Neural Belief-Propagation Decoders for Quantum Error-Correcting Codes

Ye-Hua Liu[1,*] and David Poulin[1,2,†]
[1]Département de Physique & Institut Quantique, Université de Sherbrooke, J1K 2R1 Sherbrooke, Québec, Canada
[2]Canadian Institute for Advanced Research, M5G 1Z8 Toronto, Ontario, Canada

(Received 26 November 2018; published 22 May 2019)

Belief-propagation (BP) decoders play a vital role in modern coding theory, but they are not suitable to decode quantum error-correcting codes because of a unique quantum feature called error degeneracy. Inspired by an exact mapping between BP and deep neural networks, we train neural BP decoders for quantum low-density parity-check codes with a loss function tailored to error degeneracy. Training substantially improves the performance of BP decoders for all families of codes we tested and may solve the degeneracy problem which plagues the decoding of quantum low-density parity-check codes.

DOI: 10.1103/PhysRevLett.122.200501

Statistical inference on a graph is an important paradigm in many areas of science, and equivalent heuristic algorithms have been developed by different communities, including the cavity method in statistical physics [1] and the belief-propagation (BP) algorithm in information science [2]. In the latter case, BP is the standard decoding algorithm for low-density parity-check (LDPC) codes [3], which form the backbone of modern coding theory and are widely used in wireless communication [4]. With the growing interest for quantum technologies, quantum generalizations of LDPC codes have been proposed [5–7], but BP was found to be inadequate for their decoding [8] because of error degeneracy, a feature unique to quantum codes. Despite many improvements [8–10] to BP, there is still no accurate decoding algorithm for general quantum LDPC codes. This contrasts with statistical physics where the cavity method has been generalized to the quantum setting with some success [11–14].

Recently, an exact mapping between BP and artificial neural networks has been revealed [15], which implies a general machine-learning strategy to adapt BP to any specific task. In this Letter, we use this strategy for the decoding of quantum LDPC codes. Neural-network-based decoders for quantum error-correcting codes have attracted great interest recently, particularly in the context of topological codes [16–27]. But near optimal (or very fast suboptimal) decoding algorithms are already proposed for these codes [28–31], which exploit their regular lattice structure. In contrast, for quantum LDPC codes, which are defined on random graphs, only recently has a decoding algorithm been found for the special family of expander codes [7,32,33] and the general case remains open. Our main motivation to study this problem is that quantum LDPC codes have the potential of greatly reducing the overhead required to realize robust quantum processors [34,35].

In this Letter, we train neural BP (NBP) decoders for quantum LDPC codes. To guide the learning process, we construct a loss function that takes into account error degeneracy. We present results for the toric code [36], the quantum bicycle code [5], and the quantum hypergraph-product code [6]. Decoding accuracy improves up to 3 orders of magnitude compared with the untrained BP decoder, and the improvement is even more substantial when we ignore detected but uncorrected errors. While we do not completely solve the LDPC decoding problem here, our results suggest that an important step forward was realized, and the strategy could be applied more broadly, for instance in many-body physics. That general strategy consists in training a neural network to solve a quantum problem, with initial conditions corresponding to the BP algorithm that solves the classical counterpart.

LDPC codes.—A linear error-correcting code can be represented by its parity-check matrix H with binary (0 or 1) matrix elements. Codewords $\mathbf{c}$'s satisfying $H\mathbf{c} = \mathbf{0} \bmod 2$. As a result, when an error pattern $\mathbf{e}$ is imposed on the codeword $\mathbf{c} \to \mathbf{c}' = \mathbf{c} + \mathbf{e} \bmod 2$, there will be a measurable syndrome pattern $\mathbf{s} = H\mathbf{c}' = H\mathbf{e} \bmod 2$, which signals the occurrence of the error $\mathbf{e}$. The role of the decoder is to infer the error pattern $\mathbf{e}$ from the measured syndrome pattern $\mathbf{s}$. Classical LDPC codes are error-correcting codes with sparse parity-check matrices, i.e., where the number of 1's in each column and row are bounded by constants independent of the matrix size.

