npj | Quantum Information

www.nature.com/npjqi

ARTICLE OPEN

An artificial neuron implemented on an actual quantum processor

Francesco Tacchino[1], Chiara Macchiavello[1,2,3], Dario Gerace[1] and Daniele Bajoni[4]

Artificial neural networks are the heart of machine learning algorithms and artificial intelligence. Historically, the simplest implementation of an artificial neuron traces back to the classical Rosenblatt's "perceptron", but its long term practical applications may be hindered by the fast scaling up of computational complexity, especially relevant for the training of multilayered perceptron networks. Here we introduce a quantum information-based algorithm implementing the quantum computer version of a binary-valued perceptron, which shows exponential advantage in storage resources over alternative realizations. We experimentally test a few qubits version of this model on an actual small-scale quantum processor, which gives answers consistent with the expected results. We show that this quantum model of a perceptron can be trained in a hybrid quantum-classical scheme employing a modified version of the perceptron update rule and used as an elementary nonlinear classifier of simple patterns, as a first step towards practical quantum neural networks efficiently implemented on near-term quantum processing hardware.

npj Quantum Information (2019)5:26 ; https://doi.org/10.1038/s41534-019-0140-4

INTRODUCTION

Artificial neural networks are a class of computational models that have proven to be highly successful at specific tasks like pattern recognition, image classification, and decision making.[1] They are essentially made of a set of nodes, or neurons, and the corresponding set of mutual connections, whose architecture is naturally inspired by neural nodes and synaptic connections in biological systems.[1,2] In practical applications, artificial neural networks are mostly run as classical algorithms on conventional computers, but considerable interest has also been devoted to *physical* neural networks, i.e., neural networks implemented on dedicated hardware.[3–6]

Among the possible computing platforms, prospective quantum computers seem particularly well suited for implementing artificial neural networks.[7] In fact, the intrinsic property of Quantum Mechanics of representing and storing large complex valued vectors and matrices, as well as performing linear operations on such vectors, is believed to result in an exponential increase either in memory storage or processing power for neural networks directly implemented on quantum processors.[8–18] The simplest model of an artificial neuron, the so called "perceptron", was originally proposed by R. Rosenblatt in 1957,[19] and is schematically outlined in Fig. 1a. A real-valued vector, $\vec{i}$, of dimension m represents the input, and it is combined with a real-valued weight vector, $\vec{w}$. The perceptron output is evaluated as a binary response function resulting from the inner product of the two vectors, with a threshold value deciding for the "yes/no" response. In the lowest level implementations, $\vec{i}$ and $\vec{w}$ are binary valued vectors themselves, as proposed by McCulloch and Pitts in 1943 as a simple model of a neuron.[2,20]

Perceptrons and McCulloch-Pitts neurons are limited in the operations that they can perform, but they are still at the basis of machine learning algorithms in more complex artificial neural networks made of multilayered perceptron architectures. However, the computational complexity increases with increasing number of nodes and interlayer connectivity and it is not clear whether this could eventually call for a change in paradigm, although different strategies can be put forward to optimize the efficiency of classical algorithms.[21] In this respect, several proposals have been advanced in recent years to implement perceptrons on quantum computers. The most largely investigated concept is that of a "qubit neuron", in which each qubit (the computational unit in quantum computers) acts as an individual neuron within the network. Most of the research effort has been devoted to exploit the nonlinearity of the measurement process in order to implement the threshold function.[9]

Here we introduce an alternative design that closely mimics a binary valued Rosenblatt perceptron on a quantum computer. First, the equivalent of m-dimensional classical input and weight vectors is encoded on the quantum hardware by using N qubits, where $m = 2^N$. On one hand, this evidently allows to exploit the exponential advantage of quantum information storage, as already pointed out.[10,13] On the other, we implement an original procedure to generate multipartite entangled states based on quantum information principles[22] that allows to crucially scale down the quantum computational resources to be employed. We experimentally show the effectiveness of such an approach by practically implementing a 2 qubits version of the algorithm on the IBM quantum processor available for cloud quantum computing. In this respect, the present work constitutes a key step towards the efficient use of prospective quantum processing devices for machine learning applications. In this simple case, we show that the quantum perceptron model can be used to sort out simple patterns, such as vertical or horizontal lines among all possible inputs. In order to show the potential of our proposed implementation of a quantum artificial neuron, we theoretically

[1]Dipartimento di Fisica, Università di Pavia, via Bassi 6, I-27100 Pavia, Italy; [2]INFN Sezione di Pavia, via Bassi 6, I-27100 Pavia, Italy; [3]CNR-INO, largo E. Fermi 6, I-50125 Firenze, Italy and [4]Dipartimento di Ingegneria Industriale e dell'Informazione, Università di Pavia, via Ferrata 1, I-27100 Pavia, Italy
Correspondence: Francesco Tacchino (francesco.tacchino01@ateneopv.it)

Received: 21 November 2018 Accepted: 1 March 2019
Published online: 29 March 2019

