Editors' Suggestion

Quantum Hopfield neural network

Patrick Rebentrost,[1,*] Thomas R. Bromley,[1,†] Christian Weedbrook,[1] and Seth Lloyd[2]
[1]*Xanadu, 372 Richmond Street West, Toronto, Ontario, Canada M5V 1X6*
[2]*Department of Mechanical Engineering, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, Massachusetts 02139, USA*

(Received 19 June 2018; published 5 October 2018)

Quantum computing allows for the potential of significant advancements in both the speed and the capacity of widely used machine learning techniques. Here we employ quantum algorithms for the Hopfield network, which can be used for pattern recognition, reconstruction, and optimization as a realization of a content-addressable memory system. We show that an exponentially large network can be stored in a polynomial number of quantum bits by encoding the network into the amplitudes of quantum states. By introducing a classical technique for operating the Hopfield network, we can leverage quantum algorithms to obtain a quantum computational complexity that is logarithmic in the dimension of the data. We also present an application of our method as a genetic sequence recognizer.

DOI: 10.1103/PhysRevA.98.042308

I. INTRODUCTION

Machine learning is an interdisciplinary approach that brings together the fields of computer science, mathematics, statistics, and neuroscience with the objective of giving computers the ability to make predictions and generalizations from data [1]. A typical machine learning problem falls into three main categories: supervised learning, where the computer learns from a set of training data; unsupervised learning, with the objective of identifying underlying patterns in data; and reinforcement learning, where the computer evolves its approach based on real-time feedback. Machine learning is changing how we interact with technology in areas such as autonomous vehicles, the internet of things, and e-commerce.

Quantum information science has developed from the idea that quantum mechanics can provide improvements in information processing and communication [2]. The promises of quantum information are manifold, ranging from exponentially fast quantum computers, to information-theoretic secure quantum communication networks, to high-precision measurements useful in science and technology. Over the past few decades, quantum information science has transitioned from scientific theory to a viable form of technology.

Given the encouraging technological implications of both machine learning and quantum information science, it was inevitable that their paths would cross to form quantum machine learning [3–6]. Quantum-enhanced machine learning approaches use a toolbox of quantum subroutines to achieve computational speedups for established machine learning algorithms. This toolbox includes fundamentals like quantum basic linear algebra subroutines (qBLAS), including eigenvalue finding [2], matrix multiplication [7], and matrix inversion [8]. One can also build on quantum techniques, such as amplitude amplification [9,10] and quantum annealing [11–13]. These elements have been put together in recent works on quantum machine learning [14–20], including nearest-neighbor clustering [21], the quantum support vector machine [22], and quantum principal component analysis [23,24].

Artificial neural networks are highly successful in machine learning and are hence of special interest for quantum adaptation [14,17,25–27]. A collection of binary or continuous-valued neurons are connected and evolve in such a way that each neuron decides its state based upon a weighted function of the neurons connecting to it. The neurons can be organized into layers and may be configured to allow for backflow of information (known as a recurrent network, often constructed from building blocks of long short-term memory [28]). We focus on the Hopfield network, which is a single-layered, recurrent, and fully connected neural network with undirected connections between neurons. Such networks can be trained using the Hebbian learning rule [29], based on the notion that the connection weights are stronger when they are regularly fired together from training data. The Hopfield network can act as a nonsequential associative memory, with technological application in image processing and optimization [30] and wider interest in neuroscience and medicine.

State-of-the-art neural networks are based on deep learning methods with many hidden layers and using learning rules such as stochastic gradient descent [31,32]. While the Hopfield network is not competitive with these modern neural networks, it is interesting to investigate the *quantum* context for several reasons. The fully visible structure allows a simple encoding of the information into the amplitudes of a quantum state. With such an encoding, techniques such as quantum phase estimation and matrix inversion can be applied which have exponentially fast run times in certain cases. Learning rules such as Hebbian learning find a relatively straightforward representation in the quantum domain. Finally, Hebbian learning and the Hopfield network were one of the early neural network methods and fast quantum algorithms are

*pr@patrickre.com
†tom@xanadu.ai

interesting as building blocks for more advanced quantum networks.

We present in this article a method to construct a quantum version of the Hopfield network (qHop), resulting from an adaptation of the classical Hopfield network when specialized to the situation of information erasure. The network state is embedded into the amplitudes of a quantum system composed of a register of quantum bits (qubits). Our approach differs from previous generalizations of the Hopfield network; Refs. [33,34] focused on the condensed matter and biology setting, Ref. [35] encoded neurons directly into qubits, Ref. [36] used a quantum search, while Ref. [37] harnessed quantum annealing. The training of qHop is here addressed by introducing quantum Hebbian learning, whereby the symmetric graph weighting matrix can be associated to a density matrix stored in a qubit register. We show how this density matrix can be used operationally to imprint relevant training information onto the system. The next step is to operate qHop efficiently. To this end, we propose an approach to optimizing the classical Hopfield network using matrix inversion. Matrix inversion can under certain conditions be performed efficiently using quantum algorithms with a run time $O(\text{poly}(\log d))$ in the size of the matrix d [8]. By combining these algorithms with the quantum Hebbian learning subroutine and sparse Hamiltonian simulation [38], we formalize our algorithm qHop. Using qHop can therefore provide speedups in the application of the Hopfield network as a content-addressable memory system. As an example application, we consider the problem of RNA sequence pattern recognition of the influenza A virus in genetics. We use this scenario to compare the recovery performances of both approaches to operating the Hopfield network.

II. NEURAL NETWORKS

Let us first outline some basic features of neural networks. Consider a collection of d artificial binary-valued neurons $x_i \in \{1, -1\}$ with $i \in \{1, 2, \ldots, d\}$ [39], that are together described by the activation pattern vector $\boldsymbol{x} = \{x_1, x_2, \ldots, x_d\}^\mathsf{T}$, with $\boldsymbol{x}^\mathsf{T}$ denoting the transpose of $\boldsymbol{x}$. The neurons are formed into a (potentially multilayer) network by wiring them to create a connected graph, which can be specified by a real and square $(d \times d)$-dimensional weighting matrix W. Its elements w_{ij} specify the neuronal connection strength between neurons i and j [40]. We note that each neuron is not typically self-connected, so that $w_{ii} = 0$. Furthermore, for an undirected network, W is symmetric. In addition, we may also use continuously activated neurons in both classical and quantum settings, but focus in this work on the binary case for the input and test patterns.

Setting the weight matrix W is achieved by teaching the network a set of training data. These training data can consist of known activation patterns for the visible neurons, i.e., the input and output neurons, with the learning achieved using tools such as back-propagation, gradient descent, and Hebbian learning. A network can be fully visible, so that every neuron acts as both an input and an output.

The Hopfield network is a single-layered, fully visible, and undirected neural network. Here, one can teach the network using the Hebbian learning rule [29]. This rule sets the weighting matrix elements w_{ij} according to the number of occasions in the training set that the neurons i and j fire together. Consider a training set of M activation patterns $\boldsymbol{x}^{(m)}$, with $m \in \{1, 2, \ldots, M\}$. The (normalized) weighting matrix is given by

$$W = \frac{1}{Md}\left[\sum_{m=1}^{M} \boldsymbol{x}^{(m)}(\boldsymbol{x}^{(m)})^\mathsf{T}\right] - \frac{\mathbb{I}_d}{d}, \tag{1}$$

with $\mathbb{I}_d$ the d-dimensional identity matrix.

III. QUANTUM NEURAL NETWORKS

Now we consider the task of using multiqubit quantum systems to construct quantum neural networks. One established method is to have a direct association between neurons and qubits [25], unlocking access to quantum properties of entanglement and coherence. We instead encode the neural network into the amplitudes of a quantum state. This is achieved by introducing an association rule between activation patterns of the neural network and pure states of a quantum system. Consider any d-dimensional vector $\boldsymbol{x} := \{x_1, x_2, \ldots, x_d\}^\mathsf{T}$. We associate it to the pure state $|x\rangle$ of a d-level quantum system according to $\boldsymbol{x} \to |\boldsymbol{x}|_2 |x\rangle$, with $|\boldsymbol{x}|_2 = \sqrt{\sum_{i=1}^{d} x_i^2}$ the l_2-norm of x and $|x\rangle := \frac{1}{|\boldsymbol{x}|_2}\sum_{i=1}^{d} x_i |i\rangle$ written with respect to the standard basis such that $\langle x|x\rangle = 1$. Note that for activation pattern vectors with $x_i = \pm 1$, the normalization is $|\boldsymbol{x}|_2^2 = d$. The d-level quantum system can be implemented by a register of $N = \lceil \log_2 d \rceil$ qubits, so that the qubit overhead of representing such a network scales logarithmically with the number of neurons. We discuss in the following section how the weighting matrix W can be understood in the quantum setting by using quantum Hebbian learning.

Crucial for quantum adaptations of neural networks is the classical-to-quantum read-in of activation patterns. In our setting, reading in an activation pattern $\boldsymbol{x}$ amounts to preparing the quantum state $|x\rangle$. This could in principle be achieved using the developing techniques of quantum random access memory (qRAM) [41] or efficient quantum state preparation, for which restricted, oracle-based, results exist [42]. In both cases, the computational overhead can be logarithmic in terms of d. State preparation routines can potentially be made more robust by the insight that certain errors can be tolerated in the machine learning setting [43]. One can alternatively adapt a fully quantum perspective and take the activation patterns $|x\rangle$ directly from a quantum device or as the output of a quantum channel. For the former, our preparation run time is efficient whenever the quantum device is composed of a number of gates scaling at most polynomially with the number of qubits. Instead, for the latter, we typically view the channel as some form of fixed system-environment interaction that does not require a computational overhead to implement.

