

Topological Quantum Compiling with Reinforcement Learning

Yuan-Hang Zhang^{1,2}, Pei-Lin Zheng^{3,4}, Yi Zhang^{3,4,*} and Dong-Ling Deng^{1,5,†}


¹*Center for Quantum Information, IIIS, Tsinghua University, Beijing 100084, People's Republic of China*

²*Department of Physics, University of California, San Diego, California 92093, USA*

³*International Center for Quantum Materials, Peking University, Beijing 100871, China*

⁴*School of Physics, Peking University, Beijing 100871, China*

⁵*Shanghai Qi Zhi Institute, 41th Floor, AI Tower, No. 701 Yunjin Road, Xuhui District, Shanghai 200232, China*

 (Received 10 May 2020; revised 8 September 2020; accepted 14 September 2020; published 19 October 2020)

Quantum compiling, a process that decomposes the quantum algorithm into a series of hardware-compatible commands or elementary gates, is of fundamental importance for quantum computing. We introduce an efficient algorithm based on deep reinforcement learning that compiles an arbitrary single-qubit gate into a sequence of elementary gates from a finite universal set. It generates near-optimal gate sequences with given accuracy and is generally applicable to various scenarios, independent of the hardware-feasible universal set and free from using ancillary qubits. For concreteness, we apply this algorithm to the case of topological compiling of Fibonacci anyons and obtain near-optimal braiding sequences for arbitrary single-qubit unitaries. Our algorithm may carry over to other challenging quantum discrete problems, thus opening up a new avenue for intriguing applications of deep learning in quantum physics.

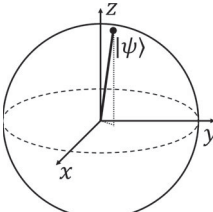
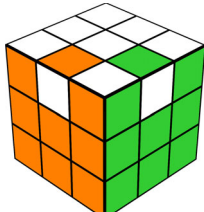
DOI: [10.1103/PhysRevLett.125.170501](https://doi.org/10.1103/PhysRevLett.125.170501)

To efficiently decompose unitaries into a sequence of elementary hardware-compatible quantum gates as short as possible is a crucial problem in a variety of quantum-information-processing tasks, such as quantum computing [1] and quantum digital simulations [2]. This problem becomes especially relevant for the noisy intermediate-scale quantum devices [3], where the depth of the quantum circuits might be limited due to the inaccuracy of the possible elementary gates and quantum decoherence. A number of notable algorithms have been proposed to compile single-qubit unitaries [4–17]. For instance, the Solovay-Kitaev algorithm runs in $O[\log^{2.71}(1/\epsilon)]$ time and can output a sequence of $O[\log^{3.97}(1/\epsilon)]$ elementary gates that approximate the targeted unitary to precision ϵ [1,4]. Other algorithms either exploit the specific structure of the Clifford + T gate set [12–16] or utilize ancillary qubits [5,6] to further reduce the running time and length of the desired gate sequences. Each of these algorithms bears its pros and cons, and the choice depends on the specific problem. Here, inspired by the similarity between quantum compiling and solving Rubik's cube (see Table I), we introduce a novel algorithm based on deep reinforcement learning, which compiles single-qubit unitaries efficiently and is generally applicable to different scenarios (see Fig. 1 for an illustration).

Machine learning, especially deep learning, has achieved dramatic success in a broad range of artificial intelligence applications, ranging from image and speech recognition to self-driving cars [24,25]. The interplay between machine learning and quantum physics has led to an emergent