Published in partnership with The University of New South Wales

npj nature partner journals

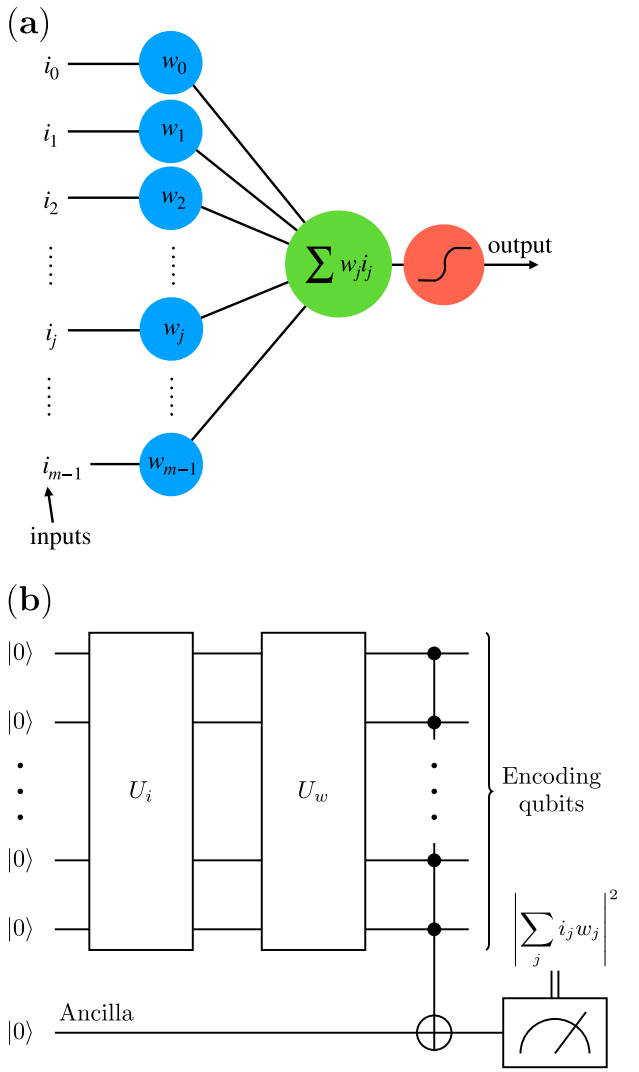

Fig. 1 Perceptron models. **a** Schematic outline of the classical perceptron as a model of artificial neuron: An input array $\vec{i}$ is processed with a weight vector $\vec{w}$ to produce a linear, binary-valued output function. In its simplest realization, also the elements of $\vec{i}$ and $\vec{w}$ are binary valued, the perceptron acting as a binary (linear) classifier. **b** Scheme of the quantum algorithm for the implementation of the artificial neuron model on a quantum processor: From the system initialized in its idle configuration, the first two unitary operations prepare the input quantum state, $|\psi_i\rangle$, and implement the U_w transformation, respectively. The final outcome is then written on an ancilla qubit, which is eventually measured to evaluate the activation state of the perceptron

simulate a $4+1$ qubits version using the IBM quantum simulator. We show that this version of the algorithm already allows to implement an elementary training scheme to recognize simple patterns.

RESULTS

Quantum circuit modeling of a classical perceptron

A scheme of the quantum algorithm proposed in this work is shown in Fig. 1b. The input and weight vectors are limited to binary values, $i_j, w_j \in \{-1, 1\}$, as in McCulloch-Pitts neurons. Hence, a m-dimensional input vector is encoded using the m coefficients needed to define a general wavefunction $|\psi_i\rangle$ of N qubits. In practice, given arbitrary input $(\vec{i})$ and weight $(\vec{w})$ vectors

$$\vec{i} = \begin{pmatrix} i_0 \\ i_1 \\ \vdots \\ i_{m-1} \end{pmatrix}, \vec{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{m-1} \end{pmatrix} \tag{1}$$

with $i_j, w_j \in \{-1, 1\}$, we define the two quantum states

$$|\psi_i\rangle = \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} i_j |j\rangle; |\psi_w\rangle = \frac{1}{\sqrt{m}} \sum_{j=0}^{m-1} w_j |j\rangle. \tag{2}$$

The states $|j\rangle \in \{|00\ldots00\rangle, |00\ldots01\rangle, \ldots, |11\ldots11\rangle\}$ form the so-called computational basis of the quantum processor, i.e., the basis in the Hilbert space of N qubits, corresponding to all possible states of the single qubits being either in $|0\rangle$ or $|1\rangle$. As usual, these states are labeled with integers $j \in \{0, \ldots, m-1\}$ arising from the decimal representation of the respective binary string. Evidently, if N qubits are used in the register, there are $m = 2^N$ basis states labeled $|j\rangle$ and, as outlined in Eq. (2), we can use factors ± 1 to encode the m-dimensional classical vectors into a uniformly weighted superposition of the full computational basis.

The first step of the algorithm prepares the state $|\psi_i\rangle$ by encoding the input values in $\vec{i}$. Assuming the qubits to be initialized in the state $|00\ldots00\rangle \equiv |0\rangle^{\otimes N}$, we perform a unitary transformation U_i such that

$$U_i |0\rangle^{\otimes N} = |\psi_i\rangle. \tag{3}$$

In principle, any $m \times m$ unitary matrix having $\vec{i}$ in the first column can be used to this purpose, and we will give explicit examples in the following. Notice that, in a more general scenario, the preparation of the input state starting from a blank register might be replaced by a direct call to a quantum memory[23] where $|\psi_i\rangle$ was previously stored.

The second step computes the inner product between $\vec{w}$ and $\vec{i}$ using the quantum register. This task can be performed efficiently by defining a unitary transformation, U_w, such that the weight quantum state is rotated as

$$U_w |\psi_w\rangle = |1\rangle^{\otimes N} = |m-1\rangle. \tag{4}$$

As before, any $m \times m$ unitary matrix having $\vec{w}^T$ in the last row satisfies this condition. If we apply U_w after U_i, the overall N-qubits quantum state becomes

$$U_w |\psi_i\rangle = \sum_{j=0}^{m-1} c_j |j\rangle \equiv |\phi_{i,w}\rangle. \tag{5}$$

Using Eq. (4), the scalar product between the two quantum states is

$$\begin{aligned} \langle \psi_w | \psi_i \rangle &= \langle \psi_w | U_w^\dagger U_w | \psi_i \rangle \\ &= \langle m-1 | \phi_{i,w} \rangle = c_{m-1}, \end{aligned} \tag{6}$$

and from the definitions in Eq. (2) it is easily seen that the scalar product of input and weight vectors is $\vec{w} \cdot \vec{i} = m \langle \psi_w | \psi_i \rangle$. Therefore, the desired result is contained, up to a normalization factor, in the coefficient c_{m-1} of the final state $|\phi_{i,w}\rangle$. For an intuitive geometrical interpretation see Supplementary Information, Sec. I.

In order to extract such an information, we propose to use an ancilla qubit (a) initially set in the state $|0\rangle$. A multi-controlled NOT gate between the N encoding qubits and the target a leads to:[24]

$$|\phi_{i,w}\rangle |0\rangle_a \rightarrow \sum_{j=0}^{m-2} c_j |j\rangle |0\rangle_a + c_{m-1} |m-1\rangle |1\rangle_a \tag{7}$$

The nonlinearity required by the threshold function at the output of the perceptron is immediately obtained by performing a quantum measurement: indeed, by measuring the state of the ancilla qubit in the computational basis produces the output $|1\rangle_a$

(i.e., an activated perceptron) with probability $|c_{m-1}|^2$. As it will be shown in the following, this choice proves simultaneously very simple and effective in producing the correct result. However, it should be noticed that refined threshold functions can be applied once the inner product information is stored on the ancilla.[25–27] We also notice that both parallel and anti-parallel $\vec{i}-\vec{w}$ vectors produce an activation of the perceptron, while orthogonal vectors always result in the ancilla being measured in the state $|0\rangle_a$. This is a direct consequence of the probability being a quadratic function, i.e., $|c_{m-1}|^2$ in the present case, at difference with classical perceptrons that can only be employed as linear classifiers in their simplest realizations. In fact, our quantum perceptron model can be efficiently used as a pattern classifier, as it will be shown below, since it allows to interpret a given pattern and its negative on equivalent footing. Formally, this intrinsic symmetry reflects the invariance of the encoding $|\psi_i\rangle$ and $|\psi_w\rangle$ states under addition of a global -1 factor.