IV. QUANTUM HEBBIAN LEARNING

Using our association rule, the training set of activation patterns $\boldsymbol{x}^{(m)}$ can be associated with an ensemble of pure quantum states $|x^{(m)}\rangle$. Let us now focus on the Hopfield network, with a weighting matrix W. We first introduce the *quantum* Hebbian learning algorithm (qHeb), which relies on

two important insights: (i) that one can associate the weighting matrix W directly to a mixed state ρ of a memory register of N qubits according to

$$\rho := W + \frac{\mathbb{I}_d}{d} = \frac{1}{M}\sum_{m=1}^{M} |x^{(m)}\rangle \langle x^{(m)}|, \tag{2}$$

and (ii) that one can efficiently perform quantum algorithms that harness the information contained in W.

To comment on insight (i), the problem of efficient preparation of $|x^{(m)}\rangle$ can be addressed using any of the techniques discussed in the previous section. We denote by T_{in} the required run time to prepare each $|x^{(m)}\rangle$. In the situations discussed above $T_{\text{in}} \in O(\text{poly}(\log d))$.

Regarding insight (ii), now suppose that we have prepared ρ in the laboratory and want to harness the training information contained within. If ρ is the direct output of an unknown quantum device, then we cannot recover the training states $|x^{(m)}\rangle$, since the decomposition of ρ into pure states is not unique. On the other hand, we can still obtain useful information about ρ, such as its eigenvalues and eigenstates. One approach to do this could be to perform a full quantum state tomography of ρ. For states with low rank r, there exist tomographical techniques with a run time $O(\text{poly}(d \log d, r))$ [44], although for some cases the required run time for full state tomography can grow polynomially with the number of qubits [45].

We show that one can use ρ as a "quantum software state" [24]. That is, it is possible to efficiently simulate $e^{i\rho t}$ for time t to precision ϵ with a required run time approximately $T_{\text{qHeb}} \in O(\text{poly}(\log d, t, M, \frac{1}{\epsilon}))$. One can then use this ability to estimate the eigenvalues and eigenstates of ρ to precision ϵ through the quantum phase estimation algorithm [2], requiring an overall run time $T_{\text{eigenvalues}} \in O(\text{poly}(\log d, \frac{1}{\epsilon}, M))$.

Let us define the set of M unitary operators $\{\mathcal{U}_k\}_{k=1}^{M}$ acting on an $N+1$ register of qubits according to

$$\mathcal{U}_k := |0\rangle\langle 0| \otimes \mathbb{I} + |1\rangle\langle 1| \otimes e^{-i|x^{(k)}\rangle\langle x^{(k)}|\Delta t}. \tag{3}$$

The unitaries apply the different memory pattern projectors $|x^{(k)}\rangle\langle x^{(k)}|$ conditionally and for a small time Δt. We now show how to simulate these unitaries and that one can simulate a conditional $e^{-i\rho t}$ by applying them for a suitably large number of times. Let S be the swap matrix between the subsystems for σ and $|x^{(k)}\rangle$. Note that

$$\begin{aligned}\mathcal{U}_S &:= e^{-i|1\rangle\langle 1|\otimes S\Delta t} \\ &= |0\rangle\langle 0| \otimes \mathbb{I} + |1\rangle\langle 1| \otimes e^{-iS\Delta t},\end{aligned} \tag{4}$$

where $|1\rangle\langle 1| \otimes S$ is 1-sparse and efficiently simulatable. For sparse Hamiltonian simulation, the methods in Refs. [38,46] can be used with a constant number of oracle calls and run time $\tilde{O}(\log d)$, where we omit polylogarithmic factors in O by use of the symbol $\tilde{O}$. Note that

$$\begin{aligned}&\text{tr}_2\{\mathcal{U}_S(|q\rangle\langle q| \otimes |x^{(k)}\rangle\langle x^{(k)}| \otimes \sigma)\mathcal{U}_S^\dagger\} \\ &\quad = \mathcal{U}_k(|q\rangle\langle q| \otimes \sigma)\mathcal{U}_k^\dagger + O(\Delta t^2).\end{aligned} \tag{5}$$

The trace is over the second subsystem containing the state $|x^{(k)}\rangle$. Thus the subsystem of ancilla qubit and σ effectively undergoes time evolution with $\mathcal{U}_k$.

We now apply the M unitaries $\mathcal{U}_k$ sequentially for n repetitions. That is, we perform

$$U_t := \left(\prod_{k=1}^{M} \mathcal{U}_k\right)^n \tag{6}$$

with $\Delta t = t/nM$. Consider for the sake of simplicity the unconditioned evolution. Using the standard Suzuki-Trotter method [47], it follows that

$$\begin{aligned}\epsilon := \|&(e^{-i|x^{(1)}\rangle\langle x^{(1)}|t/(nM)} \cdots e^{-i|x^{(M)}\rangle\langle x^{(M)}|t/(nM)})^n \\ &- e^{-i\rho t}\| \in O\left(\frac{t^2}{n}\right).\end{aligned} \tag{7}$$

Hence, we require $n \in O(\frac{t^2}{\epsilon})$ repetitions, with each repetition requiring M sparse Hamiltonian simulations. This results in a run time $O(\frac{Mt^2}{\epsilon})$. The advantages of this approach are that we can use copies of the training states $|x^{(m)}\rangle$ as "quantum software states" [24] and, in addition, we do not require superpositions of the training states. In summary, we can simulate ρ conditionally to a precision ϵ with a number of applications of $\mathcal{U}_k$ of order $O(Mt^2/\epsilon)$. Each $\mathcal{U}_k$ can be realized with logarithmic run time using sparse Hamiltonian simulation [38], resulting in the overall run time of $T_{\text{qHeb}} \in O(\text{poly}(\log d, t, M, \frac{1}{\epsilon}))$.

The quantum phase estimation algorithm [2,8] can then be implemented to find the eigenvalues $\mu_j(\rho)$ and corresponding eigenstates $|v_j(\rho)\rangle$ of ρ. Here we prepare a register of T qubits additional to our register of N qubits in the composite state $\sum_{t=1}^{2^T} |t\rangle \otimes |\psi\rangle$ for some arbitrary $|\psi\rangle$. The size of T is set by the precision with which we wish to estimate the eigenvalues. Applying the controlled unitaries U_t results in the state $\sum_j \beta_j |\tilde{\mu}_j(\rho)\rangle \otimes |v_j(\rho)\rangle$. Each $|\tilde{\mu}_j(\rho)\rangle$ contains an approximation of the eigenvalues $\mu_j(\rho)$ [2], and $\beta_j := \langle v_j(\rho)|\psi\rangle$. If we take $2^T \in O(1/\epsilon)$, we can estimate the eigenvalues of ρ to precision ϵ with a number of copies of the memory states $|x^{(m)}\rangle$ of the order $O(M/\epsilon^3)$. This results in an overall run time $T_{\text{eigenvalues}} \in O(\text{poly}(\log d, \frac{1}{\epsilon}, M))$. Our quantum Hebbian learning method thus shows how to prepare the weight matrix from the training data as a mixed quantum state and then specifies how that density matrix can be used in a quantum algorithm for higher-level machine-cognitive function, specifically to learn eigenvalues and eigenvectors.

V. THE HOPFIELD NETWORK

We return to the classical Hopfield network and discuss its operation, having already shown the Hebbian learning rule to store M activation patterns in the weighting matrix W (see also Fig. 1 for a diagram). Suppose that we are supplied with a new activation pattern, $\boldsymbol{x}^{(\text{new})}$, in the form of a noise-degraded version of one from the training set or alternatively a similar pattern that is to be compared to the training set. In the following, we show the standard way of operating the network and then develop a method based on matrix inversion.