Belief propagation.—The Tanner graph is a graphical representation of the parity-check matrix H, with a set of variable nodes $\{e_v | v = 1, \ldots, n\}$ (containing the error pattern) and a set of check nodes $\{s_c | c = 1, \ldots, m\}$ (containing the syndrome pattern). There is an edge between e_v and s_c if $H_{cv} = 1$. Neighborhoods of variables and checks are defined by $\mathcal{N}(v) = \{c | H_{cv} = 1\}$ and $\mathcal{N}(c) = \{v | H_{cv=1}\}$, respectively.

BP is an iterative algorithm for approximating the average value of each variable node e_v, over all error patterns $\mathbf{e}$'s that are consistent with the given syndrome pattern $\mathbf{s}$ (meaning $H\mathbf{e} = \mathbf{s}$). In performing the average, each error pattern $\mathbf{e}$ is weighted by a probability $P(\mathbf{e}) = \prod_v P(e_v)$, which should accurately model the noise statistics of the physical device carrying the information. Mathematically speaking, BP solves the posterior marginal probability for each variable node $P(e_v = 1|\mathbf{s}) \propto \sum_{\mathbf{e}\backslash e_v} P(\mathbf{s}|\mathbf{e}\backslash e_v, e_v = 1)P(\mathbf{e}\backslash e_v, e_v = 1)$. This goal is achieved by iterating the following simple BP equations:

$$\mu_{v\to c}^{(t+1)} = l_v + \sum_{c'\in\mathcal{N}(v)\backslash c} \mu_{c'\to v}^{(t)}, \tag{1}$$

$$\mu_{c\to v}^{(t+1)} = (-1)^{s_c} 2\tanh^{-1} \prod_{v'\in\mathcal{N}(c)\backslash v} \tanh\frac{\mu_{v'\to c}^{(t)}}{2}, \tag{2}$$

where $l_v = \log[P(e_v = 0)/P(e_v = 1)]$ is the prior log-likelihood ratio for variable e_v and $\mathcal{N}(x)\backslash y$ is the set of all neighbors of x except for y [4]. The initial condition for the iteration is $\mu_{v\to c}^{(t=0)} = 0$, and after T steps (sufficiently long), one stops the iteration and performs the following marginalization for the posterior log-likelihood ratio:

$$\mu_v = l_v + \sum_{c\in\mathcal{N}(v)} \mu_{c\to v}^{(T)}. \tag{3}$$

The posterior marginal probability relates to μ_v according to $\log[P(e_v = 0|\mathbf{s})/P(e_v = 1|\mathbf{s})] = \mu_v$. Equivalently $P(e_v = 1|\mathbf{s}) = \sigma(\mu_v)$ and $P(e_v = 0|\mathbf{s}) = 1 - \sigma(\mu_v)$, where $\sigma(x) = 1/(e^x + 1)$ is the Fermi function (or horizontally flipped sigmoid function). The inferred error pattern maximizes these marginal probabilities; i.e., e_v is inferred to be 0 (1) when μ_v is positive (negative).

Neural belief propagation.—The above iterative procedure can be exactly mapped to a deep neural network, where each neuron represents a message $\mu_{c\to v}$ or $\mu_{v\to c}$ [15]. (See Fig. 1.) This permits generalization of the original BP algorithm by introducing additional "trainable" weights $w_{c'v,vc}^{(t)}$ and $w_{cv,v}^{(T)}$, and trainable biases $b_v^{(t)}$ and $b_v^{(T)}$. Specifically, in this NBP algorithm, Eqs. (1)–(3) are modified to

$$\mu_{v\to c}^{(t+1)} = l_v b_v^{(t)} + \sum_{c'\in\mathcal{N}(v)\backslash c} \mu_{c'\to v}^{(t)} w_{c'v,vc}^{(t)}, \tag{4}$$

$$a(\mu_{c\to v}^{(t+1)}) = i\pi s_c + \sum_{v'\in\mathcal{N}(c)\backslash v} a(\mu_{v'\to c}^{(t)}), \tag{5}$$

$$\mu_v = l_v b_v^{(T)} + \sum_{c\in\mathcal{N}(v)} \mu_{c\to v}^{(T)} w_{cv,v}^{(T)}, \tag{6}$$