Implementation of the unitary transformations

One of the most critical tasks to be practically solved when implementing a quantum neural network model is the efficient realization of unitary transformations. In machine learning applications, this might eventually discriminate between algorithms that show truly quantum advantage over their classical counterparts.[13] Here we discuss an original strategy for practically implementing the preparation of the input state $|\psi_i\rangle$ and the unitary transformation U_w on a quantum hardware. In particular, we will first outline the most straightforward algorithm one might think of employing, i.e., the "brute force" application of successive sign flip blocks. Then, we will show an alternative and more effective approach based on the generation of hypergraph states. In the next Section, we will see that only the latter allows to practically implement this quantum perceptron model on a real quantum device.

As a first step we define a sign flip block, $\mathrm{SF}_{N,j}$, as the unitary transformation acting on the computational basis of N qubits in the following way:

$$\mathrm{SF}_{N,j}|j'\rangle = \begin{cases} |j'\rangle & \text{if} \quad j \neq j' \\ -|j'\rangle & \text{if} \quad j = j' \end{cases}. \tag{8}$$

In general, $\mathrm{SF}_{N,j}$ is equivalent to a multi-controlled quantum operation between N qubits: for example, for any N, $m = 2^N$, a controlled Z operation between N qubits ($\mathrm{C}^N\mathrm{Z}$) is a well-known quantum gate[24] realizing $\mathrm{SF}_{N,m-1}$, while a single qubit Z gate acts as $\mathrm{SF}_{1,1}$. For a given input i, the unitary U_i can be obtained by combining simple single qubit rotations and sign flip blocks to introduce the required -1 factors in front of $|j\rangle$ basis states in the representation of the target $|\psi_i\rangle$ (see details in the Methods section). As already anticipated, the whole problem is symmetric under the addition of a global -1 factor (i.e., $|\psi_i\rangle$ and $-|\psi_i\rangle$ are fully equivalent). Hence, there can be only at most $m/2 = 2^{N-1}$ independent -1 factors, and 2^{N-1} sign flip blocks are needed in the worst case. A similar procedure can also lead to the other unitary operation in the quantum perceptron algorithm, U_w. Evidently, the above strategy is exponentially expensive in terms of circuit depth as a function of the number of qubits, and requires an exponential number of N-controlled gates.

A more efficient solution can be given after realizing that the class of possible input- and weight-encoding states, Eq. (2), coincides with the set of the so-called *hypergraph* states (see Supplementary Information, Sec. II, available at https://doi.org/10.1038/s41534-019-0140-4). The latter are ubiquitous ingredients of many renown quantum algorithms, and have been extensively studied and theoretically characterized.[22,28] In particular, hypergraph states can be mapped into the vertices and hyperedges of generalized graphs, and can be prepared by using single qubit and (multi)-controlled Z gates, with at most a single N-controlled $\mathrm{C}^N\mathrm{Z}$ and with the possibility of performing many p-controlled $\mathrm{C}^p\mathrm{Z}$ gates (involving only p qubits, with $p < N$) in parallel. The design of the sequence for U_i and U_w involves a series of iterative steps described in the Methods section, and directly includes the algorithmic generation of hypergraph states based on a procedure that we will call the "hypergraph states generation subroutine" (HSGS). An example of the full sequence for a specific $N = 4$ case is shown in Fig. 2. Notice that our optimized algorithm involving hypergraph states successfully reduces the required quantum resources with respect to the brute force approach outlined in the previous paragraph. However, it may still involve an exponential cost in terms of circuit depth or clock cycles, i.e., of temporal steps of the quantum circuit when all possible parallelization of independent operations on the qubits is taken into account. Indeed, the sign-flip algorithm described above requires $O(2^N)$ N-controlled Z gates when running on N qubits, in the worst case. Since any $\mathrm{C}^N\mathrm{Z}$ can be decomposed into poly(N) elementary single and two-qubit gates,[24] the overall scaling of the sign-flip approach is $O(\mathrm{poly}(N)2^N)$. On the other hand, the worst case for the HSGS, namely the fully connected hypergraph with N vertices, corresponds to applying once all the possible Z and $\mathrm{C}^p\mathrm{Z}$ operations for $2 \le p \le N$. Since all these operations commute, they can be arranged in such a way that all the qubits are always involved either in a single-qubit operation or a multi-controlled one (e.g., a Z on a certain qubit and the $\mathrm{C}^{N-1}\mathrm{Z}$ on the remaining ones can be done in parallel), for any progressive clock cycle. The overall number of clock cycles is still $O(2^N)$, as in the previous case, but now at most one slice contains a N-qubit operation, while all other slices with $p < N$ can be decomposed into poly(p) elementary operations. In this respect, the proposed HSGS optimizes the number of multi-qubit operations. This may be a significant advantage, e.g., in currently available superconducting quantum

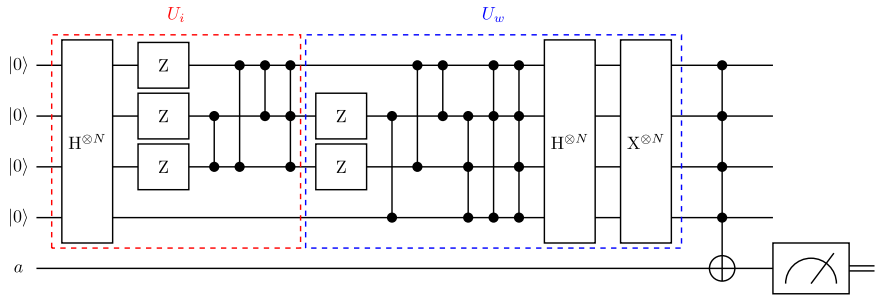

Fig. 2 Quantum circuit of a $N = 4$ perceptron. An example of a typical quantum circuit for a perceptron model with $N = 4$ qubits (i.e., capable of processing $m = 2^4 = 16$ dimensional input vectors), which employs the algorithm for the generation of hypergraph states, including the HSGS (see main text). In this example, the input vector has elements $i_0 = i_1 = -1$, and $i_j = 1$ for $j = 2,\ldots,15$, while the weight vector has elements $w_2 = w_3 = w_4 = -1$, and 1 in all other entries. Multi-controlled $\mathrm{C}^p\mathrm{Z}$ gates are denoted by vertical lines and black dots on the qubits involved. The HSGS is realized inside the U_i block after the initial $\mathrm{H}^{\otimes N}$ gate, and in the U_w block before the final $\mathrm{H}^{\otimes N}$ and $\mathrm{NOT}^{\otimes N}$ operations

processors, in which multi-qubit operations are not natively available.

Before proceeding, it is also worth pointing out the role of U_w in this algorithm, which is essentially to cancel some of the transformations performed to prepare $|\psi_i\rangle$, or even all of them if the condition $\vec{i} = \vec{w}$ is satisfied. Further optimization of the algorithm, lying beyond the scope of the present work, might, therefore, be pursued at the compiling stage. However, notice that the input and weight vectors can, in practical applications, remain unknown or hidden until runtime.