The standard method of operating the Hopfield network proceeds by initializing it in the activation $\boldsymbol{x}^{(\text{new})}$ and then running an iterative process whereby neuron i is selected at

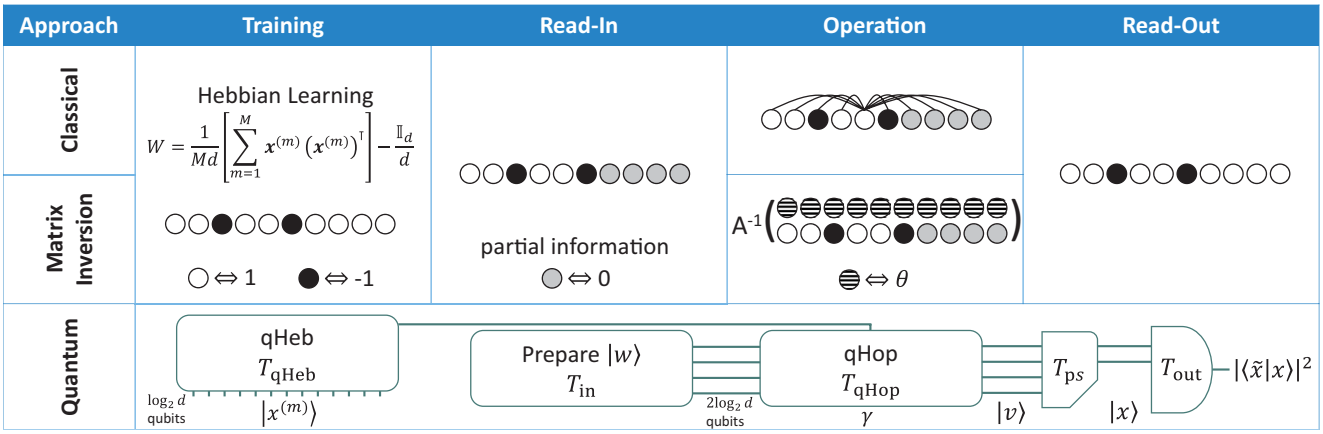

FIG. 1. The classical and quantum Hopfield networks. We discuss three approaches to operating the network. The standard classical approach is to iteratively update the neurons based on the connections to neighboring neurons. Our developed classical approach solves a relaxation of the problem posed by a linear equation system and solvable through matrix inversion. Hebbian learning is employed to set the weighting matrix W from d-length training data $\{\mathbf{x}^{(m)}\}_{m=1}^{M}$. The third approach uses qHop, encoding data in order $\log_2 d$ qubits. Here, the pure state $|w\rangle$ is first prepared which contains user-defined neuron thresholds and a partial memory pattern. Our qHop algorithm proceeds to calculate $|v\rangle = A^{-1}|w\rangle$, with the matrix A containing information on the training data and regularization γ. To achieve this, we introduce the quantum Hebbian learning algorithm qHeb for density matrix exponentiation of the mixture ρ detailing training data $|x^{(m)}\rangle$. The output pure state $|v\rangle$ contains information on the reconstructed state $|x\rangle$ and Lagrange multipliers, which are postselected out. The result $|x\rangle$ can be accessed through global properties such as the swap test, which uses multiple copies of $|x\rangle$ to measure the fidelity $|\langle\tilde{x}|x\rangle|^2$ with another state $|\tilde{x}\rangle$. The required run time for each step is given by the subscripted T.

random and updated according to the rule

$$x_i \to \begin{cases} +1 & \text{if } \sum_{j=1}^{d} w_{ij} x_j \geqslant \theta_i \\ -1 & \text{otherwise,} \end{cases} \tag{8}$$

with $\boldsymbol{\theta} := \{\theta_i\}_{i=1}^{d} \in \mathbb{R}^d$ a user-specified neuronal threshold vector that determines the switching threshold for each neuron. Each element θ_i should be set so that its magnitude is of order at most 1. The result of every update is a nonincrease of the network energy

$$E = -\tfrac{1}{2}\boldsymbol{x}^{\mathsf{T}} W \boldsymbol{x} + \boldsymbol{\theta}^{\mathsf{T}} \boldsymbol{x}, \tag{9}$$

with the network eventually converging to a local minimum of E after a large number of iterations.

Since W has been fixed due to the Hebbian learning rule so that each $\boldsymbol{x}^{(m)}$ is a local minimum of the energy, the output of the Hopfield network is ideally one of the trained activation patterns. The utility of such a memory system is clear and the Hopfield network has been directly employed, for example, in imaging [30].

We now introduce another approach to operating the classical Hopfield network (see Fig. 1). Suppose that we are supplied with incomplete data on a neuronal activation pattern such that we only know the values of $l < d$ neurons with labels $\mathcal{L} \subset \{1, 2, \ldots, d\}$. This setting corresponds to noise-free information erasure. We can initialize our activation pattern to be $\boldsymbol{x}^{(\text{inc})} := \{x_1^{(\text{inc})}, x_2^{(\text{inc})}, \ldots, x_d^{(\text{inc})}\}^{\mathsf{T}}$ with $x_i^{(\text{inc})} = x_i^{(\text{new})}$ if $i \in \mathcal{L}$ and $x_i^{(\text{inc})} = 0$ otherwise. Our objective is to use the trained Hopfield network to recover the original activation pattern $\boldsymbol{x}^{(\text{new})}$. An alternative use of the Hopfield network when supplied with a noisy new pattern is shown in Appendix A.

Let us first define the projector P onto the subspace of known neurons, such that P is diagonal with respect to the standard basis. We proceed by minimizing the energy E in Eq. (9) subject to the constraint that $P\boldsymbol{x} = \boldsymbol{x}^{(\text{inc})}$. The Lagrangian for this optimization is

$$\mathscr{L} = -\frac{1}{2}\boldsymbol{x}^{\mathsf{T}} W \boldsymbol{x} + \boldsymbol{\theta}^{\mathsf{T}} \boldsymbol{x} - \boldsymbol{\lambda}^{\mathsf{T}}(P\boldsymbol{x} - \boldsymbol{x}^{(\text{inc})}) + \frac{\gamma}{2}\boldsymbol{x}^{\mathsf{T}} \boldsymbol{x}, \tag{10}$$

where we introduce a Lagrange multiplier vector $\boldsymbol{\lambda} \in \mathbb{R}^d$ and a fixed regularization parameter $\gamma \geqslant 1$. The first-order derivative conditions for optimization are evaluated as

$$\begin{aligned} \frac{\partial \mathscr{L}}{\partial \boldsymbol{x}} &= (\gamma \mathbb{I}_d - W)\boldsymbol{x} + \boldsymbol{\theta} - P\boldsymbol{\lambda} \stackrel{!}{=} 0, \\ \frac{\partial \mathscr{L}}{\partial \boldsymbol{\lambda}} &= -P\boldsymbol{x} + \boldsymbol{x}^{(\text{inc})} \stackrel{!}{=} 0. \end{aligned} \tag{11}$$

One can equivalently consider this as a system of linear equations $A\boldsymbol{v} = \boldsymbol{w}$ with

$$\begin{aligned} A &:= \begin{pmatrix} W - \gamma \mathbb{I}_d & P \\ P & 0 \end{pmatrix}, \\ \boldsymbol{v} &:= \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{\lambda} \end{pmatrix}, \qquad \boldsymbol{w} := \begin{pmatrix} \boldsymbol{\theta} \\ \boldsymbol{x}^{(\text{inc})} \end{pmatrix}. \end{aligned} \tag{12}$$

The solution of this system then provides a vector $\boldsymbol{v}$ consisting of $\boldsymbol{x}$ and $\boldsymbol{\lambda}$, where $\boldsymbol{x}$ extremizes the energy E subject to $P\boldsymbol{x} = \boldsymbol{x}^{(\text{inc})}$. With $\|X\|$ the spectral norm (largest absolute eigenvalue) of a Hermitian matrix X, note from the definition in Eq. (1) for the weight matrix W that $\|W\| \leqslant 1$. In addition, $\|\sigma_x \otimes P\| \leqslant 1$ and hence $\|A\| \in O(\gamma)$. We set a reasonable choice of value for the regularization parameter to be $\gamma \in O(1)$. It is shown in Appendix B that the result of the optimization is necessarily a constrained local minimum of the energy whenever γ is chosen such that $\gamma > \|W\|$.

Hence, it suffices to choose $\gamma > 1$. As the matrix A is rank deficient, we solve the system of equations by applying the pseudoinverse A^{-1} to $\boldsymbol{w}$, recovering a least-squares solution to $\boldsymbol{v}$.

We find that the elements of the resultant vector $\boldsymbol{x}$ are continuous valued; i.e., $x_i \in \mathbb{R}$. This can be interpreted as a larger positive or negative value indicating a stronger confidence for the activation ± 1, respectively. For a particular neuron, the value can then be projected to the nearest element ± 1 to obtain a prediction for the activation of that neuron. The regularization term in the Lagrangian furthermore serves to minimize the l_2-norm $|\boldsymbol{x}|_2$ of $\boldsymbol{x}$, and can be adapted by the user to prevent the optimization returning overly large unconstrained elements; see Appendix C for further details. Our approach to operating the Hopfield network through matrix inversion is tested in the Application section, using the example of RNA sequencing in genetics.

VI. THE QUANTUM HOPFIELD NETWORK

We now show how the Hopfield network can be run efficiently as a combination of quantum algorithms that we call qHop to perform the matrix-inversion-based approach. Utilizing the embedding method for quantum neural networks already discussed, the system of linear equations (12) can be written in terms of pure quantum states as $A|\boldsymbol{v}|_2 |v\rangle = |\boldsymbol{w}|_2 |w\rangle$, with A as before, $P = \sum_{i\in\mathcal{L}} |i\rangle\langle i|$, and

$$\begin{aligned} |v\rangle &:= \frac{1}{|\boldsymbol{v}|_2}(|\boldsymbol{x}|_2 |0\rangle \otimes |x\rangle + |\boldsymbol{\lambda}|_2 |1\rangle \otimes |\lambda\rangle), \\ |w\rangle &:= \frac{1}{|\boldsymbol{w}|_2}(|\boldsymbol{\theta}|_2 |0\rangle \otimes |\theta\rangle + |\boldsymbol{x}^{(\mathrm{inc})}|_2 |1\rangle \otimes |x^{(\mathrm{inc})}\rangle), \end{aligned} \tag{13}$$

being pure states of $N+1$ qubits. Here, $|x^{(\mathrm{inc})}\rangle$ is the normalized quantum state corresponding to the incomplete activation pattern and $|\boldsymbol{x}^{(\mathrm{inc})}|_2^2 = l$. The objective is to optimize the energy function E in Eq. (9) by solving for $\boldsymbol{v} = |\boldsymbol{v}|_2 |v\rangle = A^{-1}|\boldsymbol{w}|_2 |w\rangle$, with A^{-1} the pseudoinverse of A.