research frontier of quantum machine learning, which has attracted tremendous attention [26–29]. Quantum learning algorithms with potential exponential advantages have been proposed, and machine learning techniques have also been invoked in various applications in quantum physics, including representing quantum many-body states [30,31], quantum state tomography [32,33], nonlocality detection [34], and learning phases of matter [35–45], etc. In this work, we introduce deep reinforcement learning [46], which has been exploited to build AlphaGo [47] (a computer program of Go that defeated the world's best players) and more recently DeepCubeA that solves the Rubik's cube—a classic combinatorial puzzle that posed unique challenges for artificial intelligence [48] to the task of quantum compiling. We observe that compiling unitaries to a sequence of elementary gates is analogous to finding a sequence of basic moves that solves the Rubik's cube (see Table I). Since unitaries are invertible, finding a gate sequence approximating a target unitary U is equivalent to finding a gate sequence that “restores” U back to identity. In this way, the identity matrix becomes our target state (corresponding to the solved cube), and the unitary U is the initial state (corresponding to a scrambled cube). Both problems have several discretized, noncommuting operations; the goal of both problems is to find the shortest sequences available; for a state seemingly close to the targeted one, the actual number of required operations may still be surprisingly large. Similar to the fact that DeepCubeA can solve an arbitrarily scrambled cube in a near-optimal fashion [48], our algorithm can efficiently

TABLE I. Comparison between quantum compiling and solving the Rubik's cube shows that these two seemingly irrelevant problems have a lot in common. The shown cube has only two misplaced cubelets, yet it takes at least 16 steps [23] to solve. Similarly, the shown quantum state $|\psi\rangle$ is close to the target state $|0\rangle$, but with a discretized universal gate set, it may still take many steps to transform into $|0\rangle$.

System		
Initial state	The unitary to be approximated	The scrambled cube
Target state	The identity matrix	The solved cube
Basic move	A gate from the universal set	Rotation of one face

compile an arbitrary unitary into a near-optimal sequence of elementary gates.

The algorithm.—First, we introduce our general algorithm; later, we will apply it to the case of topological compiling of Fibonacci anyons as a concrete example. In previous reinforcement learning algorithms such as deep Q learning [47,49], a function approximator such as a deep neural network (DNN) represents a reward function defined on all states, which dictates the strategy to maximize the reward and performs the actions step by step. Then, the resulting experiences are added to the regression to optimize the DNN further, and so on and so forth. However, when such an algorithm is directly applied to bring an arbitrary quantum state to a specific target, it faces immediate failure: With a large state space, discretized actions at each step, a single target, and giving the reward only extremely close to the target, the reward may never be

received at all, making it almost impossible to train a valid reward function.

To resolve this issue, we start from the target state instead and perform backward search operations, similar to the value iteration algorithm [50]. The cost-to-go function $J(s)$ is defined as the minimum cost for a state s to reach the target state within the designated precision, represented approximately by a DNN. During training, we update the cost-to-go function according to [48]

$$J'(s) = \min_a \{g(s, a) + J[S(s, a)]\}, \quad (1)$$

where $S(s, a)$ is the state obtained after applying the action a to the state s and $g(s, a)$ is the corresponding cost. $J(s_0) = 0$ for the target state s_0 , and $J(s)$ for other states can be computed with Eq. (1) successively. In practice, Eq. (1) uses the DNN itself for target updating, which may lead to instabilities. Therefore, we use two neural networks during training [48,49]: a policy network that is constantly being trained and a target network that estimates of the target value $J'(s)$ for training and updates to the policy network only periodically.

To enhance the search performance and derive the shortest sequence possible, we further complement the cost-to-go function $J(s)$ with a weighted A* search algorithm [51,52]. We define an evaluation function $f(s)$ from the initial state s_i to the target state s_0 via an intermediate state s :

$$f(s) = \lambda G(s) + J(s), \quad (2)$$

where $G(s)$ is the actual cost from the initial state s_i to the current state s . $\lambda \in [0, 1]$ is a weighting factor, and smaller λ reduces the number of states evaluated and alleviates the difficulty of a large state space at the expense of potentially longer paths [52]. During the search, we start with a set of the intermediate states $\{s\}$ with only the initial state s_i ; iteratively, we pick the state s in $\{s\}$ with the minimum