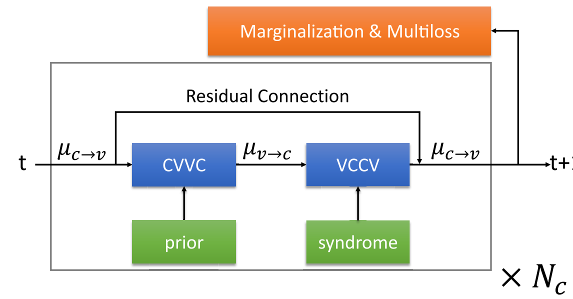

FIG. 1. Schematics of the NBP decoder. The main cycle in the gray box is repeated N_c times. Inside one cycle there are two phases of computation, the $cv \to vc$ and $vc \to cv$ phases which are governed by Eqs. (4) and (5), respectively. The inputs to the neural network are denoted by the green boxes, where the prior and syndrome correspond to $\{l_v\}$ in Eq. (4) and $\{s_c\}$ in Eq. (5), respectively. After each cycle, the set of output $\{\mu_{c\to v}\}$ is marginalized by Eq. (6) and the resulting $\{\mu_v\}$ is sent to the loss function Eq. (8). We also introduce residual connections to facilitate training of deep networks [40]. (See Supplemental Material [41] for details.)

respectively [15,37]. Notice that all equations above have the form of weighted sum plus bias, interleaved with the nonlinear function $a(x) = \log[\tanh(x/2)]$. This is the canonical form of feed-forward neural networks [38]. When setting all newly introduced parameters to 1, these equations became the standard BP equations [39].

To train these weights, one minimizes a carefully designed loss function $\mathcal{L}$ by backpropagating its gradients with respect of all trainable parameters. E.g., biases are updated according to $\Delta b_v^{(t)} = -lr \times \partial\mathcal{L}/\partial b_v^{(t)}$, where lr is the learning rate. For classical codes, one aims for reproducing the whole error pattern exactly, so the natural choice of the loss function is the binary cross entropy function between the inferred error pattern and the true error pattern:

$$\mathcal{L}(\vec{\mu}; \mathbf{e}) = -\sum_v e_v \log\sigma(\mu_v) + (1 - e_v)\log[1 - \sigma(\mu_v)]. \tag{7}$$

Quantum setting.—Quantum noise can be modeled by random Pauli operators I, $\hat{X}$, $\hat{Y}$, and $\hat{Z}$ on the qubits. A convenient way of bookkeeping a N-qubit error uses a $2N$-bit string $\mathbf{e}$ representing the Pauli operator: $\hat{\mathcal{P}}(\mathbf{e}) = \prod_{1\le i\le N} [\hat{X}_i]^{e_i} [\hat{Z}_i]^{e_{i+N}}$. In this representation, two Pauli-string operators $\hat{\mathcal{P}}(\mathbf{a})$ and $\hat{\mathcal{P}}(\mathbf{b})$ commute (anticommute) when $\mathbf{a}^T M\mathbf{b}$ is even (odd), where the symplectic inner product is defined with $M = \begin{pmatrix} & 1_{N\times N} \\ 1_{N\times N} & \end{pmatrix}$. Note that all Pauli-string operators satisfy $\hat{\mathcal{P}}^2 = 1$.

Likewise, the quantum codewords $|\psi\rangle$ are defined by a set of constraints $S_j|\psi\rangle = +|\psi\rangle$ where each stabilizer generator S_j is a Pauli-string operator. For these equations to have a solution, it is necessary for the S_j to mutually commute and to not generate -1 under multiplication. Using the above bookkeeping, we can represent each stabilizer generator S_j by a $2N$-bit string, and assemble these strings as rows of a parity-check matrix H. A quantum LDPC code is one whose parity-check matrix is row sparse and column sparse.