It is possible to prepare $A^{-1}|w\rangle$ with a potential run time logarithmic in the dimension of A by utilizing a combination of quantum subroutines. The objective is to use the quantum matrix inversion algorithm in Ref. [8]. This algorithm requires the ability to perform quantum phase estimation using efficient Hamiltonian simulation of A. We now show that one can simulate e^{iAt} by concurrently executing the simulation of a sparse Hamiltonian linked to the projector P as well as qHeb. To achieve efficiency, certain conditions must be met. These conditions are outlined in the following sections.

We want to simulate the unitary e^{iAt} to a fixed error ϵ for arbitrary t. Let us first write

$$\begin{aligned} A &= \begin{pmatrix} \rho - \left(\gamma + \frac{1}{d}\right)\mathbb{I}_d & P \\ P & 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 & P \\ P & 0 \end{pmatrix} + \begin{pmatrix} -\gamma'\mathbb{I}_d & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} \rho & 0 \\ 0 & 0 \end{pmatrix} \\ &=: B + C + D, \end{aligned} \tag{14}$$

where we introduce the $(2d \times 2d)$-dimensional block matrices

$$\begin{aligned} B &= \begin{pmatrix} 0 & P \\ P & 0 \end{pmatrix}, \quad C = \begin{pmatrix} -\gamma'\mathbb{I}_d & 0 \\ 0 & 0 \end{pmatrix}, \\ D &= \begin{pmatrix} \rho & 0 \\ 0 & 0 \end{pmatrix}, \end{aligned} \tag{15}$$

with $\gamma' = \gamma + \frac{1}{d}$. We now split the simulation time t into n small time steps Δt, i.e., so that $t = n\Delta t$, and consider $e^{iA\Delta t}$. The time evolution $e^{iA\Delta t}$ can be simulated by using applications of $e^{iB\Delta t}$, $e^{iC\Delta t}$, and $e^{iD\Delta t}$ via the standard Suzuki-Trotter method. Suppose that one has operators $\mathcal{U}_B(\Delta t)$, $\mathcal{U}_C(\Delta t)$, and $\mathcal{U}_D(\Delta t)$ that simulate $e^{iB\Delta t}$, $e^{iC\Delta t}$, and $e^{iD\Delta t}$ to errors at most $O(\Delta t^2)$, respectively. In many cases much better error scalings exist. Then, $e^{iB\Delta t}e^{iC\Delta t}e^{iD\Delta t}$ is simulated to error also $O(\Delta t^2)$. By simply using the Taylor expansion, we see that the error $\epsilon_{\Delta t}$ of simulating $e^{iA\Delta t}$ is

$$\epsilon_{\Delta t} := \|e^{iA\Delta t} - \mathcal{U}_B(\Delta t)\mathcal{U}_C(\Delta t)U_D(\Delta t)\| \in O(\Delta t^2). \tag{16}$$

This means that by using n repetitions of $\mathcal{U}_B(\Delta t)\mathcal{U}_C(\Delta t)\mathcal{U}_D(\Delta t)$ we can simulate e^{iAt} to an error of $\epsilon \in O(n\Delta t^2)$. Hence, for a fixed error ϵ and time t, one needs to perform $n \in O(\frac{t^2}{\epsilon})$ repetitions of $\mathcal{U}_B(\Delta t)\mathcal{U}_C(\Delta t)\mathcal{U}_D(\Delta t)$.

We now evaluate the run time of performing one such repetition. Consider the block matrix B. Because P is a diagonal projector, B is a 1-sparse self-adjoint matrix, where sparsity is the maximum number of elements in any column or row. A large series of works have addressed the efficient Hamiltonian simulation of sparse matrices. Reference [38] shows that sparse Hamiltonian simulation for a simulation time t to error ϵ can be performed with a run time $T_B \in \tilde{O}(t\log(d)/\epsilon)$. In our case, for the maximum matrix element of B we have $\|B\|_{\max} = 1$ and also $\|B\| = O(1)$. The operator $\mathcal{U}_C(\Delta t)$ is treated in a similar way. Turning these operators $\mathcal{U}$ into their conditional versions and extending into a larger space as in Eqs. (15) is in principle straightforward with the sparse matrix methods. Simulating the operator $\mathcal{U}_D(\Delta t)$ is achieved using Hebbian learning (see Sec. IV) and including a conditioning on an additional ancilla qubit in state $|0\rangle$.

The essential steps of the algorithm are as follows and also summarized in Fig 1. Let the spectral decomposition of A be given by

$$\begin{aligned} A = &\sum_{j:\,|\mu_j(A)|\geqslant\mu} \mu_j(A)\,|v_j(A)\rangle\langle v_j(A)| \\ &+ \sum_{j:\,|\mu_j(A)|<\mu} \mu_j(A)\,|v_j(A)\rangle\langle v_j(A)|\,, \end{aligned} \tag{17}$$

where we have split into two separate sums dependent upon the size of the eigenvalues $\mu_j(A)$ in comparison to a fixed user-defined number $\mu > 0$. As we see in the following, as well as in Appendix D, the chosen value of μ is a trade-off between the run time and the error in calculating the pseudoinverse. The primary matrix inversion algorithm returns (up to

normalization) [8]

$$A^{-1}|w\rangle = \sum_{j:\,|\mu_j(A)|\geqslant\mu} \frac{\beta_j}{\mu_j(A)}|v_j(A)\rangle\,, \tag{18}$$

where $\beta_j = \langle v_j(A)|w\rangle$.

To begin, we first prepare the input state $|w\rangle$ (which contains the threshold data and incomplete activation pattern) and consider it in the eigenbasis of A, i.e., so that $|w\rangle = \sum_j \beta_j |v_j(A)\rangle$. Our qHeb algorithm is then initialized along with sparse Hamiltonian simulation [38] to perform quantum phase estimation, allowing us to obtain $\sum_j \beta_j |\tilde{\mu}_j(A)\rangle \otimes |v_j(A)\rangle$ with $\tilde{\mu}_j(A)$ an approximation of the eigenvalue $\mu_j(A)$ to precision ϵ. We then use a conditional rotation of an ancilla and a filtering process discussed in Ref. [8] to select only the eigenvalues larger than or equal to μ. This is followed by an uncomputing of the first register of T qubits by reversing the quantum phase estimation protocol. After measurement of the ancilla qubit, our result is (up to normalization) the pure state $A^{-1}|w\rangle$.

A note regarding the input state $|x^{(\mathrm{inc})}\rangle$: In principle, for each reconstruction of a new input state, we require new runs of qHeb and qHop. This feature arises from the no-cloning theorem for quantum states. Different from classical computing, one in general cannot efficiently copy intermediate data of single runs of the algorithm for reuse to reconstruct other input patterns. However, one can envision scenarios where one can reconstruct multiple patterns simultaneously via a quantum superposition of the input patterns. Let $|x^{(\mathrm{inc,k})}\rangle$, $k = 1, \ldots, K$ be K patterns. Assume we can prepare superpositions of the form $|x^{(\mathrm{inc,total})}\rangle = \sum_{k=1}^{K} \alpha_k |x^{(\mathrm{inc,k})}\rangle$ or $|x^{(\mathrm{inc,total})}\rangle = \sum_{k=1}^{K} \alpha_k |k\rangle\,|x^{(\mathrm{inc,k})}\rangle$, with coefficients α_k such that the total state is normalized in each case and $|k\rangle$ a label register. Then we can use the qHop algorithm by replacing $|x^{(\mathrm{inc})}\rangle$ by $|x^{(\mathrm{inc,total})}\rangle$. We then are able to extract information about the K patterns from the resulting state (see the discussion of the output state in Sec. VIII). Of course obtaining information on each individual pattern will again require $O(K)$ operations of qHop, but we can hope to extract summary statistics with fewer resources.

VII. ALGORITHM EFFICIENCY

We now turn to addressing the efficiency of qHop. The overall efficiency is not just dependent upon the run time of our primary algorithm, and we must also consider the read-in efficiency of inputting $|w\rangle$ as well as the read-out efficiency of extracting useful information from the output state $|v\rangle$. Here we review the input and run-time efficiencies, while the next section discusses various ways of using the output and their efficiency. The section after briefly compares our qHop to other classical and quantum approaches to operating the Hopfield network.

The input pure state $|w\rangle$ contains data on the user-specified neuronal thresholds $\boldsymbol{\theta}$, along with the incomplete activation pattern $\boldsymbol{x}^{(\mathrm{inc})}$. As we have discussed, the read-in of activation patterns can add a computational overhead to quantum neural network algorithms, potentially canceling any speedups yielded by the algorithm itself. This can be addressed using, e.g., qRAM [41] or efficient state preparation techniques [42], or alternatively by directly accessing the output of a quantum device. Let us denote by T_{in} the run time of inputting $|w\rangle$, which we take to be $O(\mathrm{poly}(\log d))$ using any of the discussed techniques. Note that state preparation techniques may introduce errors themselves, but these can be fixed to ϵ and will typically add a polynomial overhead in ϵ to the run time [42].