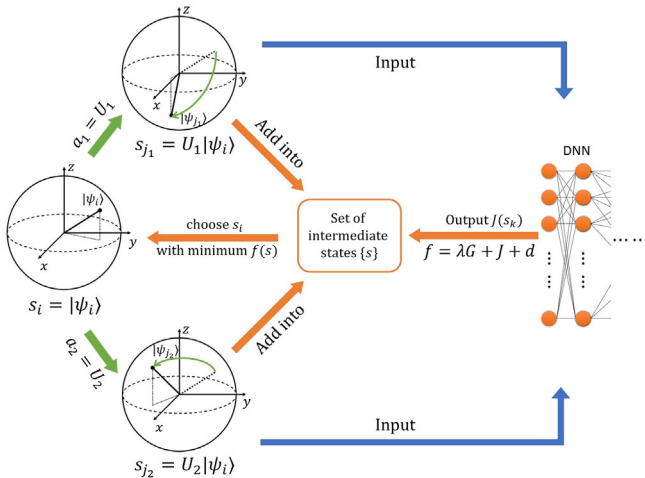


FIG. 1. A schematic illustration of the algorithm: During the search, we start from the initial state $s_i = |\psi_i\rangle$, then iteratively pick the state with the minimum evaluation function $f(s)$, and evaluate all its successors with the DNN until we reach the target state (see [18]).

$f(s)$ and replace it with its successors $S(s, a)$ (if they are not already in or have not previously been in $\{s\}$)—see Fig. 1; once a state with a distance less than a designated termination accuracy ϵ_T from the target state s_0 is present in $\{s\}$, we have obtained the desired sequence between s_i and s_0 within the desired accuracy threshold.

We also make several additional modifications to the weighted A^* search algorithm to better fit our quantum compiling problem. First, we introduce a maximum searching depth D_{\max} , beyond which the search terminates and returns the best state found so far. This cutoff resolves the possible nonconvergence induced by the search along a discrete graph on a continuous state space. Second, it is natural for the DNN to generalize the cost-to-go function $J(s)$ to states never present in training. Sometimes, such a state is mistaken for a small $J(s)$ [e.g., $J(s) \sim 1.5$], although its actual distance from the target state is considerably farther away, and the weighted A^* searches are stuck there. To handle this problem, we introduce a decimal-penalty term to the evaluation function $f(s) = \lambda G(s) + J(s) + d(s)$:

$$d(s) = \gamma \frac{\{J(s) - \text{round}[J(s)]\}^2}{J(s)}, \quad (3)$$

where γ is a constant tuning parameter. $d(s)$ put preferences on states used to train the DNN with near-integer $J(s)$ over states whose $J(s)$ values containing decimal parts and are likely estimations and interpolations.

Without loss of generality, we apply our algorithm to topological compiling with Fibonacci anyons, which are quasiparticle excitations of topological states that obey non-Abelian braiding statistics [53]. Unlike Majorana bound states [54], whose braiding gives only elementary gates in the Clifford group unless additional multistep protocols are incorporated [55], Fibonacci anyons are the simplest non-Abelian quasiparticles that enable universal topological quantum computation [56,57] by braiding alone [58]. They are theoretically predicted to exist in the $\nu = 12/5$ fractional quantum Hall liquid [59] and rotating Bose condensates [60], as well as quantum spin systems [61,62]. The only nontrivial fusion rule for Fibonacci anyons reads $\tau \times \tau = \mathbf{I} + \tau$, where \mathbf{I} and τ denote the vacuum and the Fibonacci anyon, respectively. We encode logical qubits into triplets of anyons with total topological charge one [58]: $|0_L\rangle = |[(\bullet, \bullet)_{\tau}, \bullet]_{\tau}\rangle$ and $|1_L\rangle = |[(\bullet, \bullet)_{\tau}, \bullet]_{\mathbf{I}}\rangle$, and neglect the noncomputational state $|NC\rangle = |[(\bullet, \bullet)_{\tau}, \bullet]_{\mathbf{I}}\rangle$, since we mainly focus on braidings within a single logical qubit and the leakage error is not relevant in this case. Based on this encoding scheme, the two elementary single-qubit gates correspond to the braidings of two Fibonacci anyons are σ_1 and σ_2 as shown in Fig. 2(a), which form a universal set for single-qubit unitaries.