There is a crucial difference between classical and quantum error correction. In the classical case, successful decoding means the inferred error $\mathbf{e}^{\mathrm{inf}}$ is exactly the same as the true error $\mathbf{e}$; while in the quantum case, one only requires the total error $\mathbf{e}^{\mathrm{tot}} = \mathbf{e} + \mathbf{e}^{\mathrm{inf}} \bmod 2$ to belong to the "stabilizer group"—the set of all Pauli-string operators spanned by the rows of H. This is because two Pauli-string operators E and $F = ES_j$ that differ by a stabilizer have identical action on all code states. To test if $\mathbf{e}^{\mathrm{tot}}$ belongs to the stabilizer group, one simply needs to check that it commutes with all the operators that commute with the stabilizers, i.e., that $H^{\perp} M \mathbf{e}^{\mathrm{tot}} = \mathbf{0} \bmod 2$ where $H^{\perp}$ is the matrix that generates the orthogonal complement of H with respect to the symplectic inner product, $HM(H^{\perp})^T = \mathbf{0} \bmod 2$.

The above analysis motivates the design the following loss function tailored for quantum error correction:

$$\mathcal{L}(\vec{\mu}; \mathbf{e}) = \sum_i f\left(\sum_{jk} H^{\perp}_{ij} M_{jk} [e_k + \sigma(\mu_k)]\right). \tag{8}$$

Note the parity check $\mathrm{parity}(x) = x \bmod 2$ is replaced by the continuous and differentiable function $f(x) = |\sin(\pi x/2)|$ to facilitate gradient-based machine-learning techniques. This loss is minimized when the true error $\mathbf{e}$ and the inferred error $\mathbf{e}^{\mathrm{inf}}$ sum to a stabilizer.

The loss function can also be averaged over all NBP cycles $\bar{\mathcal{L}} = (1/N_c) \sum_{i=1}^{N_c} \mathcal{L}(\vec{\mu}^{(i)}; \mathbf{e})$, which requires marginalization after each cycle. In this Letter, we use a variation of this form. See Supplemental Material [41] for more details.

Toric code.—We first study the toric code [36] on an $L \times L$ square lattice, which is a simple and widely studied quantum LDPC code. (See Fig. 3 for the local Tanner graph.) During training, we generate error patterns consisting of independent X and Z errors with physical error rate p_{err}, i.e., $P(e_v = 1) = p_{\mathrm{err}}$ for all v. In each minibatch, 120 error patterns are drawn from six physical error rates that are uniformly distributed in the range $p_{\mathrm{err}} \in [0.01, 0.05]$. After $\sim$10 000 minibatches, we test the performance of the trained decoder. Figure 2 compares the original BP decoder (before training) and the trained NBP decoder at $p_{\mathrm{err}} = 0.01$ for various code sizes. Training significantly enhances decoding accuracy up to 3 orders of magnitude [Fig. 2(a)], and we observe that the training time required for convergence depends weakly on the code size L. (See Supplemental Material [41] for details.)

We can distinguish two types of decoding failure. "Flagged" failures occur when the correction inferred by the decoder does not return the system to the code space—there remains a nontrivial syndrome after decoding. "Unflagged" failures occur when the correction returns the system to the wrong code state. These two contributions to the overall logical error rates are shown in Figs. 2(b) and 2(d), respectively. We observe that training greatly reduces flagged failures at the expense of slightly increasing unflagged failures, and overall there is a significant net decrease of failures. It should be noted that flagged failures are benign because they can be redecoded, using either a more accurate but more expensive decoder (e.g., the minimum-weight perfect matching [42]) or a higher layer

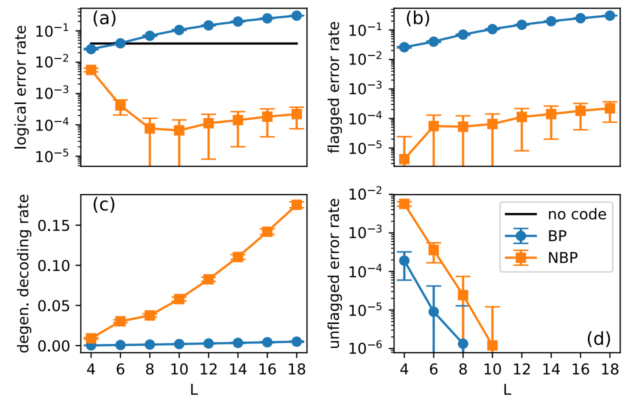

FIG. 2. Training the NBP decoder for the toric code with different code sizes. (a) The logical error rate decreases substantially after training (tested at $p_{\mathrm{err}} = 0.01$). Here, the logical error rate is broken up into two terms in (b) and (d), corresponding to "flagged" and "unflagged" errors, respectively. (See main text for details.) (c) The NBP decoder exploits degeneracy by correctly decoding with an error pattern that is not exactly the same as the true error pattern. Training parameters: $N_c = 25$, $lr = 2 \times 10^{-4}$.