Following similar calculations to those discussed in Ref. [8], we see that our algorithm proceeds by a combination of phase estimation of A with run time T_{phase} along with filtering and amplification operations to select the eigenvalues $|\lambda_j(A)| \geqslant \mu$ [8], requiring a run time T_{filter}. Let us consider first phase estimation, which requires us to perform $O(\frac{1}{\epsilon^3})$ calls to e^{iAt}. One can decompose A into three block matrices B, C, and D, corresponding to the off-diagonal projector P, an on-diagonal identity $\mathbb{I}_d$, and, when using Hebbian learning, the embedded mixed training state ρ [see Eq. (12)]. As we have shown, e^{iAt} is well approximated by applying for n short times Δt the unitaries $U_{B/C/D}$ generated by these block matrices, resulting in an error $\epsilon \in O(\frac{t^2}{n})$ or equivalently requiring a number of steps $n = O(t^2/\epsilon)$.

Since both B and C are 1-sparse matrices, we can use efficient sparse Hamiltonian simulation techniques [38] to evaluate $U_{B/C}(\Delta t)$ with run time $T_{B/C} \in O(\mathrm{poly}(\Delta t, \log d, \log(\frac{1}{\epsilon})))$. For the matrix D, we can use the quantum Hebbian learning techniques discussed earlier to simulate for a time Δt, requiring a run time $T_D \in O(\mathrm{poly}(\Delta t, M, \frac{1}{\epsilon}, \log d))$. Note that the state exponentiation technique used for D means that T_D is the dominant run time compared to $T_{B/C}$. Hence, overall we have $T_{\mathrm{phase}} \in O(\mathrm{poly}(M, \log d, \frac{1}{\epsilon}))$. The run time for filtering and amplification adds an additional overhead $T_{\mathrm{filter}} \in O(\frac{1}{\mu})$ [8], meaning that the user should set $1/\mu \in O(\mathrm{poly}(\log d))$ to maintain efficiency. We hence achieve an overall algorithm run time of

$$T_{\mathrm{qHop}} \in O\left(\mathrm{poly}\left(M, \log d, \frac{1}{\epsilon}, \frac{1}{\mu}\right)\right). \tag{19}$$

A note on the M dependence: The maximum capacity of the classical Hopfield network is approximately $d/(2\log d)$ [48] memory patterns. The linear dependence on M of the quantum algorithm means that for achieving a logarithmic dependency on the dimension, qHop has to be operated substantially below the maximum capacity. Any potential exponential speedup arises from the processing of these d-dimensional memory patterns, while the number of the memory patterns has to be relatively small. To extend the range when one may observe speedups, we can consider a scenario when the density weight matrix is directly given and we can use the original density matrix exponentiation scheme [23,24]. This scenario does not require our Hebbian learning and avoids the M dependence. Moreover, in the case when the weight matrix is given via oracle access to the matrix elements and is sparse, one uses the sparse simulation techniques [38]. In this case, we can directly use qHop without requiring the Hebbian learning procedure and the M dependence is absorbed into the oracle.

The output of our algorithm is the pure state $|v\rangle$ given in Eq. (13). We can then measure the first qubit in our $N+1$

qubit register and postselect on $|0\rangle$ to obtain $|x\rangle$. This succeeds with probability $|\boldsymbol{x}|_2^2/(|\boldsymbol{x}|_2^2 + |\boldsymbol{\lambda}|_2^2)$, adding a processing overhead $T_{\text{ps}} \in O(|\boldsymbol{\lambda}|_2^2/|\boldsymbol{x}|_2^2)$. One can see from Eq. (11) that $x_i \in O(1)$ for the constrained neurons $i \in \mathcal{L}$ and $x_i \in O(\frac{1}{\gamma})$ for the unconstrained neurons, so that $|\boldsymbol{x}|_2^2 \in O(d)$ whenever the number of constrained neurons l is of the order d. On the other hand, since $\lambda_i \in O(\gamma)$ for $i \in \mathcal{L}$ and $\lambda_i = 0$ otherwise, we have $|\boldsymbol{\lambda}|_2^2 \in O(d\gamma^2)$. Hence, overall our processing overhead is $T_{\text{ps}} \in O(\gamma^2)$. This means that our choice of γ is in fact a compromise; one must pick $\gamma \geqslant \|W\|$ to guarantee a local minimum, but if γ is too large then we add a run-time overhead to qHop. The next section discusses what to do with the output state at what cost to the efficiency.

VIII. OUTPUT

The next step is naturally to use the information contained in $|x\rangle$ for a given task. One way to use the state is to read out the amplitudes of $|x\rangle$ by performing tomography. However, even for pure states, tomographical techniques can introduce an overhead that scales polynomially with the dimension d [44]. Instead, one has to extract useful information from $|x\rangle$ using other approaches, which typically act globally on $|x\rangle$ rather than directly accessing each of the d amplitudes. Such extraction of global information aligns well with typical situations in machine learning. Machine learning tasks often involve dimensionality reduction or compression. For example an image of many pixels is compressed to a single label ("cat" or "dog") or a short description of the scene in that image. Classification tasks often involve a small number of classes; for example, users of a movie streaming service can be assigned to a relatively small number of categories [49]. In the context of neural networks, both artificial and biological, the state of a single intermediate neuron is rarely important to a learning task, but rather the final goal is to obtain a low-dimensional explanation or action which relies on the output patterns of a larger collection of neurons.

One option to extract global information could be to measure the fidelity with another state $|\tilde{x}\rangle$, such as one of the training states, which can be achieved by performing a swap test with success probability $P_{\text{swap}} = \frac{1}{2}(1 - |\langle\tilde{x}|x\rangle|^2)$ [50]. We can then determine the fidelity to a precision ϵ by performing $O(\frac{P_{\text{swap}}(1-P_{\text{swap}})}{\epsilon^2})$ swap tests between copies of $|x\rangle$ and $|\tilde{x}\rangle$, with each swap test requiring $O(\log d)$ qubit swaps and hence giving an additional run time to qHop of $T_{\text{out}} \in O(\text{poly}(\log d, \frac{1}{\epsilon}))$.

Alternatively, following the spirit of supervised learning, one may have access to a set of p binary-valued observables, corresponding to membership of some classification categories. Measuring the expectation values of these observables with respect to $|x\rangle$ then allows for a classification of $|x\rangle$ with respect to such categories. For a given precision ϵ, each expectation value can be measured with $O(\frac{1}{\epsilon^2})$ repetitions, resulting in a run-time overhead to qHop of $T_{\text{out}} \in O(\text{poly}(\frac{1}{\epsilon}, p, T_{\text{obs}}))$, with T_{obs} the time of the observable measurement.

In addition, one can adopt a fully quantum perspective and view the state $A^{-1}|w\rangle$ (or the postselected activation pattern state $|x\rangle$) as the final output of the algorithm. Our qHop algorithm then acts as an element of a given quantum toolchain, whose action is to reconstruct a quantum state from an incomplete superposition based on the memory stored in ρ, and then to output to the next element in the chain.

IX. COMPARISON

To summarize, the full operation of qHop can be achieved with a run time $O(\text{poly}(M, \log d, \frac{1}{\epsilon}, \frac{1}{\mu}))$, where Fig. 1 visualizes the individual run-time contributions. We now compare this efficiency with both of the classical approaches: the original Hopfield procedure [40], as well as the matrix-inversion-based approach introduced here. It is clear that the original Hopfield procedure has a run-time polynomial in the number of neurons, since one must typically sample every one of the d neurons at least once. On the other hand, the best sparse classical matrix inversion techniques have a run time $O(\text{poly}(d, \frac{1}{\sqrt{\mu}}, \log(\frac{1}{\epsilon}), s))$ [51], where s is the sparsity, and it has been shown in Ref. [8] that this run time cannot be improved even if one needs access only to the expectation values of A. We hence see that qHop is potentially able to operate with lower computational demands for a suitably large d. Of course, better classical algorithms can be found, for example, harnessing the similarities between the Hopfield network and the Ising model that is studied in depth in quantum physics [6,52]. Techniques such as simulated annealing [53] and mean-field theory [54] can help provide a better account of classical performances.

We briefly compare with other quantum approaches. The first quantum Hopfield network [36] encodes the data in the basis states of an exponentially large quantum state (instead of using the amplitudes) and uses a Grover search to attain quantum speedups for memory recall. Such Grover search speedups are possible in rather generic settings, achieving a performance of about $\sqrt{d}$. In the adiabatic quantum computing framework, quantum Hopfield networks are developed, exploiting the natural connection of the Hopfield network and Ising-like energy functions [37]. The critical quantity for the run time is the spectral gap of the associated Hamiltonian. In many cases, this spectral gap is exponentially small, leading to similar run times as the classical methods. In other cases, when the gap is only polynomially small, exponential speedups may be possible. Reference [35] considers an open quantum system treatment of the Hopfield network and develops the resulting phase diagram. Quantum effects are shown to be included by an effective temperature. Other works [33] have discussed single-electron quantum tunneling in the context of the Hopfield network, which can overcome local energy minima, where actual performance will be determined by the physical implementation. Another work has discussed the potential occurrence of quantum effects in cellular microtubules at low temperatures [34].