Numerical results and quantum simulations

We implemented the algorithm for a single quantum perceptron both on classical simulators working out the matrix algebra of the circuit and on cloud-based quantum simulators, specifically the IBM Quantum Experience real backends (https://quantumexperience.ng.bluemix.net) using the Qiskit Python development kit (https://qiskit.org/). Due to the constraints imposed by the actual IBM hardware in terms of connectivity between the different qubits, we limited the quantum simulation on the actual quantum processor to the $N = 2$ case. Nevertheless, even this small-scale example is already sufficient to show all the distinctive features of our proposed set up, such as the exponential growth of the analyzable problems dimension, as well as the pattern recognition potential. In general, as already mentioned, in this encoding scheme N qubits can store and process 2^N-dimensional input and weight vectors. Thus, 2^{2^N} different input patterns can be analyzed, corresponding also to the number of different $\vec{w}$ that could be defined. Moreover, all binary inputs and weights can easily be converted into black and white patterns, thus providing a visual interpretation of the artificial neuron activity.

Going back to the case study with $N = 2$, $2^2 = 4$ binary images can be managed, and thus $2^{2^2} = 16$ different patterns could be analyzed. The conversion between $\vec{i}$ or $\vec{w}$ and 2×2 pixels visual patterns is done as follows. As depicted in Fig. 3a, we label each image by ordering the pixels left to right, top to bottom, and assigning a value $n_j = 1(0)$ to a white (black) pixel. The corresponding input or weight vector is then built by setting $i_j = (-1)^{n_j}$, or $w_j = (-1)^{n_j}$. We can also univocally assign an integer label k_i (or k_w) to any pattern by converting the binary string $n_0n_1n_2n_3$ to its corresponding decimal number representation. Under this encoding scheme, e.g., numbers 3 and 12 are used to label patterns with horizontal lines, while 5 and 10 denote patterns with vertical lines, and 6 and 9 are used to label images with checkerboard-like pattern. An example of the sequence of operations performed on the IBM quantum computer using hypergraph states is shown in Fig. 3c for $\vec{i}$ corresponding to the index $k_i = 11$, and $\vec{w}$ corresponding to $k_w = 7$.

The Hilbert space of 2 qubits is relatively small, with a total of 16 possible values for $\vec{i}$ and $\vec{w}$. Hence, the quantum perceptron model could be experimentally tested on the IBM quantum computer for all possible combinations of input and weights. The results of these experiments, and the comparison with classical numerical simulations, are shown in Fig. 3d–f. First, we plot the ideal outcome of the quantum perceptron algorithm in Fig. 3d, where both the global -1 factor and the input-weight symmetries are immediately evident. In particular, for any given weight vector $\vec{w}$, the perceptron is able to single out from the 16 possible input patterns only $\vec{i} = \vec{w}$ and its negative (with output $|c_{m-1}|^2 = 1$, i.e., the perfect activation of the neuron), while all other inputs give outputs smaller than 0.25. If the inputs and weights are translated into 2×2 black and white pixel grids, it is not difficult to see that a single quantum perceptron can be used to recognize, e.g., vertical lines, horizontal lines, or checkerboard patterns.

The actual experimental results are then shown in Fig. 3e, f, where the same algorithm is run on the IBM Q 5 "Tenerife" quantum processor.[29] First, we show in panel 3e the results of the non-optimized approach introduced in the previous Section, which makes direct use of sign flip blocks. We deliberately did not take into account the global sign symmetry, thus treating any $|\psi_i\rangle$ and $-|\psi_i\rangle$ as distinct input quantum states and using up to 2^N sign

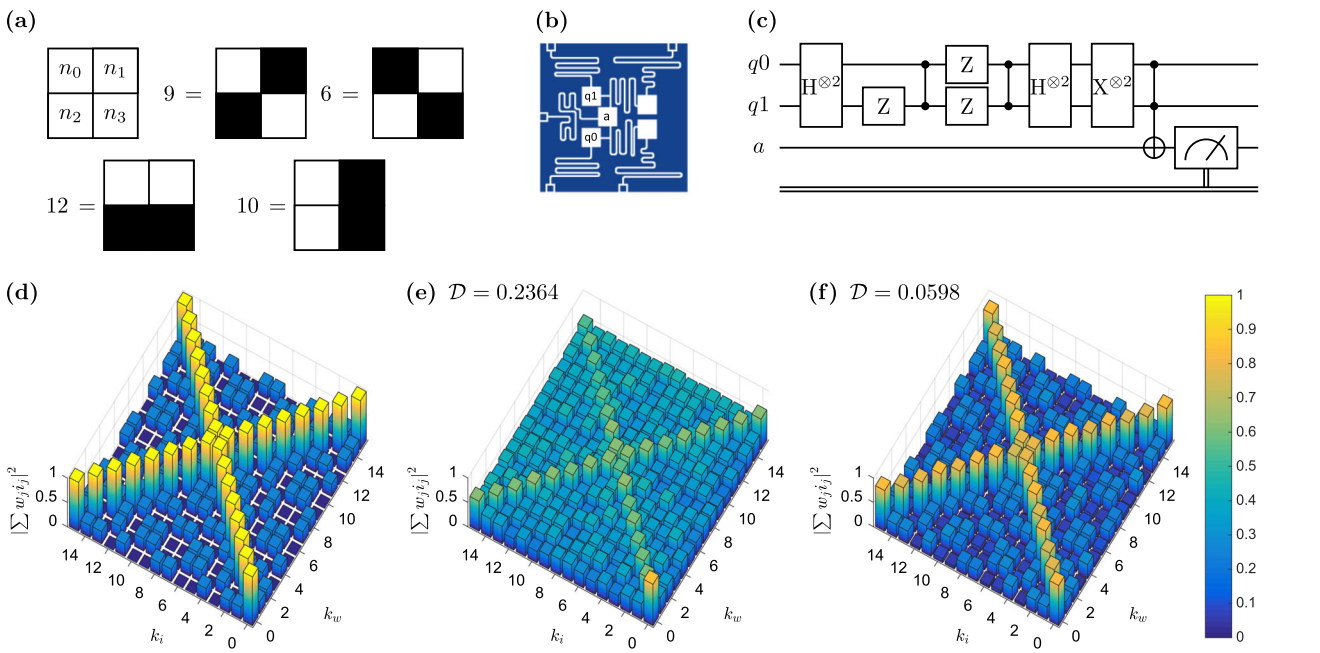

Fig. 3 Results for $N = 2$ quantum perceptron model. **a** The scheme used to label the 2×2 patterns and a few examples of patterns. **b** Scheme of IBM Q-5 "Tenerife" backend quantum processor. **c** Example of the gate sequence for the $N = 2$ case, with input and weight vectors corresponding to labels $k_i = 11$ and $k_w = 7$. **d** Ideal outcome of the quantum perceptron algorithm, simulated on a classical computer. **e** Results from the Tenerife processor using the algorithm with multi-controlled sign flip blocks. **f** Results from the Tenerife processor using the algorithm for the generation of hypergraph states. In **e** and **f** we explicitly indicate the corresponding average discrepancies calculated with respect to the ideal case, as defined in the main text