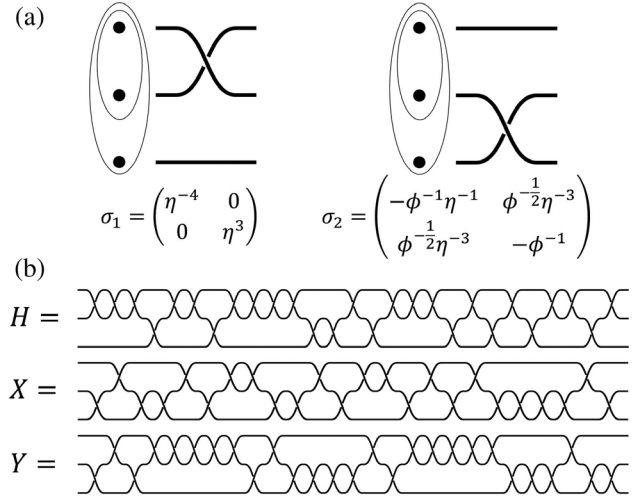


FIG. 2. (a) The two elementary gates by braiding Fibonacci anyons. Logical qubits are encoded into triplets of anyons (enclosed in the ovals), and time flows from left to right. Here, $\eta = e^{i\pi/5}$ and $\phi = (\sqrt{5} + 1)/2$ [53]. (b) The approximate braiding sequences obtained by the reinforcement-learning-based algorithm for the Hadamard gate (H), the Pauli σ^x gate (X), and the Pauli σ^y gate (Y), respectively. The quaternion distances between the braiding sequences and their corresponding targets are 4.4×10^{-3} , 2.4×10^{-3} , and 2.3×10^{-3} , respectively. After the DNN is trained, the algorithm takes only a couple of seconds to output each of the sequences.

In the literature, topological compiling with Fibonacci anyons has been extensively studied, and different algorithms have been proposed [63–69]. Notable examples include the quantum hashing algorithm [67], which runs in $O[\log(1/\epsilon)]$ time and outputs a sequence of length $O[\log^2(1/\epsilon)]$, and the probabilistically polynomial algorithm [68], which runs in $O[\text{polylog}(1/\epsilon)]$ time on average and outputs an asymptotically depth-optimal sequence of length $O[\log(1/\epsilon)]$. Here we apply the introduced reinforcement-learning algorithm. To measure the accuracy of the output sequence, we use the quaternion distance [70]: $d(q_b, q_t) = \sqrt{1 - \langle q_b, q_t \rangle^2}$, where q_b and q_t are the unit quaternions corresponding to the unitary from the braiding sequence and the target unitary, respectively, and $\langle q_b, q_t \rangle$ denotes their inner product. We employ a DNN with the state s as the input, two fully connected hidden layers, and six residue blocks [71], followed by one output neuron representing the approximate cost-to-go function $J(s)$. We train this DNN via PyTorch routines with randomly sampled sequences whose lengths are shorter than a given constant [18]. The training process takes about two days running on an NVIDIA TITAN V GPU. Without loss of generality, we set $g(s, a) = 1$ for all gates in Eq. (1). In situations where certain elementary gates are harder to implement or the cost is state dependent, we can simply adjust $g(s, a)$ and retrain the DNN. The optimal values for parameters λ , γ in the evaluation function $f(s)$ and the maximum searching depth D_{\max} are determined by a grid

search (see [18]). Unless noted otherwise, we set $\lambda = 1$, $\gamma = 400$, and $D_{\max} = 100$.

In Fig. 2(b), we show the braiding sequences derived by our reinforcement-learning algorithm to approximate the Hadamard gate, the Pauli σ^x gate, and the Pauli σ^y gate (up to a trivial global phase). Compared to the brute-force searches in previous works [63,65,66], we have achieved comparable accuracy and sequence length but with much less computational time. Also, we randomly generate 1000 unitaries in $SU(2)$ and use the reinforcement-learning algorithm to generate their corresponding braiding sequences. The algorithm can efficiently output the desired sequence for any of these unitaries with a running time of less than a couple of seconds on a single GPU. The typical average length of these sequences is ~ 24.79 , and the average precision is $\sim 3.1 \times 10^{-3}$ (see [18]), on par with the results from the brute-force search. To compare our algorithm with the Solovay-Kitaev algorithm, we apply the latter to the same 1000 unitaries and find the obtained braiding sequences are typically 10 times longer.