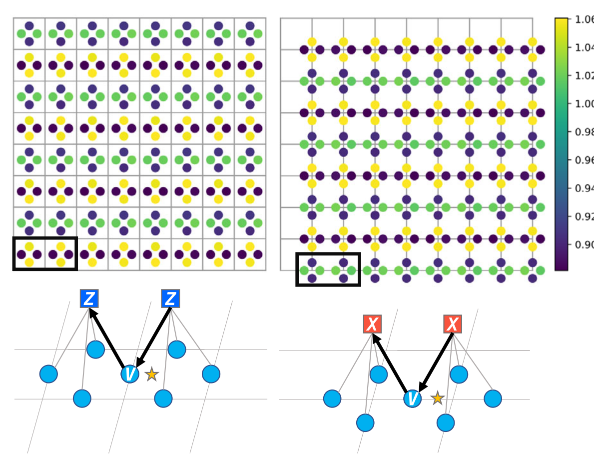

FIG. 3. (Top panels) One of the observed patterns of the learned weights for the toric code. To obtain this clear pattern, we implemented weights sharing (see main text) with $\mathbf{G} = (2i, 2j)$, $i, j = 0, 1, \ldots$. A noisy version of the same pattern is observed when weight sharing is turned off. (Bottom panels) Tanner graph of the toric code and the position of $cvvc$ weights (yellow star). Correspondence to the top panels is marked by the black box.

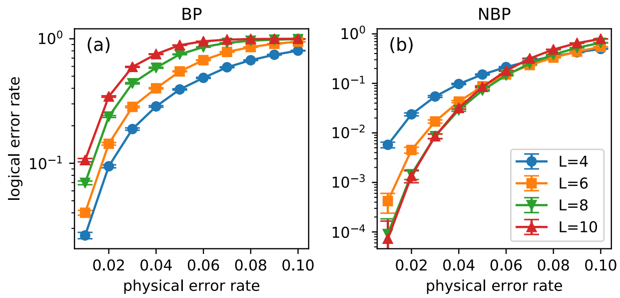

FIG. 4. Evolution of the logical error rate as a function of the physical error rate, for the BP (NBP) decoder before (after) training. (a) Before training the performance of the BP decoder is worse for larger code sizes at all physical error rates. (b) After training, the performance improves substantially for all code sizes at all physical error rates, and the performance curves start to cross each other. This indicates the development of a threshold.

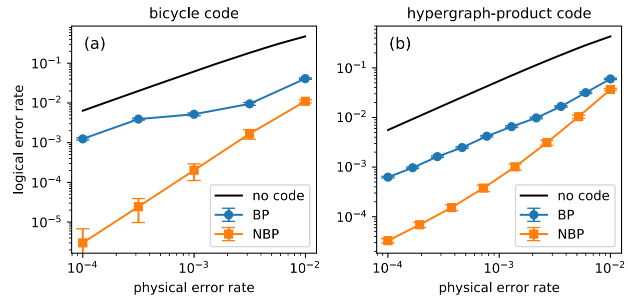

FIG. 5. Training greatly improves the BP decoder for quantum LDPC codes with high rates. (a) Quantum bicycle code with code parameter [[256, 32]] and rate $r = 0.125$. Training parameters: $N_c = 12$, $lr = 1 \times 10^{-4}$. (b) Quantum hypergraph-product code with code parameter [[129, 28]] and rate $r \sim 0.2$. Training parameters: $N_c = 12$, $lr = 1 \times 10^{-4}$.

of code for erasure errors. Such a mixed decoding strategy would combine the speed and flexibility of BP decoder and reliability of a more expensive decoder used on a very small fraction (e.g., 10^{-4}) of instance.