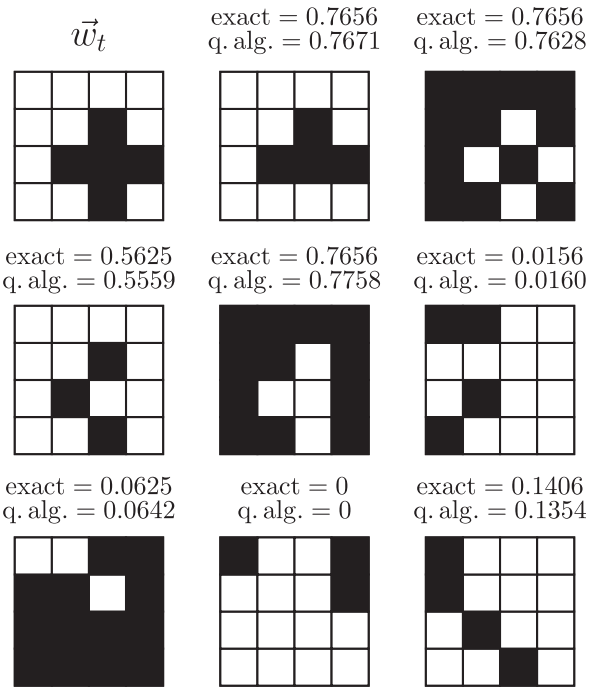

Fig. 4 Pattern recognition for $N = 4$. A possible choice of the weight vector, $\vec{w}_t$, for the $N = 4$ case is represented in the first panel (top left), and a small selection of different input vectors are then simulated with the quantum model of perceptron. Above each input pattern, the quantitative answers of the artificial neuron, namely the values of $|c_{m-1}|^2$, are reported as obtained either through standard linear algebra (ideal 'exact' results) or resulting from the simulation of the quantum algorithm ('q. alg', run on a classical computer, averaged over $n_{shots} = 8192$ repetitions). The two versions agree within statistical error

flip blocks. We notice that even in such an elementary example the algorithm performs worse and worse with increasing number of blocks. However, it should also be emphasized that the underlying structure of the output is already quite evident, despite the quantitative inaccuracy of the quantum simulated outputs: indeed, a threshold of 0.5 applied to the measured output would be sufficient to successfully complete all the classification tasks, i.e. the simulated artificial neuron can correctly single out from all possible inputs any given (precalculated) weight vector.

On the other hand, a remarkably better accuracy, also on the quantitative side and with smaller errors, is obtained when using the algorithm based on the hypergraph states formalism, whose experimental results are shown in panel 3f and represent the main result of this work. In this case, the global phase symmetry is naturally embedded in the algorithm itself, and the results show symmetric performances all over the range of possible inputs and weights. All combinations of $\vec{i}$ and $\vec{w}$ yield results either larger than 0.75 or smaller than 0.3, in good quantitative agreement with the expected results plotted in panel 3d. Again, as it is also clear from the appearance of the plots, all the classification tasks are correctly carried out. In order to give a quantitative measure of the overall agreement between the ideal (Fig. 3d), sign flips (Fig. 3e) and hypergraph states (Fig. 3f) versions, we introduce the average discrepancy as

$$\mathcal{D} = \frac{\sum_{i,w} |O(i,w) - O_{ideal}(i,w)|}{2^{2^{N+1}}} \tag{9}$$

where $O(i,w) = |\sum_j i_j w_j|^2 = |c_{m-1}|^2$ is the outcome of the artificial neuron for a given pair of input and weights as obtained on a real device, $O_{ideal}(i,w)$ is the corresponding ideal result and $2^{2^{N+1}}$ is the total number of possible $\vec{i} - \vec{w}$ pairs. As reported also in Fig. 3, we find $\mathcal{C} \simeq 0.2364$ for the sign flips case, and $\mathcal{D} \simeq 0.0598$ for the version involving hypergraph states. As a technical warning, we finally notice that in all of the three cases shown in panels d-f of Fig. 3, the C^pZ operations were obtained by adding single qubit Hadamard gates on the target qubit before and after the corresponding C^pNOT gate. For $p = 1$ this is a CNOT gate, which is natively implemented on the IBM quantum hardware, while the case $p = 2$ is known as the Toffoli gate, for which a standard decomposition into 6 CNOTs and single qubit rotations is known.[24]

Finally, in the spirit of showing the potential scalability and usefulness of this quantum model of a classical perceptron for classification purposes, we have applied the HSGS-based algorithm to the $N = 4$ qubits case by using the circuit simulator feature available in Qiskit (https://qiskit.org/ and https://qiskit.org/terra). For $N = 4$, 2^{32} possible combinations of $\vec{i}$ and $\vec{w}$ vectors are possible, far too many to explore the whole combinatorial space as previously done for the 2 qubits in Fig. 3. To explicitly show a few examples, we have chosen a single weight vector, $\vec{w}_t$, corresponding to a simple cross-shaped pattern when represented as a 4×4 pixels image (encoded along the same lines of the $N = 2$ case, see first panel in Fig. 3), and weighted it against several possible choices of input vectors. When a threshold $O(i, w_t) > 0.5$ is applied to the outcome of the artificial neuron, 274 over the total 2^{16} possible inputs are selected as positive cases, and they correspond to patterns differing from $\vec{w}_t$ (or from its negative) by at most two pixels. Geometrically speaking, the vectors corresponding to positive cases all lie within a cone around $\vec{w}_t$. Some results are reported in Fig. 4 for a selected choice of input vectors, where the artificial neuron output is computed both with standard linear algebra and with a quantum circuit on a virtual and noise-free quantum simulator run on a classical computer. A larger set of examples is also reported in the Supplementary Information, Sec. IV.