Our work uses an exponential encoding of the neuronal information into quantum amplitudes, the gate model of quantum computing, and a setting where quantum phase estimation and matrix inversion can be used for the Hopfield network. As discussed these techniques can lead to a potential performance logarithmic in the number of neurons for specific applications. However, let us emphasize that this analysis does not constitute a comprehensive benchmark of qHop against possible

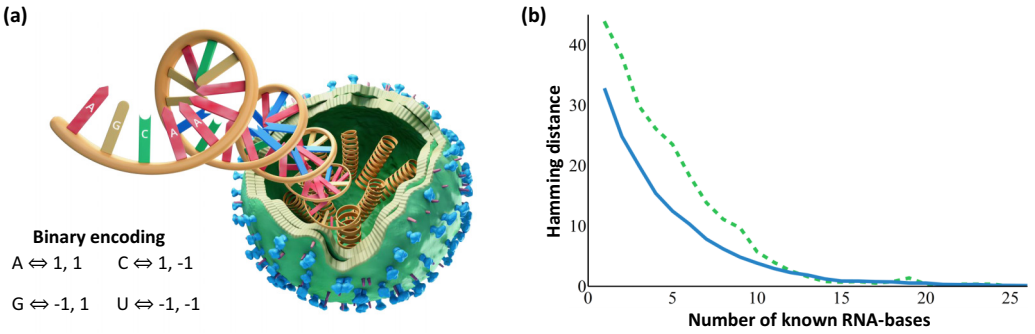

FIG. 2. RNA recognition. (a) The Hopfield network can be used as a content-addressable memory system for RNA recognition (data from Ref. [55]). We encode 50 RNA bases of the $M = 8$ strands originating from the H1N1 influenza A virus in W, and then run the Hopfield network on partial information from a limited number of randomly selected RNA bases from the first strand. (b) The result of operating the Hopfield network on this example using the standard classical approach (dotted line) and the matrix-inversion-based approach (solid line). The resultant Hamming distance to the true data is averaged over 1000 repetitions for varying amounts of partial information.

classical and quantum approaches to running the Hopfield network.

X. APPLICATION

Here we outline an application of the Hopfield network in RNA sequencing. Consider the H1N1 strain of the influenza A virus, which has eight RNA segments that code for different functions in the virus. The segments are composed of a string of RNA bases: A, C, G, and U. Each segment can in turn be converted to a double-sized binary string, as shown in Fig. 2, which can be stored in the weighting matrix of a Hopfield network. Suppose that we are provided with partial information on a new RNA sequence and would like to verify whether it belongs to the H1N1 virus. For example, our sequence could be from a recently collected sample originating in an area with a new influenza outbreak. This scenario can be addressed by resorting to the Hopfield network.

We use this setting as a motivation for our numerics presented in Fig. 2, which contains a comparison of the performance of the standard classical approach to operating the Hopfield network with our matrix-inversion-based approach. Here, we store the first 50 RNA bases from each of the eight segments of the influenza A H1N1 strain (i.e., so that $d = 100$, $M = 8$) in the weighting matrix W using the Hebbian learning rule (data from Ref. [55]). For this small example, the weighting matrix is filled to classical capacity, i.e., $M = 5 \approx d/(2 \log d)$ [48], so that imperfect recoveries are more easily identified. Note the discussion on the M dependency after Eq. (19). We then generate incomplete data from the first segment of H1N1 by randomly selecting $l/2$ RNA bases for $l/2 \in \{1, 2, \ldots, 50\}$. Both approaches to operating the Hopfield network are then implemented to reconstruct the full activation pattern, with the Hamming distance measured between the result and the original pattern. This is averaged over 1000 repetitions of random choices of $l/2$ RNA bases, with the resultant data plotted in Fig. 2. We see that both the conventional approach to the Hopfield network and the matrix-inversion-based approach have comparable performances, with each able to recover the input segment for a suitably large $l/2$. Yet, by using qHop to perform the matrix-inversion-based approach, we could operate with a run time logarithmic in the system dimension and hence increase the dimension far beyond $d = 100$ (see the previous section and Fig. 1 for a comparison of run times). Note that for the matrix-inversion-based approach, we set $\gamma = 1$ to guarantee a local minimum since $\|W\| \approx 0.185$. Moreover, the objective is to classify whether the collected sample is the H1N1 virus. For the quantum version of the Hopfield network, this can be achieved by performing a swap test with the target state $|\tilde{x}\rangle$ set to correspond to the encoded H1N1 virus.

XI. DISCUSSION

Quantum effects have a profound potential to yield advancements in machine learning over the coming decade. We have presented a quantum implementation (qHop) for the Hopfield network that encodes an exponential number of neurons within the amplitudes of only a polynomially large register of qubits. This complements alternative encodings focusing on a one-to-one correspondence between neurons and qubits. Crucially, the learning and operation steps of the quantum Hopfield network can be exponentially quicker in run time when compared to classical approaches. We have also introduced a method of training a quantum neural network via quantum Hebbian learning (qHeb).

As with many quantum algorithms, the efficient operation of qHop is subject to some important considerations. One must first be able to efficiently read the classical initialization data of the neural network into our quantum device, which can be achieved using efficient pure state preparation techniques [42] or qRAM [41], or alternatively by directly using the output of a quantum device. Next, it must be possible to operate efficiently qHeb, and matrix inversion [8]. This relies on efficient Hamiltonian simulation of the system matrix, which we show to be possible by resorting to sparse Hamiltonian simulation techniques [38] and density matrix exponentiation [23,24]. When using qHeb as a learning method, we obtain a linear dependence on the number of training examples, which affects the capacity of the quantum Hopfield network. This linear dependence may be avoided by using sparse simulations or density matrix simulation directly on the

qHop matrix. The matrix inversion algorithm then outputs the inverse only on a well-conditioned subspace with (absolute) eigenvalues larger than a chosen fixed value μ whose inverse controls the algorithm efficiency. It is crucial to note that classical sparse matrix inversion algorithms also have a similar efficiency dependence on μ. Finally, it must be possible to efficiently access the output of qHop, which is a pure quantum state representing a continuous-valued neuronal activation pattern. Since quantum state tomography is typically resource intensive, one can instead access global information such as the fidelity with previously trained activation patterns or the expectation values with respect to observables.

We have introduced the subroutine qHeb, which adapts the standard Hebbian learning approach [29] to the quantum setting, an addition to studies on quantum learning. Our subroutine relies on the important observation that the weight matrix W describing a neural network can be alternatively represented by a mixed quantum state (or more generally, a Hamiltonian). Using density matrix exponentiation [23,24], this quantum state can then be used operationally for the extraction of, e.g., eigenvalues and eigenvectors of the weight matrix. We have shown that quantum Hebbian learning can be implemented by performing a sequential imprinting of memory patterns, represented as pure quantum states, onto a register of memory qubits. Although introduced here within the context of the quantum Hopfield network, quantum Hebbian learning can be of wider interest as a quantum subroutine within other quantum neural networks.

Our findings, along with other work [14,25–27,56–62], including quantum Hopfield networks [33–35,63], contribute to the goal of developing a practical quantum neural network. The approach we use encodes an exponential number of neurons into a polynomial number of qubits. We have discussed a specific neural network, the Hopfield network, which is a content-addressable memory system. As an application, we have shown how the matrix-inversion-based Hopfield network can be utilized for identifying genetic segments of RNA in viruses. Future developments may focus on the nature of quantum neural networks themselves, identifying entirely new applications that harness purely quantum properties without being based upon previous classical networks. The natural next step to benefit from the fruits of quantum neural networks, and developments in quantum machine learning more generally, is to implement these algorithms on near-term quantum devices.

ACKNOWLEDGMENTS

We thank Juan Miguel Arrazola, Mayank Bhatia, and Nathan Killoran for fruitful discussions. S.L. was supported by OSD/ARO under the Blue Sky Initiative.

APPENDIX A: PERTURBED DATA

Instead of incomplete data, we here discuss the problem of correcting perturbed data. The new element is $\boldsymbol{x}^{(\mathrm{new})}$ and its perturbed version is $\boldsymbol{x}^{(\mathrm{pert})}$. We pose the classical problem by including the closeness to the perturbed version via an l2-norm constraint. Instead of the Lagrangian Eq. (10), we have the error function

$$\mathscr{E} := -\frac{1}{2}\boldsymbol{x}^{\intercal} W \boldsymbol{x} + \boldsymbol{\theta}^{\intercal}\boldsymbol{x} + \frac{\beta}{2}|\boldsymbol{x} - \boldsymbol{x}^{(\mathrm{pert})}|_2^2 + \frac{\gamma}{2}\boldsymbol{x}^{\intercal}\boldsymbol{x}, \quad \text{(A1)}$$

where instead of Lagrange multipliers we use the regularization parameter β. The first-order criterion now is

$$\frac{\partial \mathscr{E}}{\partial \boldsymbol{x}} = ((\gamma+\beta)\mathbb{I}_d - W)\boldsymbol{x} + \boldsymbol{\theta} - \beta \boldsymbol{x}^{(\mathrm{pert})} \stackrel{!}{=} 0. \quad \text{(A2)}$$

This leads to the matrix inversion problem for finding $\boldsymbol{x}$ as

$$((\gamma+\beta)\mathbb{I}_d - W)\boldsymbol{x} = \beta \boldsymbol{x}^{(\mathrm{pert})} - \boldsymbol{\theta}. \quad \text{(A3)}$$

The resulting quantum algorithm is similar, even slightly simpler, than the method discussed in the main part of this work, and not further discussed here.

