not have an early detection mechanism. For this reason, three scenarios per configuration were evaluated, using 100,000, 200,000, and 400,000 episodes, respectively. In this case, we talk about episodes and not optimization steps.

Once a model was trained, it underwent an evaluation phase. This consisted of attempting to solve the problem again using the trained model, starting from the initial state and measuring the final transmission fidelity. The resulting fidelity is reported as the performance metric in this work. Since the problem is deterministic, a single evaluation run is sufficient.

5 Results

In this section, we report the results obtained and compare them with related works. The results, presented in Tables 6 and 7, show that our study not only stands out in terms of absolute fidelity and stability, but also exceeds the scope of previous work in the literature, particularly that of Zhang et al. [14].

Experimental results show that multiple control strategies achieve high fidelities for the quantum transmission task. In particular, the tabular method **Monte Carlo (MC)**, in most cases, was the most efficient in terms of average fidelity across different chain lengths (see Table 7). These approaches achieve results above 0.93 even on chains of up to 16 qubits, and they do so without requiring complex neural structures, as they store estimates per state or action.

Table 6. Experimental results disaggregated by reward type, some models have in their names the string **RS**, **GA** or **BO** only to remark the optimization method used to reach that solution.

Reward	Fid	Model	Reward	Fid	Model
	Qbit7			Qbit10	
$LastAuto.$	0.993	MC	$LastAuto.$	0.987	MC
$RandomAuto.$	0.975	$CIFC_{GA}$	$RandomAuto.$	0.951	$MSATWC_{BO}$
$SiteEvol.$	0.994	MFC_{RS}	$SiteEvol.$	0.913	$SAMFC_{GA}$
$Transm.Evol.$	0.993	$DoubleQN$	$Transm.Evol.$	0.876	$SAMFC_{BO}$
	Qbit13			Qbit16	
$LastAuto.$	0.967	MC	$LastAuto.$	0.937	MC
$RandomAuto.$	0.928	$CIMFC_{GA}$	$RandomAuto.$	0.892	MFC_{BO}
$SiteEvol.$	0.846	$SACIMFC_{BO}$	$SiteEvol.$	0.854	$SAFC_{GA}$
$Transm.Evol.$	0.828	TWC_{GA}	$Transm.Evol.$	0.925	FC_{RS}

However, the advantage of tabular methods is not overwhelming: several neural network-based models achieved similar fidelities. For example, the fully connected architecture (**FC**) and its deep modular version (**MFC**) also achieved high reliability (more than 0.95). However, the complexity of these networks scales significantly with the number of qubits. According to Table 5, for 16 qubits, an **MFC** network exceeds 900 trainable parameters, which implies a substantially higher computational cost.

Table 7. Experimental results disaggregated by model type, some models have in their names the string **RS**, **GA**, or **BO** only to remark the optimization method used to reach that solution. Every problem is highlighted with dark gray the best obtained value($\mathcal{B}$), with gray a value whose distance $\mathcal{D}$ to $\mathcal{B}$ is less than $\mathcal{B} * 0.01$, and with light gray the values with $\mathcal{B} * 0.01 \leq \mathcal{D} \leq \mathcal{B} * 0.05$. To allow some additional analysis, we also show in the *ip* column the number of internal parameters in the case of FANNs and results obtained running the code coming from Zhang et al. [14].