The loss function Eq. (8) takes into account error degeneracy, and we see in Fig. 2(c) that the frequency of successful decoding where the actual and the inferred error differ by a stabilizer increases with the code length. This rate was nearly zero with the untrained decoder (see Supplemental Material [41] for examples of learned stabilizers).

The periodic nature of the toric code inspired us to utilize a weight-sharing technique, where the weights are invariant under lattice translation **G**. We can control the amount of sharing by the size of **G** (similar to the filter size in convolutional neural networks). Figure 3 is a graphical representation of the trained weights, and suggests that symmetry breaking improves BP for quantum codes. We also observe that weights trained on one code size can also increase the performance when applied to codes of different sizes, which implies that the learning is universal or transferable (see Supplemental Material [41] for more details).

Figure 4 shows that significant improvement can be achieved across a range of physical error rates. Using the original BP, increasing the code size leads to worse performance. After training, performance improves with size for sufficiently low error rates, and the trend indicates that further improved training might lead to a BP decoder with a finite threshold.

When the neural network is initialized away from BP, training gets stuck at a much worse local minimum. This illustrates the importance of incorporating domain knowledge (when possible) before using general machine-learning methods as black boxes, which contrasts with prior uses of neural net decoding of the toric code [16,18].

Quantum LDPC codes with high rate.—The toric code encodes a constant number $K = 2$ of qubits in a growing number N of physical qubits, thus achieving a vanishing rate $r = K/N$. We now turn to quantum LDPC codes with constant rates.

The quantum bicycle code [5] is a quantum LDPC code constructed from a random binary vector $\mathbf{A}$ of size $N/2$. First, all cyclic permutations of $\mathbf{A}$ are collected as columns in a matrix C. Then C is concatenated with its transpose to form $H_0 = [C, C^T]$, from which $K/2$ rows are chosen randomly and removed. After these constructions, H_0 is a self-dual matrix (meaning $H_0 H_0^T = \mathbf{0} \bmod 2$) of size $(N-K)/2 \times N$. The final parity-check matrix for the quantum bicycle code is $H = \binom{H_0 \quad}{\quad H_0}$. The sparsity of this matrix can be controlled by the number of nonzero elements in $\mathbf{A}$. Training the NBP decoder for a quantum bicycle code with $N = 256$, $K = 32$, and $\sum_i A_i = 8$ improves the accuracy up to 3 orders of magnitude [Fig. 5(a)].

The quantum hypergraph-product code [6] is constructed from two classical codes with parity-check matrices $[H_1]_{m_1 \times n_1}$ and $[H_2]_{m_2 \times n_2}$. The following products are constructed $H_X = [H_1 \otimes I_{n_2 \times m_2}, I_{m_1 \times n_1} \otimes H_2^T]$ and $H_Z = [I_{n_1 \times m_1} \otimes H_2, H_1^T \otimes I_{m_2 \times n_2}]$, and the parity-check matrix of the quantum code follows $H = \binom{H_X \quad}{\quad H_Z}$, which performs $m = m_1 n_2 + n_1 m_2$ checks on $n = m_1 m_2 + n_1 n_2$ qubits. In this Letter, we study a hypergraph-product code, for which H_1 and H_2 are the classical $[7, 4, 3]$ and $[15, 7, 5]$ Bose-Chaudhuri-Hocquenghem codes, respectively. This code has rate $r = 28/129 \sim 0.2$. Training the NBP decoder for this code improves the accuracy up to one order of magnitude [Fig. 5(b)].

Conclusions.—We significantly improved the belief-propagation decoders for quantum LDPC codes by training them as deep neural networks. Our results on the toric code, the quantum bicycle code, and the quantum hypergraph-product code all show orders of magnitude of enhancement in decoding accuracy. The original belief propagation is known to have bad performance for quantum error-correcting codes [8]. On the other hand, training a neural decoder with general architecture has been reported to be hard for large codes [20,43]. Our results indicate that combining the general framework of machine learning and the specific domain knowledge of quantum error correction is a promising approach, when neither works well individually.

The significance of this result is supported by the tremendous success of BP with classical LDPC codes [4], and the fact that quantum LDPC codes promise a low-overhead fault-tolerant quantum computation architecture [34]. In addition, our techniques could be adapted to uses of BP in other quantum many-body problems, such as improving the quantum cavity method [11–14].