To further analyze the time complexity and the length complexity as the scalings of the precision inverse $1/\epsilon$, we explicitly control the approximation accuracy by terminating the weighted A^* search once a state with a distance less than ϵ_T from the target state s_0 is found. To ensure that most instances reach the desired accuracy ϵ_T , here we set the maximum searching depth to a larger value $D_{\max} = 1000$. Figure 3(a) shows the averaged actual accuracy $\bar{\epsilon}$ as a function of ϵ_T . When ϵ_T is large, it is easier to find a sequence with a precision smaller than ϵ_T , and the search terminates before hitting the depth limit D_{\max} ; thus, $\bar{\epsilon}$ is noticeably smaller than ϵ_T . As ϵ_T becomes smaller, the constraint of limited searching depth becomes dominant, and more and more target unitaries may require the weighted A^* searches with a depth larger than the given $D_{\max} = 1000$ to attain an accuracy smaller than ϵ_T , as shown in Fig. 3(b). We plot the averaged searching depth \bar{D} as a function of $\bar{\epsilon}$ in Fig. 3(c). From this figure, when $\bar{\epsilon}$ is large, \bar{D} scales logarithmically with $1/\bar{\epsilon}$: $\bar{D} \sim 6.56 \log(1/\bar{\epsilon})$, leading to a nearly linear time complexity—the search time scales as $\bar{\tau} \sim 0.274 \log(1/\bar{\epsilon})$; see Fig. 3(d). As $\bar{\epsilon}$ decreases further, however, the searching depth and time start to increase dramatically. This is likely due to the relatively limited sequence length (no larger than $D \sim 40$) during the training; thus, the DNN has not yet learned enough information for approximating unitaries with higher precision. One way to improve the performance of the algorithm for smaller $\bar{\epsilon}$ is to increase D_{\max} . Also, we plot the average length \bar{L} of the braiding sequences obtained by different algorithms as a function of $\bar{\epsilon}$ in Fig. 4. From this figure, \bar{L} scales as $\bar{L} \sim \log^{1.6}(1/\bar{\epsilon})$ for our reinforcement-learning algorithm, which is slightly worse than the scaling for the brute-force approach but notably better than that for the Solovay-Kitaev algorithm. We note that one may further improve the performance of the

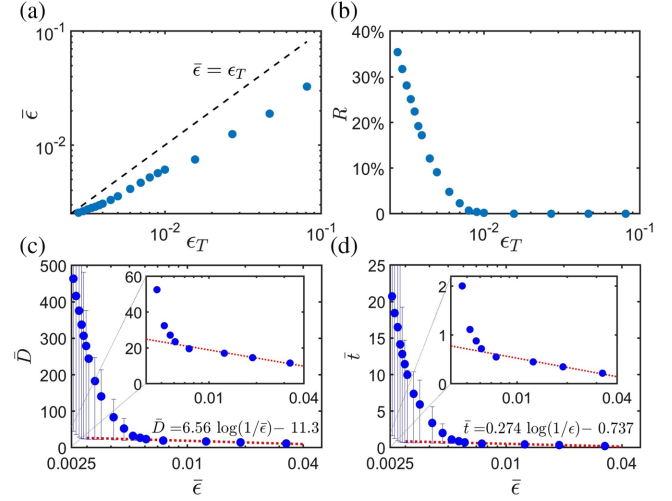


FIG. 3. (a) The averaged actual accuracy ($\bar{\epsilon}$) versus the termination accuracy (ϵ_T). Since the distribution of the accuracy has a long tail, we employ the typical average $\bar{\epsilon} = \exp(\overline{\log \epsilon_i})$. Here, the weighted A^* search terminates once a state with an accuracy smaller than ϵ_T is found or the searching depth exceeds $D_{\max} = 1000$. (b) The ratio (R) of target unitaries that require a depth larger than D_{\max} to attain an accuracy smaller than ϵ_T . In general, we need longer sequences for higher accuracy (c) The average searching depth (\bar{D}) as a function of the average actual accuracy ($\bar{\epsilon}$). The red dotted line is a logarithmic fitting of \bar{D} into $\log(1/\bar{\epsilon})$. (d) The average searching time $\bar{\tau}$ versus the average actual accuracy $\bar{\epsilon}$. The red dotted line is a logarithmic fitting of $\bar{\tau}$ into $\log(1/\bar{\epsilon})$. In these figures, each data point represents an average on the solutions of 1000 random unitaries.