Archit	Fid	Model	ip	Archit	Fid	Model	ip
Qbit7				Qbit10			
TWC	0.978	TWC_{GA}	48	TWC	0.936	TWC_{BO}	48
$SATWC$	0.963	$SATWC_{RS}$	48	$SATWC$	0.925	$SATWC_{GA}$	48
$MTWC$	0.964	$MTWC_{BO}$	74	$MTWC$	0.931	$MTWC_{RS}$	74
$MSATWC$	0.973	$MSATWC_{RS}$	74	$MSATWC$	0.951	$MSATWC_{BO}$	74
FC	0.98	FC_{GA}	106	FC	0.949	FC_{GA}	106
MFC	0.994	MFC_{RS}	178	MFC	0.96	MFC_{GA}	178
$CIFC$	0.975	$CIFC_{GA}$	41	$CIFC$	0.946	$CIFC_{GA}$	41
$CIMFC$	0.973	$CIMFC_{BO}$	62	$CIMFC$	0.913	$CIMFC_{BO}$	62
$SAFC$	0.981	$SAFC_{BO}$	106	$SAFC$	0.96	$SAFC_{GA}$	106
$SAMFC$	0.965	$SAMFC_{GA}$	178	$SAMFC$	0.953	$SAMFC_{GA}$	178
$SACIFC$	0.954	$SACIFC_{BO}$	41	$SACIFC$	0.897	$SACIFC_{GA}$	41
$SACIMFC$	0.919	$SACIMF_{BO}$	62	$SACIMFC$	0.89	$SACIMFC_{BO}$	62
QN	0.992	QN	–	QN	0.829	QN	–
$DoubleQN$	0.993	$DoubleQN$	–	$DoubleQN$	0.946	$DoubleQN$	–
MC	0.993	MC	–	MC	0.987	MC	–
$Zhang$	0.999	–	7296	$Zhang$	0.977	–	8016
Qbit13				Qbit16			
TWC	0.909	TWC_{BO}	48	TWC	0.907	TWC_{BO}	48
$SATWC$	0.895	$SATWC_{GA}$	48	$SATWC$	0.883	$SATWC_{BO}$	48
$MTWC$	0.885	$MTWC_{BO}$	74	$MTWC$	0.901	$MTWC_{RS}$	74
$MSATWC$	0.91	$MSATWC_{GA}$	74	$MSATWC$	0.872	$MSATWC_{BO}$	74
FC	0.939	FC_{GA}	106	FC	0.93	FC_{RS}	106
MFC	0.917	MFC_{GA}	178	MFC	0.892	MFC_{BO}	178
$CIFC$	0.919	$CIFC_{GA}$	41	$CIFC$	0.921	$CIFC_{BO}$	41
$CIMFC$	0.928	$CIMFC_{GA}$	62	$CIMFC$	0.849	$CIMFC_{BO}$	62
$SAFC$	0.945	$SAFC_{GA}$	106	$SAFC$	0.909	$SAFC_{GA}$	106
$SAMFC$	0.916	$SAMFC_{GA}$	178	$SAMFC$	0.883	$SAMFC_{BO}$	178
$SACIFC$	0.908	$SACIFC_{BO}$	41	$SACIFC$	0.85	$SACIFC_{RS}$	41
$SACIMFC$	0.884	$SACIMFC_{BO}$	62	$SACIMFC$	0.814	$SACIMFC_{BO}$	62
QN	0.739	QN	–	QN	0.889	QN	–
$DoubleQN$	0.922	$DoubleQN$	–	$DoubleQN$	0.918	$DoubleQN$	–
MC	0.967	MC	–	MC	0.937	MC	–
$Zhang$	0.966	–	8736	$Zhang$	0.943	–	9456

In this context, bio-inspired architectures stand out, especially the **TWC**, which achieves very competitive fidelities (between 0.90 and 0.95) with just 11 neurons and about 40 estimable parameters. These architectures do not depend on chain length and maintain their size constant even when scaling the system, as seen in the upper part of Table 5. This makes them ideal candidates for scenarios where structural robustness and implementation efficiency are required.

Compared with previous work, our approach has clear advantages. We successfully reproduced the results of Zhang et al.'s work in the evaluation phase, achieving similar fidelities on chains of up to 11 qubits. However, there are fundamental differences in the methodological approach. Zhang et al.'s work does not seek to train a generalizing **RL** agent, but rather to optimize a policy on a validation set under ideal conditions. Their implementation used fully connected networks of more than 200 neurons, exceeding 7,000 trainable parameters. In contrast, our models achieve similar or higher fidelities using two to three orders of magnitude fewer parameters.