APPENDIX B: CONSTRAINED MINIMIZATION OF THE ENERGY FUNCTION

Here we show that the result of the constrained optimization outlined in the Hopfield network section of the main text is necessarily a local minimum. Suppose that we are to optimize a real-valued scalar function $f(\boldsymbol{x})$ of a vector $\mathbf{x} \in \mathbb{R}^d$ subject to $l < d$ constraints composed into a real-valued vector function $\mathbf{g}(\mathbf{x}) = \mathbf{0}$. The corresponding Lagrangian is $\mathscr{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \boldsymbol{\lambda}^{\intercal}\mathbf{g}(\mathbf{x})$ with Lagrange multiplier vector $\boldsymbol{\lambda}$. Optimization can be achieved by identifying vectors $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{\lambda}})$ satisfying $\partial_{\boldsymbol{x}}\mathscr{L} = \mathbf{0}$ and $\partial_{\boldsymbol{\lambda}}\mathscr{L} = \mathbf{0}$. To classify these optimal vectors we must consider the $((l+d)\times(l+d))$-dimensional bordered Hessian matrix [64]

$$\mathscr{H}(\boldsymbol{x}, \boldsymbol{\lambda}) := \begin{pmatrix} 0_l & \nabla \mathbf{g}(\mathbf{x}) \\ \nabla \mathbf{g}(\mathbf{x})^{\intercal} & \frac{\partial^2 \mathscr{L}}{\partial \boldsymbol{x}^2} \end{pmatrix}. \quad \text{(B1)}$$

In particular, $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{\lambda}})$ is a local minimum if

$$(-1)^l \det(\mathscr{H}_k(\boldsymbol{x}, \boldsymbol{\lambda})) > 0 \quad \text{(B2)}$$

for all $k \in \{2l+1, 2l+2, \ldots, l+d\}$, where $\mathscr{H}_k(\boldsymbol{x}, \boldsymbol{\lambda})$ is the kth-order leading principle submatrix of $\mathscr{H}(\boldsymbol{x}, \boldsymbol{\lambda})$, composed of taking the first k rows and the first k columns.

We now show that this condition is satisfied when $f(\boldsymbol{x})$ is the energy E in Eq. (9) of the main text and $\mathbf{g}(\mathbf{x}) = P\boldsymbol{x} - \boldsymbol{x}^{(\mathrm{inc})}$. The bordered Hessian matrix is then

$$\mathscr{H}(\boldsymbol{x}, \boldsymbol{\lambda}) = \begin{pmatrix} 0_l & -\tilde{P} \\ -\tilde{P}^{\intercal} & \gamma\mathbb{I}_d - W \end{pmatrix}, \quad \text{(B3)}$$

with $\tilde{P}$ a rectangular $(l \times d)$-dimensional matrix of rows of unit vectors e_i for $i \in \mathcal{L}$, or equivalently the projector P with all zero rows removed. We note that in our setting the bordered Hessian matrix is in fact independent of $\boldsymbol{x}$ and $\boldsymbol{\lambda}$, meaning that we can classify any extremum found. We therefore herein drop the following brackets around $\mathscr{H}$. Now consider the leading principle minor $\mathscr{H}_k$ for any $k \in \{2l+1, 2l+2, \ldots, l+d\}$, given by

$$\mathscr{H}_k = \begin{pmatrix} 0_l & -\tilde{P}_{l\times(k-l)} \\ -(\tilde{P}_{l\times(k-l)})^{\intercal} & (\gamma\mathbb{I}_d - W)_{(k-l)} \end{pmatrix}, \quad \text{(B4)}$$

with $\tilde{P}_{l\times(k-l)}$ composed of the first $k-l$ columns of $\tilde{P}$ and $(\gamma\mathbb{I}_d - W)_{(k-l)}$ the $(k-l)$th-order leading principal submatrix of $\gamma\mathbb{I}_d - W$.

Let us consider $\gamma > \|W\|$ with $\|W\|$ the largest eigenvalue of W, so that $\gamma \mathbb{I}_d - W > 0$. Sylvester's criterion tells us that $(\gamma \mathbb{I}_d - W)_{(k-l)} > 0$ and is hence invertible. Using the Schur complement, we have that

$$\det(\mathscr{H}_k) = (-1)^l \det((\gamma \mathbb{I}_d - W)_{(k-l)}) \times \det(\tilde{P}_{l\times(k-l)}((\gamma \mathbb{I}_d - W)_{(k-l)})^{-1}(\tilde{P}_{l\times(k-l)})^{\intercal}), \tag{B5}$$

with X^{-1} the inverse of X. On the other hand, we know that $((\gamma \mathbb{I}_d - W)_{(k-l)})^{-1} > 0$. The action of $\tilde{P}_{l\times(k-l)}((\gamma \mathbb{I}_d - W)_{(k-l)})^{-1}(\tilde{P}_{l\times(k-l)})^{\intercal}$ is to select an lth-order principal minor of $((\gamma \mathbb{I}_d - W)_{(k-l)})^{-1}$. It is a well-known result in linear algebra that any principle minor of a positive-definite matrix is itself positive definite [65], so that we know $\tilde{P}_{l\times(k-l)}((\gamma \mathbb{I}_d - W)_{(k-l)})^{-1}(\tilde{P}_{l\times(k-l)})^{\intercal} > 0$ for any k. Since the determinant of a positive-definite matrix is positive, we hence know that

$$\det((\gamma \mathbb{I}_d - W)_{(k-l)}) > 0,$$
$$\det(\tilde{P}_{l\times(k-l)}((\gamma \mathbb{I}_d - W)_{(k-l)})^{-1}(\tilde{P}_{l\times(k-l)})^{\intercal}) > 0.$$

This means that the sign of $\det(\mathscr{H}_k)$ is given by $(-1)^l$, and that overall

$$(-1)^l \det(\mathscr{H}_k) > 0, \tag{B6}$$

satisfying the condition for a minimum given above.

APPENDIX C: SETTING THE REGULARIZATION PARAMETER

From the previous section, we see that it is necessary to introduce the regularization parameter to provide a sufficient condition that our constrained optimization reaches a local minimum. From the perspective of machine learning, the regularization parameter also functions to penalize large values of $|\boldsymbol{x}|_2$ in the minimization to prevent overfitting. In Fig. 3, following the example outlined in the main text, we plot the average Hamming distance between the reconstructed pattern (using our matrix-inversion-based approach with discretized postprocessing) and the original pattern for increasing values of regularization parameter and a constant number of known neurons, $l = 50$. Here, the average Hamming distance drops off dramatically to zero for a sufficiently high regularization parameter $\gamma > \|W\| \approx 0.185$. However, if one chooses an arbitrary large γ then this adds a polynomial run time onto qHop (see the efficiency discussion in the main text). In the numerics of the main part, we set $\gamma = 1$.

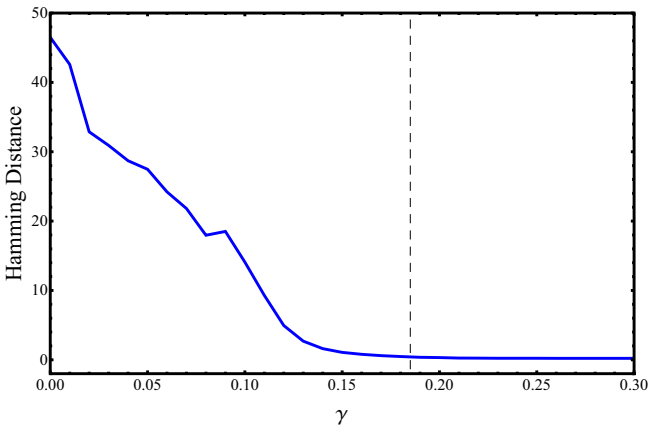

FIG. 3. The average Hamming distance between one of the reconstructed memory patterns and the original when $l = 50$ neurons are known *a priori*, given as a function of the regularization parameter γ. The maximum eigenvalue $||W|| \approx 0.185$ of W, which γ must exceed to guarantee a local minimum, is shown as the vertical dashed line. Note that no increase of the Hamming distance is observed for $\gamma > 0.3$.

APPENDIX D: SETTING THE VALUE OF μ

Our algorithm finds the inverse of

$$\tilde{A} := \sum_{j:\,|\mu_j(A)|\geqslant\mu} \mu_j(A)|v_j(A)\rangle\langle v_j(A)|, \tag{D1}$$

[see Eq. (17) of the main text for comparison to A.] It holds that $\tilde{A}^{-1}$ is equal to the pseudoinverse A^{-1} whenever μ does not exceed the smallest nonzero singular value $|\mu_{\min}|$ of A. Otherwise, $\tilde{A}^{-1}|w\rangle$ approximates $A^{-1}|w\rangle$ to an error

$$\eta := |\tilde{A}^{-1}|w\rangle - A^{-1}|w\rangle|_2. \tag{D2}$$

From Eq. (19) of the main text, it can be seen that qHop maintains the polylogarithmic efficiency in run time whenever μ is such that $1/\mu \in O(\mathrm{poly}(\log d))$. Hence, for the matrix inversion to be effective, we require A to be such that either (1) $|\mu_{\min}| \geqslant \mu$, with no additional errors in finding the pseudoinverse, or (2) $|\mu_{\min}| < \mu$ but with $\eta \in O(\epsilon)$ so that the error η accumulates in accordance with an overall desired error $O(\epsilon)$.