Based on these results, we have implemented an elementary hybrid quantum-classical training scheme, which is an adaptation of the perceptron update rule[30] to our algorithm. After preparing a random training set containing a total of 3050 different inputs, of which 50 positive and 3000 negative ones according to the threshold defined above, the binary valued artificial neuron is trained to recognize the targeted $\vec{w}_t$. This is obtained by using the noiseless Qiskit simulator feature, in which the artificial neuron output is computed through our proposed quantum algorithm, and the optimization of the weight vector is performed by a classical processor. We selected a random $\vec{w}_0$ vector to start with, and then we let the artificial neuron process the training set according to well-defined rules and learning rates l_p and l_n for positive and negative cases, respectively, without ever conveying explicit information about the target $\vec{w}_t$ (see further details in the Methods). An example of the trajectory of the system around the configuration space of possible patterns is shown in Fig. 5a, in which we computed the fidelity of the quantum state $|\psi_w\rangle$ encoding the trained $\vec{w}$ with respect to the target state $|\psi_{w_t}\rangle$. In Fig. 5b, we report the average value of such fidelity over 500 realizations of the training scheme, all with the same initial pattern $\vec{w}_0$ and the same training set. As it can be seen, the quantum artificial neuron effectively learns the targeted cross-shaped pattern: an animated plot of a sample trajectory is also available in the Supplementary Information (see animated gif online at https://doi.org/10.1038/s41534-019-0140-4). Finally, we mention that $l_p = 0.5$ is found to be the optimal learning rate in our case, with little effect produced by l_n (which we also set to $l_n = 0.5$ for simplicity in the simulations reported here).

DISCUSSION

In summary, we have proposed a model for perceptrons to be directly implemented on near-term quantum processing devices.

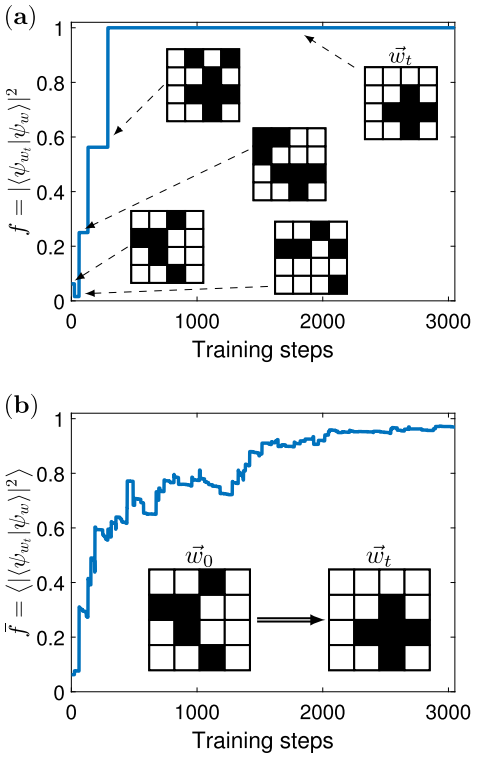

Fig. 5 Training scheme for pattern recognition in the $N = 4$ case. **a** A sample learning trajectory from the initial pattern $\vec{w}_0$, reaching the target cross-shaped $\vec{w}_t$, with some of the intermediate steps. **b** Average fidelity of the quantum state encoding the learned pattern with respect to the target one, obtained by repeating the learning procedure 500 times on the same training set. In both panels, simulations are performed with $l_p = l_n = 0.5$

We have experimentally tested it on a 5-qubits IBM quantum computer based on superconducting technology, and we have proposed a possible hybrid quantum-classical training procedure. In principle, our algorithm presents an exponential advantage in terms of storage capabilities over classical perceptron models, as we have explicitly shown by representing and classifying 4 bits strings using 2 qubits, and 16 bits strings using only 4 qubits. However, a more extended discussion is required on this point. Indeed, generic quantum states or unitary transformations require an exponentially large number of elementary gates to be implemented, and this could somehow hinder the effective advantages brought by quantum computers for machine learning applications. This currently represents a general problem of most quantum machine learning algorithms. Moreover, with increasing N, severe issues can arise from the practical necessity to decompose multiply controlled operations by only using single- and two-qubit gates.[31,32] The latter limitation strictly depends on the effective constraints imposed by the given quantum processor, and on the required degree of accuracy. In fact, it has been shown that several classes of quantum states can be efficiently approximated with arbitrary precision, with oracle based approaches[33–35] or by using a number of multi-controlled rotations that is linear with the number of qubits.[36] Using these results, it might then be possible to design a version of our proposed quantum perceptron algorithm working with approximated encoding quantum states instead of exact ones, which would have the potential to scale exponentially better than any classical algorithm implementing a perceptron model. In this respect, it is also worth pointing out that our procedure is fully general and could be implemented and run on any platform capable of performing universal quantum computation. While we have employed a quantum hardware that is based on superconducting technology and qubits, a very promising alternative is the trapped-ion based quantum computer,[37] in which multi-qubit entangling gates might be readily available.[38,39]

As a further strategy for future developments, we notice that in the present work we restricted the whole analysis to binary inputs and weight vectors (the so-called "McCollough-Pitts" neuron model), mainly for clarity and simplicity of implementation. A possible improvement for the algorithm presented is obviously to encode continuously valued vectors (equivalent to gray-scale images), followed by a hybrid training scheme featuring e.g. backpropagation. This could be achieved by using continuously valued phase factors in $|\psi_i\rangle$ and $|\psi_w\rangle$.[10] Finally, a potentially very exciting continuation of this work would be to connect multiple layers of our quantum perceptrons to build a feedforward deep neural network, which could be fully run on dedicated quantum hardware. As such, our work thus constitutes a concrete first step towards an actual application of near-term (i.e., with few tens of non-error corrected qubits) quantum processors to be employed as fast and efficient trained artificial quantum neural networks.

METHODS

Implementation of the sign flip algorithm

We can implement in practice the whole family of sign-flip blocks for N qubits by using C^NZ gates in combination with single qubit NOT gates (i.e. single bit flip operations):

$$\mathrm{SF}_{N,j} = \mathrm{O}_j(C^N\mathrm{Z})\mathrm{O}_j, \tag{10}$$

where

$$\mathrm{O}_j = \otimes_{l=0}^{m-1}(\mathrm{NOT}_l)^{1-j_l}. \tag{11}$$

In the expression above, NOT_l means that the bit flip is applied to the l-th qubit and $j_l = 0(1)$ if the l-th qubit is in state $|0\rangle(|1\rangle)$ in the computational basis state $|j\rangle$. Alternatively, the same result can also be obtained by using an extra ancillary qubit and multi-controlled NOT gates (C^NNOT), i.e., bit flip operations conditioned on the state of some control qubits (more details and circuit identities can be found in the Supplementary Information, Sec. III, available at https://doi.org/10.1038/s41534-019-0140-4). We explicitly point out that, as it is easily understood from the definition in Eq. (8), any $\mathrm{SF}_{N,j}$ is the inverse of itself. The full sequence to implement U_i can then be summarized as follows: starting from the initialized register $|0\rangle^{\otimes N}$, parallel Hadamard (H) gates are applied to create an equal superposition of all the elements of the computational basis:

$$|0\rangle^{\otimes N} \xrightarrow{\mathrm{H}^{\otimes N}} \frac{1}{\sqrt{m}}\sum_{j=0}^{m-1}|j\rangle \equiv |\psi_0\rangle, \tag{12}$$

where we remind that[24]

$$\mathrm{H}|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}; \mathrm{H}|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \tag{13}$$