Y. H. L. is thankful for helpful discussions with Pavithran Iyer, Anirudh Krishna, Xin Li, Alex Rigby, and Colin Trout. Special thanks go to Liang Jiang and Stefan Krastanov, who shared similar ideas. This research was undertaken thanks in part to funding from the Canada First Research Excellence Fund. Computations were made on the supercomputer Helios managed by Calcul Québec and Compute Canada. The operation of this supercomputer is funded by the Canada Foundation for Innovation (CFI), the ministère de l'Économie, de la science et de l'innovation du Québec (MESI), and the Fonds de recherche du Québec–Nature et technologies (FRQ-NT). We used TensorFlow [44] to build and train neural belief-propagation decoders.

[*]Yehua.Liu@USherbrooke.ca
[†]David.Poulin@USherbrooke.ca

[1] M. Mezard, G. Parisi, and M. Virasoro, *Spin Glass Theory and Beyond*, World Scientific Lecture Notes in Physics Vol. 9 (World Scientific, Singapore, 1986).

[2] J. Pearl, Reverend Bayes on inference engines: A distributed hierarchical approach, Proc. Second Natl. Conf. Artif. Intell. **AAAI-82**, 133 (1982).

[3] R. G. Gallager, Low-density parity-check codes, IEEE Trans. Inf. Theory **8**, 21 (1962).

[4] T. Richardson and R. Urbanke, *Modern Coding Theory* (Cambridge University Press, Cambridge, England, 2008).

[5] D. J. C. MacKay, G. Mitchison, and P. L. McFadden, Sparse-graph codes for quantum error correction, IEEE Trans. Inf. Theory **50**, 2315 (2004).

[6] J.-P. Tillich and G. Zemor, Quantum LDPC codes with positive rate and minimum distance proportional to the square root of the blocklength, IEEE Trans. Inf. Theory **60**, 1193 (2014).

[7] A. Leverrier, J.-P. Tillich, and G. Zemor, Quantum expander codes, in *2015 IEEE 56th Annual Symposium on Foundations of Computer Science* (IEEE, Berkeley, 2015), Vol. 2015-Decem, pp. 810–824.

[8] D. Poulin and Y. Chung, On the iterative decoding of sparse quantum codes, QIC **8**, 987 (2008).

[9] Y.-J. Wang, B. C. Sanders, B.-m. Bai, and X.-m. Wang, Enhanced feedback iterative decoding of sparse quantum codes, IEEE Trans. Inf. Theory **58**, 1231 (2012).

[10] Z. Babar, P. Botsinis, D. Alanis, S. X. Ng, and L. Hanzo, Fifteen years of quantum LDPC coding and improved decoding strategies, IEEE Access **3**, 2492 (2015).

[11] M. B. Hastings, Quantum belief propagation: An algorithm for thermal quantum systems, Phys. Rev. B **76**, 201102(R) (2007).

[12] D. Poulin and E. Bilgin, Belief propagation algorithm for computing correlation functions in finite-temperature quantum many-body systems on loopy graphs, Phys. Rev. A **77**, 052318 (2008).

[13] C. Laumann, A. Scardicchio, and S. L. Sondhi, Cavity method for quantum spin glasses on the Bethe lattice, Phys. Rev. B **78**, 134424 (2008).

[14] D. Poulin and M. B. Hastings, Markov Entropy Decomposition: A Variational Dual for Quantum Belief Propagation, Phys. Rev. Lett. **106**, 080403 (2011).

[15] E. Nachmani, Y. Be'ery, and D. Burshtein, Learning to decode linear codes using deep learning, in *2016 54th Annual Allerton Conference on Communication, Control, and Computing* (IEEE, Allerton, 2016), pp. 341–346.

[16] G. Torlai and R. G. Melko, Neural Decoder for Topological Codes, Phys. Rev. Lett. **119**, 030501 (2017).

[17] P. Baireuther, T. E. O'Brien, B. Tarasinski, and C. W. J. Beenakker, Machine-learning-assisted correction of correlated qubit errors in a topological code, Quantum **2**, 48 (2018).