[1] C. M. Bishop, *Pattern Recognition and Machine Learning* (Springer, Berlin, 2006).

[2] M. A. Nielsen and I. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, U.K., 2002).

[3] M. Schuld, I. Sinayskiy, and F. Petruccione, Contemp. Phys. **56**, 172 (2015).

[4] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Nature (London) **549**, 195 (2017).

[5] C. Ciliberto, M. Herbster, A. D. Ialongo, M. Pontil, A. Rocchetto, S. Severini, and L. Wossnig, Proc. R. Soc. A **474**, 20170551 (2017).

[6] V. Dunjko and H. J. Briegel, Rep. Prog. Phys. **81**, 074001 (2018).

[7] N. Wiebe, D. Braun, and S. Lloyd, Phys. Rev. Lett. **109**, 050505 (2012).

[8] A. W. Harrow, A. Hassidim, and S. Lloyd, Phys. Rev. Lett. **103**, 150502 (2009).

[9] G. Brassard, P. Hoyer, M. Mosca, and A. Tapp, Contemp. Math. **305**, 53 (2002).

[10] L. K. Grover, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing* (ACM, New York, 1996), pp. 212–219.

[11] T. Kadowaki and H. Nishimori, Phys. Rev. E **58**, 5355 (1998).

[12] A. Finnila, M. Gomez, C. Sebenik, C. Stenson, and J. Doll, Chem. Phys. Lett. **219**, 343 (1994).

[13] S. Boixo, T. F. Rønnow, S. V. Isakov, Z. Wang, D. Wecker, D. A. Lidar, J. M. Martinis, and M. Troyer, Nat. Phys. **10**, 218 (2014).

[14] N. Wiebe, A. Kapoor, and K. M. Svore, Quantum Info. Comput. **16**, 541 (2016).

[15] V. Dunjko, J. M. Taylor, and H. J. Briegel, Phys. Rev. Lett. **117**, 130501 (2016).

[16] M. Benedetti, J. Realpe-Gómez, R. Biswas, and A. Perdomo-Ortiz, Phys. Rev. A **94**, 022308 (2016).

[17] J. Romero, J. Olson, and A. Aspuru-Guzik, Quantum Sci. Technol. **2**, 045001 (2017).

[18] M. Schuld, I. Sinayskiy, and F. Petruccione, *Pacific Rim International Conference on Artificial Intelligence* (Springer, Berlin, 2014), pp. 208–220.

[19] Z. Zhao, J. K. Fitzsimons, and J. F. Fitzsimons, arXiv:1512.03929.

[20] L. Wossnig, Z. Zhao, and A. Prakash, Phys. Rev. Lett. **120**, 050502 (2018).

[21] N. Wiebe, A. Kapoor, and K. M. Svore, Quantum Inf. Comput. **15**, 0316 (2015).

[22] P. Rebentrost, M. Mohseni, and S. Lloyd, Phys. Rev. Lett. **113**, 130503 (2014).

[23] S. Lloyd, M. Mohseni, and P. Rebentrost, Nat. Phys. **10**, 631 (2014).

[24] S. Kimmel, C. Y.-Y. Lin, G. H. Low, M. Ozols, and T. J. Yoder, npj Quantum Inf. **3**, 13 (2017).

[25] M. Schuld, I. Sinayskiy, and F. Petruccione, Quantum Inf. Proc. **13**, 2567 (2014).

[26] M. H. Amin, E. Andriyash, J. Rolfe, B. Kulchytskyy, and R. Melko, Phys. Rev. X **8**, 021050 (2018).

[27] M. Benedetti, J. Realpe-Gómez, and A. Perdomo-Ortiz, Quantum Sci. Technol. **3**, 034007 (2018).

[28] S. Hochreiter and J. Schmidhuber, Neural Comput. **9**, 1735 (1997).

[29] D. O. Hebb, *The Organization of Behavior* (Wiley, Hoboken, NJ, 1949).

[30] K.-S. Cheng, J.-S. Lin, and C.-W. Mao, IEEE Trans. Med. Imaging **15**, 560 (1996).

[31] G. E. Hinton, S. Osindero, and Y.-W. Teh, Neural Comput. **18**, 1527 (2006).

[32] Y. Bengio, *Learning Deep Architectures for AI* (Now Publishers, Boston, 2009).

[33] M. Akazawa, E. Tokuda, N. Asahi, and Y. Amemiya, Analog Integr. Circuits Signal Process. **24**, 51 (2000).

[34] E. C. Behrman, K. Gaddam, J. Steck, and S. Skinner, *The Emerging Physics of Consciousness*, edited by J. A. Tuszynski (Springer, Berlin, 2006), pp. 351–370.

[35] P. Rotondo, M. Marcuzzi, J. Garrahan, I. Lesanovsky, and M. Muller, J. Phys. A: Math. Theor. **51**, 115301 (2018).

[36] D. Ventura and T. Martinez, in *Proceedings of the 1998 IEEE International Joint Conference on Neural Networks* (IEEE, New York, 1998), Vol. 1, pp. 509–513.

[37] H. Seddiqi and T. S. Humble, Front. Phys. **22**, 79 (2014).

[38] D. W. Berry, A. M. Childs, and R. Kothari, in *2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)* (IEEE, New York, 2015), pp. 792–809.

[39] W. S. McCulloch and W. Pitts, Bull. Math. Biophys. **5**, 115 (1943).

[40] J. J. Hopfield, Proc. Natl. Acad. Sci. USA **79**, 2554 (1982).

[41] V. Giovannetti, S. Lloyd, and L. Maccone, Phys. Rev. Lett. **100**, 160501 (2008).

[42] A. N. Soklakov and R. Schack, Phys. Rev. A **73**, 012307 (2006).

[43] Z. Zhao, V. Dunjko, J. K. Fitzsimons, P. Rebentrost, and J. F. Fitzsimons, arXiv:1804.00281.

[44] D. Gross, Y.-K. Liu, S. T. Flammia, S. Becker, and J. Eisert, Phys. Rev. Lett. **105**, 150401 (2010).

[45] M. Cramer, M. B. Plenio, S. T. Flammia, D. Gross, S. D. Bartlett, R. Somma, O. Landon-Cardinal, Y.-K. Liu, and D. Poulin, Nat. Commun. **1**, 149 (2010).

[46] D. W. Berry, G. Ahokas, R. Cleve, and B. C. Sanders, Commun. Math. Phys. **270**, 359 (2007).

[47] A. M. Childs, D. Maslov, Y. Nam, N. J. Ross, and Y. Su, Proc. Natl. Acad. Sci. USA **115**, 9456 (2018).

[48] R. McEliece, E. Posner, E. Rodemich, and S. Venkatesh, IEEE Trans. Inf. Theory **33**, 461 (1987).

[49] I. Kerenidis and A. Prakash, in *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, edited by C. H. Papadimitriou, Leibniz International Proceedings in Informatics (LIPIcs) (Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017), Vol. 67, pp. 49:1–49:21.

[50] H. Buhrman, R. Cleve, J. Watrous, and R. de Wolf, Phys. Rev. Lett. **87**, 167902 (2001).

[51] J. R. Shewchuk *et al.*, Carnegie-Mellon University, 1994 (unpublished).

[52] E. Ising, Z. Phys. A **31**, 253 (1925).

[53] P. J. Van Laarhoven and E. H. Aarts, *Simulated Annealing: Theory and Applications* (Springer, Berlin, 1987), pp. 7–15.

[54] M. Opper and D. Saad, *Advanced Mean Field Methods: Theory and Practice* (MIT Press, Cambridge, MA, 2001).

[55] H. Zaraket, R. Saito, Y. Suzuki, T. Baranovich, C. Dapat, I. Caperig-Dapat, and H. Suzuki, J. Clin. Microbiol. **48**, 1085 (2010).

[56] S. Kak, Inf. Sci. **83**, 143 (1995).

[57] G. Bonnell and G. Papini, Int. J. Theor. Phys. **36**, 2855 (1997).

[58] M. Altaisky, arXiv:quant-ph/0107012.

[59] A. Narayanan and T. Menneer, Inf. Sci. **128**, 231 (2000).

[60] A. A. Ezhov and D. Ventura, *Future Directions for Intelligent Systems and Information Sciences*, Studies in Fuzziness and Soft Computing Vol. 45 (Springer, Berlin, 2000), p. 213.

[61] K. H. Wan, O. Dahlsten, H. Kristjánsson, R. Gardner, and M. S. Kim, npj Quantum Inf. **3**, 36 (2017).

[62] M. Kieferova and N. Wiebe, Phys. Rev. A **96**, 062327 (2017).

[63] E. C. Behrman, L. Nash, J. E. Steck, V. Chandrashekar, and S. R. Skinner, Inf. Sci. **128**, 257 (2000).

[64] A. Ghosh, D. Wassermann, and R. Deriche, *Information Processing in Medical Imaging* (Springer, Berlin, 2011), pp. 723–734.

[65] G. Shilov, *Linear Algebra* (Dover, New York, 1977).