Then, the $\mathrm{SF}_{N,j}$ blocks are applied one by one whenever there is a -1 factor in front of $|j\rangle$, in the representation of the target $|\psi_i\rangle$. Notice that any $\mathrm{SF}_{N,j}$ only affects a single element of the computational basis while leaving all others unchanged. Moreover, all $\mathrm{SF}_{N,j}$ blocks commute with each other, so they can actually be performed in any order. The unitary U_w encoding the weight vector can be designed along the same lines. Indeed, applying first the $\mathrm{SF}_{N,j}$ blocks that would be needed to flip all the -1 signs in front of the computational basis elements in the associated $|\psi_w\rangle$ leads to the balanced superposition $|\psi_w\rangle \to |\psi_0\rangle$. This quantum state can then be brought into the desired $|11\ldots11\rangle \equiv |1\rangle^{\otimes N}$ state by applying parallel Hadamard and NOT gates:

$$|\psi_0\rangle \xrightarrow{\mathrm{H}^{\otimes N}} |0\rangle^{\otimes N} \xrightarrow{\mathrm{NOT}^{\otimes N}} |1\rangle^{\otimes N}. \tag{14}$$

HSGS based algorithm

Given a target input vector $\vec{i}$, the unitary U_i can be obtained starting from initial $\mathrm{H}^{\otimes N}$ gate (see Eq. (12)). Then, the algorithm takes a series of iterative steps[22] that are described below. First, we check whether there is any component with only one qubit in state $|1\rangle$ (i.e. of the form $|0\ldots010\ldots0\rangle$)

requiring a -1 factor, in the representation of $|\psi_i\rangle$ on the computational basis. If so, the corresponding single qubit Z gate is applied by targeting the only qubit in state $|1\rangle$. Notice that this might introduce additional -1 factors in front of states with more than one qubit in state $|1\rangle$. Then, for $p = 2, \ldots, N$, we consider the components of the computational basis with exactly p qubits in state $|1\rangle$. For each of them, an additional -1 sign is introduced in front of its current amplitude (if it is needed and it was not previously introduced) by applying the corresponding C^pZ between the p qubits in state $|1\rangle$. Similarly, if an unwanted sign is already present due to a previous step, this can be easily removed by applying the same C^pZ. Since C^pZ acts non trivially only on the manifold with p or more qubits being in state $|1\rangle$, the signs of all the elements with a lower number of $|1\rangle$ components are left unchanged. As a consequence, when $p = N$ all the signs are the desired ones. As in the sign flip case, U_w can be obtained by slightly modifying the sequence of gates that would be used to generate $|\psi_w\rangle$. Indeed, one can start by first performing the HSGS tailored according to the ± 1 factors in $|\psi_w\rangle$. Since all the gates involved in HSGS are the inverse of themselves and commute with each other, this step is equivalent to the unitary transformation bringing $|\psi_w\rangle$ back to the equally balanced superposition of the computational basis states $|\psi_0\rangle$. The desired transformation U_w is finally completed by adding parallel $H^{\otimes N}$ and $NOT^{\otimes N}$ gates (see Eq. (14)).

Training procedure of the artificial neuron

Starting from the initial $\vec{w}_0$, the current state of the artificial neuron, $\vec{w}$, is updated during a training session as follows: whenever the outcome for a given $\vec{i}$ in the training set and the current $\vec{w}$, $O(i, w)$, is correctly above or below the threshold fixed at the beginning $\vec{w}$ is left unchanged, while it is updated when the input is wrongly classified. In particular, if the artificial neuron classifies an input $\vec{i}$ as a positive case ($O(i, w) > 0.5$ here) while it is actually labeled as a negative case in the precalculated training set, $\vec{w}$ is moved farther apart from $\vec{i}$ by flipping a fraction l_n of the ± 1 signs, randomly selected among those positions where the components of $\vec{i}$ and $\vec{w}$ coincide. On the other hand, whenever a positive input $\vec{i}$ is wrongly recognized as a negative one, $\vec{w}$ is moved closer to $\vec{i}$ by flipping a fraction l_p of the ± 1 signs in $\vec{w}$ randomly chosen among those positions where the components of $\vec{i}$ and $\vec{w}$ differ from each other. Here, l_n and l_p play the role of learning rates for the positive and negative cases, and the classical part of the training is performed within a single geometrical m-dimensional hemisphere, thus avoiding confusion between a pattern and its negative. Notice that this distinction is unnecessary in the quantum algorithm. It is also worth pointing out that the fidelity with respect to the target state $|\psi_t\rangle$ is offered in Fig. 5 as a convenient way of conveying to the reader the convergence of the training method. However, such fidelity is never used as a figure of merit during the training procedure: no information about the target $\vec{w}_t$ is provided to the artificial neuron during training, but only the set of inputs labeled as positive and negative.

IBM-Q experiments

The experimental tests performed on real quantum hardware and reported in the main text were obtained by remotely accessing the IBM Quantum Experience platform (https://quantumexperience.ng.bluemix.net). All of the available IBM quantum processors are based on superconducting technology, with transmon qubits pairwise connected through coplanar waveguide (CPW) resonators. Additional CPWs are coupled to each qubit for microwave control and readout. For the proof-of-principle realization of our scheme with $N = 2$ we used three qubits (two of them encoding the artificial neuron inputs and weights, and the other one used as ancilla) on the IBM Q-5 "Tenerife" backend,[29] according to the scheme reported in Fig. 3. Typical, average values for the device parameters are the following: qubit frequency 5.25 GHz, $T_1 = 49.70\,\mu$s, $T_2 = 40.60\,\mu$s, gate error $0.86 \cdot 10^{-3}$, readout error $7.00 \cdot 10^{-2}$. We notice that these values may have changed during calibrations of the hardware following our simulations, see https://quantumexperience.ng.bluemix.net/qx/devices.

Each sequence of quantum gates was designed using the Qiskit python library (https://qiskit.org/), and then run on the quantum processor $n_{shots} = 8192$ times (the maximum allowed on IBM-Q experience) in order to reconstruct the measurement statistics with sufficient accuracy. It is worth pointing out that the real IBM processors natively implement a universal set of quantum gates that contains only general single-qubit Bloch sphere rotations and CNOT gates between selected pairs of qubits, and therefore any unitary operation is decomposed into elementary ones belonging to this set. A reduced version of this work appeared as a preprint on arxiv with reference number arXiv:1811.02266.

DATA AVAILABILITY

All the data and simulations that support the findings of this study are available from the corresponding author upon reasonable request.

CODE AVAILABILITY

The custom python scripts used in this study are available from the corresponding author upon reasonable request.

ACKNOWLEDGEMENTS

We acknowledge the University of Pavia Blue Sky Research project number BSR1732907. This research was also supported by the Italian Ministry of Education, University and Research (MIUR): "Dipartimenti di Eccellenza Program (2018-2022)", Department of Physics, University of Pavia. We acknowledge use of the IBM Quantum Experience for this work. The views expressed are those of the authors and do not reflect the official policy or position of IBM company or the IBM-Q team.