6 Conclusion and future work

This study demonstrates that it is possible to design quantum control agents that are efficient in both fidelity and structural complexity. Tabular methods such as **MC** and **Double Q-learning** emerged as the most effective, although architectures such as **TWC** and **FC** achieved comparable fidelities, with varying degrees of computational cost. In particular, **TWC** achieved fidelities above 0.9 using only a fraction of the parameters required by traditional fully connected networks.

This result validates the hypothesis that performance does not depend exclusively on the number of parameters, but on the agent's structural design.

Bio-inspired architectures, by incorporating functional motifs such as synaptic modulation and distinct neuronal types (inhibitory, excitatory, electrical), were able to maintain high performance with fixed, compact, and scalable topologies.

This represents a promising alternative to dense neural networks, especially for applications on limited hardware or real quantum systems.

Unlike the work of Zhang et al. [14], whose main objective was to validate effective protocols without training generalizable agents, our approach aims to develop truly autonomous agents. Although in this work the agent is deterministic (due to the characteristics of the environment), the future objective is to extend training to non-deterministic environments. To this end, we propose:

- **Incorporate structured and random noise during training** to robustify the learned policy against realistic perturbations and errors.
- **Develop adaptive agents** that do not require ex post selection of the best policy, but rather converge through experience and reinforcement.
- **Explore novel bio-inspired dynamics**, such as rhythmic oscillations or recurrent couplings, which could improve coordination in information transfer along chains of qubits.
- **Evaluate the combination of strategies**, integrating tabular methods with compact neural modules, to take advantage of the best of both worlds.

Taken together, these results suggest that the use of neurobiologically inspired functional structures, combined with robust reinforcement learning techniques, can lead to the development of scalable, efficient, and adaptable quantum controllers suitable for applications in experimental physics and emerging technologies.

References

1. Laurence F Abbott and Peter Dayan. Theoretical neuroscience, December 2021. Accessed: 2025-7-30.
2. D.S. Acosta Coden, S.S. Gómez, A. Ferrón, and O. Osenda. Controlled quantum state transfer in xx spin chains at the quantum speed limit. *Physics Letters A*, 387:127009, 2021.
3. Fiuri Ariel M., Dominguez Martin A., and Francisco Tamarit. Reinforcement learning and biologically inspired artificial neural networks. In *Information Management and Big Data*, pages 62–79, Cham, 2024. Springer Nature Switzerland.
4. D. Hall B. Chen and D. Chklovskii. Wiring optimization can relate neuronal structure and function. *Proceedings of the National Academy of Sciences*, 103(12):4723–4728, 2006.
5. J. Bergstra, D. Yamins, and D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 115–123, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.
6. Eric Brochu, Vlad M. Cora, and Nando de Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning, 2010.
7. Ahmed Fawzy Gad. Pygad: An intuitive genetic algorithm python library, 2023.
8. Jiangbin Gong and Paul Brumer. Controlled quantum-state transfer in a spin chain. *Phys. Rev. A*, 75:032331, Mar 2007.
9. Michael Murphy, Simone Montangero, Vittorio Giovannetti, and Tommaso Calarco. Communication at the quantum speed limit along a spin chain. *Phys. Rev. A*, 82:022318, Aug 2010.
10. Sofía Perón Santana. Control dinámico para transmisión de estados cuánticos, 2025.
11. Sofía Perón Santana, Ariel Fiuri, Martín Domínguez, and Omar Osenda. The pros and cons of using deep reinforced learning or genetic algorithms to design control schemes for quantum state transfer on qubit chains. *in preparation*, 2025.
12. R.S. Sutton and A.G. Barto. Reinforcement learning: An introduction. *IEEE Transactions on Neural Networks*, 9(5):1054–1054, 1998.
13. J G White, E Southgate, J N Thomson, and S Brenner. The structure of the nervous system of the nematodecaenorhabditis elegans. *Philos. Trans. R. Soc. Lond.*, 314(1165):1–340, November 1986.
14. Xiao-Ming Zhang, Zi-Wei Cui, Xin Wang, and Man-Hong Yung. Automatic spin-chain learning to explore the quantum speed limit. *Phys. Rev. A*, 97:052333, 2018.