[18] S. Krastanov and L. Jiang, Deep neural network probabilistic decoder for stabilizer codes, Sci. Rep. **7**, 11003 (2017).

[19] S. Varsamopoulos, B. Criger, and K. Bertels, Decoding small surface codes with feedforward neural networks, Quantum Sci. Technol. **3**, 015004 (2018).

[20] N. Maskara, A. Kubica, and T. Jochym-O'Connor, Advantages of versatile neural-network decoding for topological codes, arXiv:1802.08680.

[21] C. Chamberland and P. Ronagh, Deep neural decoders for near term fault-tolerant experiments, Quantum Sci. Technol. **3**, 044002 (2018).

[22] T. Fösel, P. Tighineanu, T. Weiss, and F. Marquardt, Reinforcement Learning with Neural Networks for Quantum Feedback, Phys. Rev. X **8**, 031084 (2018).

[23] A. Davaasuren, Y. Suzuki, K. Fujii, and M. Koashi, General framework for constructing fast and near-optimal machine-learning-based decoder of the topological stabilizer codes, arXiv:1801.04377.

[24] P. Baireuther, M. D. Caio, B. Criger, C. W. J. Beenakker, and T. E. O'Brien, Neural network decoder for topological color codes with circuit level noise, New J. Phys. **21**, 013003 (2019).

[25] N. P. Breuckmann and X. Ni, Scalable neural network decoders for higher dimensional quantum codes, Quantum **2**, 68 (2017).

[26] X. Ni, Neural network decoders for large-distance 2D toric codes, arXiv:1809.06640.

[27] R. Sweke, M. S. Kesselring, E. P. L. van Nieuwenburg, and J. Eisert, Reinforcement learning decoders for fault-tolerant quantum computation, arXiv:1810.07207.

[28] S. Bravyi, M. Suchara, and A. Vargo, Efficient algorithms for maximum likelihood decoding in the surface code, Phys. Rev. A **90**, 032326 (2014).

[29] A. S. Darmawan and D. Poulin, Linear-time general decoding algorithm for the surface code, Phys. Rev. E **97**, 051302 (R) (2018).

[30] N. Delfosse and N. H. Nickerson, Almost-linear time decoding algorithm for topological codes, arXiv: 1709.06218.

[31] G. Duclos-Cianci and D. Poulin, Fast Decoders for Topological Quantum Codes, Phys. Rev. Lett. **104**, 050504 (2010).

[32] O. Fawzi, A. Grospellier, and A. Leverrier, Efficient decoding of random errors for quantum expander codes, arXiv:1711.08351.

[33] A. Grospellier and A. Krishna, Numerical study of hypergraph product codes, arXiv:1810.03681.

[34] D. Gottesman, Fault-tolerant quantum computation with constant overhead, Quantum Inf. Comput. **14**, 1338 (2014).

[35] O. Fawzi, A. Grospellier, and A. Leverrier, Constant overhead quantum fault-tolerance with quantum expander codes, arXiv:1808.03821.

[36] A. Kitaev, Anyons in an exactly solved model and beyond, Ann. Phys. (Amsterdam) **321**, 2 (2006).

[37] H. Wymeersch, F. Penna, and V. Savic, Uniformly reweighted belief propagation: A factor graph approach, in *Proceedings of the 2011 International Symposium on Information Theory* (IEEE, Saint-Petersburg, 2011), Vol. 2, pp. 2000–2004.

[38] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, Cambridge, 2016).

[39] For numerical stability, argument of $\tanh^{-1}(\cdot)$ is truncated to $(-1+\epsilon, 1-\epsilon)$ and we choose $\epsilon = 10^{-4}$. Decreasing ϵ does not change the main conclusion of the Letter.

[40] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, arXiv:1512.03385.

[41] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevLett.122.200501 for implementation details and additional data.

[42] J. Edmonds, Paths, trees and flowers, Can. J. Math. **17**, 449 (1965).

[43] T. Gruber, S. Cammerer, J. Hoydis, and S. ten Brink, On deep learning-based channel decoding, arXiv:1701.07738.

[44] M. Abadi *et al.*, TensorFlow: Large-scale machine learning on heterogeneous distributed systems, arXiv:1603.04467.