AUTHOR CONTRIBUTIONS

D.B. conceived the original idea, F.T. and D.G. performed quantum simulations, F.T. and C.M. devised the algorithmic procedure based on hypergraph states, D.B. and D. G. supervised the work. All authors contributed to manuscript writing and discussions about the results.

ADDITIONAL INFORMATION

Supplementary information accompanies the paper on the *npj Quantum Information* website (https://doi.org/10.1038/s41534-019-0140-4).

Competing interests: The authors declare no competing interests.

Publisher's note: Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

REFERENCES

1. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **61**, 85–117 (2015).
2. Zurada J. M. *Introduction to Artificial Neural Systems* (West Group, St. Paul, MN, USA, 1992).
3. Rojas R. *Neural Networks: A Systematic Introduction* (Springer, Berlin, Germany, 1996).
4. Schuman C. D. et al. *A Survey of Neuromorphic Computing and Neural Networks in Hardware*, arXiv:1705.06963 (2017).
5. Merolla, P. A. et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* **345**, 668–673 (2014).
6. Sanz, M., Lamata, L. & Solano, E. Quantum memristors in quantum photonics. *APL Photonics* **3**, 080801 (2018).
7. Biamonte, J. et al. Quantum machine learning.*Nature.* **549**, 195–202 (2017).
8. Neukart F. & Moraru S.-A. On Quantum Computers and Artificial Neural Networks. Journal of Signal Processing Research **2** (2013).
9. Schuld, M., Sinayskiy, I. & Petruccione, F. The quest for a Quantum Neural Network. *Quantum Inf. Process.* **13**, 2567–2586 (2014).
10. Schuld, M., Sinayskiy, I. & Petruccione, F. Simulating a perceptron on a quantum computer. *Phys. Lett. A* **7**, 660–663 (2015).
11. Kapoor, A., Wiebe, N. & Svore, K. Quantum Perceptron Models. *Adv. Neural Inf. Process. Syst. (NIPS 2016)* **29**, 3999–4007 (2016).
12. Lloyd S., Mohseni M. & Rebentrost P. Quantum algorithms for supervised and unsupervised machine learning. arXiv:1307.0411 (2013).
13. Schuld, M., Fingerhuth, M. & Petruccione, F. Implementing a distance-based classifier with a quantum interference circuit. *Europhys. Lett.* **119**, 6002 (2017).
14. Lamata, L. Basic protocols in quantum reinforcement learning with superconducting circuits. *Sci. Rep.* **7**, 1609 (2017).
15. Alvarez-Rodriguez, U., Lamata, L., Escandell-Montero, P., Martín-Guerrero, J. D. & Solano, E. Supervised Quantum Learning without Measurements. *Sci. Rep.* **7**, 13645 (2017).
16. Otterbach J. S. et al. Unsupervised Machine Learning on a Hybrid Quantum Computer. arXiv:1712.05771 (2017).

17. Wan, K. H., Dahlsten, O., Kristjánsson, H., Gardner, R. & Kim, M. S. Quantum generalisation of feedforward neural networks. *npj Quantum Inf.* **3**, 36 (2017).
18. Rebentrost, P., Bromley, T. R., Weedbrook, C. & Lloyd, S. Quantum Hopfield neural network. *Phys. Rev. A* **98**, 042308 (2018).
19. Rosenblatt F. The Perceptron: A perceiving and recognizing automaton, Tech. Rep. Inc. Report No. 85-460-1 (Cornell Aeronautical Laboratory, 1957).
20. McCulloch, W. S. & Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.* **5**, 115–133 (1943).
21. Mocanu, D. C. et al. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nat. Commun.* **9**, 2383 (2018).
22. Rossi, M., Huber, M., Bruß, D. & Macchiavello, C. Quantum hypergraph states. *New J. Phys.* **15**, 113022 (2013).
23. Giovannetti, V., Lloyd, S. & Maccone, L. Quantum Random Access Memory. *Phys. Rev. Lett.* **100**, 160501 (2008).
24. Nielsen M. A. & Chuang I. L. *Quantum Computation and Quantum Information (Cambridge Series on Information and the Natural Sciences)* (Cambridge University Press, Cambridge, United Kingdom, 2004).
25. Hu, W. Towards a Real QuantumNeuron. *Nat. Sci.* **10**, 99–109 (2018).
26. Cao Y., Guerreschi G. G. & Aspuru-Guzik A. Quantum Neuron: an elementary building block for machine learning on quantum computers. arXiv:1711.11240 (2017).
27. Torrontegui E. & Garcia-Ripoll J. J. Universal quantum perceptron as efficient unitary approximators. arXiv:1801.00934 (2018).
28. Ghio, M., Malpetti, D., Rossi, M.., Bruß, D. & Macchiavello, C. Multipartite entanglement detection for hypergraph states. *J. Phys. A: Math. Theor.* **51**, 045302 (2018).
29. For hardware specifications and images, see IBM Q team, *IBM Q 5 Tenerife backend specification V1.2.0 and V1.3.0*, (2018), https://ibm.biz/qiskit-tenerife.
30. Bishop C. M. *Pattern Recognition and Machine Learning* (Springer-Verlag, New York, USA, 2006).
31. Bergholm, V., Vartiainen, J. J., Möttönen, M. & Salomaa, M. M. Quantum circuits with uniformly controlled one-qubit gates. *Phys. Rev. A* **71**, 052330 (2005).
32. Plesch, M. & Brukner, Č. Quantum-state preparation with universal gate decompositions. *Phys. Rev. A* **83**, 032302 (2011).
33. Grover L. & Rudolph T. Creating superpositions that correspond to efficiently integrable probability distributions. arXiv:0208112 [quant-ph] (2002).
34. Soklakov, A. N. & Schack, R. Efficient state preparation for a register of quantum bits. *Phys. Rev. A* **73**, 012307 (2006).
35. Clader, B. D., Jacobs, B. C. & Sprouse, C. R. Preconditioned quantum linear system algorithm. *Phys. Rev. Lett.* **110**, 250504 (2013).
36. Mosca M. & Kaye P. *Quantum Networks for Generating Arbitrary Quantum States*, Optical Fiber Communication Conference and International Conference on Quantum Information, PB28 (Optical Society of America, Washington, D.C., USA, 2001).
37. Schindler, P. et al. A quantum information processor with trapped ions. *New J. Phys.* **15**, 123012 (2013).
38. Mølmer, K. & Sørensen, A. Multiparticle entanglement of hot trapped ions. *Phys. Rev. Lett.* **82**, 1835–1838 (1999).
39. Schindler, P. et al. Quantum simulation of dynamical maps with trapped ions. *Nat. Phys.* **9**, 361–367 (2013).