reinforcement-learning algorithm in the above example, through increasing the size of the DNN, the length of the braiding sequences in the training set, or the searching depth when generating sequences, etc. In fact, we used a much smaller DNN in this work than that for AlphaGo [47] and only a single GPU.

The reinforcement learning algorithm can also compile two- or multiqubit gates, with enlarged state space (target unitary matrices) and action space (gates in the universal set) accordingly, which demands a larger DNN and, inevitably, increases the cost for its training. For simplicity, here we consider the compiling of arbitrary two-qubit gates for demonstration. The action space involves braiding six Fibonacci anyons within the 87-dimensional Hilbert space, much larger than the case for single-qubit gates [63]. Alternatively, we can decompose an arbitrary two-qubit gate into seven single-qubit gates and three controlled-NOT (CNOT) gates analytically and optimally [72]. In turn, the CNOT gate can be decomposed into a single-qubit rotation and a controlled- iX gate, whose braiding sequence is available [63,65]. Finally, our reinforcement-learning algorithm can compile the component single-qubit unitaries. Indeed, this buildup decomposes arbitrary two-qubit gates into braiding sequences with notably better performance than the Solovay-Kitaev algorithm [18].

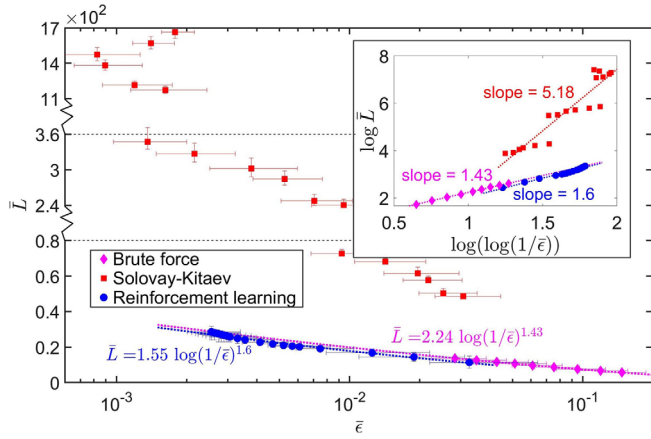


FIG. 4. Comparison between different algorithms on the length complexity of the generated gate sequences. Each data point represents an average on the solutions of 1000 random unitaries, and the error bars show the interval between the lower 25th percentile and the upper 75th percentile. Note the breaks in the vertical scale. The inset is a log-log plot for \bar{L} versus $\log(1/\bar{\epsilon})$, and the dotted lines are linear fits of the data.

Discussion and conclusion.—In experiments, it is common that elementary gates cost differently, and reducing the use of the expensive ones in compiling is of crucial importance for applications in quantum computing. Notably, each elementary gate’s cost can be naturally incorporated into our reinforcement-learning approach by adjusting the cost function $[g(s, a)$ in Eq. (1)]—another striking advantage of the proposed approach over traditional algorithms. Moreover, our approach carries over straightforwardly to other quantum control problems [73] as well.

In summary, we have introduced a reinforcement-learning-based quantum compiling algorithm to decompose an arbitrary unitary into a sequence of elementary gates from a finite universal set. This algorithm uses no ancillary qubit or group-theory relevance and is generally applicable to various scenarios regardless of the choice of universal gate sets. It generates near-optimal gate sequences that approximate arbitrary unitaries to given accuracy in an efficient fashion. To illustrate how the algorithm works, we have also applied it to topological compiling with Fibonacci anyons. Our results build a new connection between reinforcement learning and quantum compiling, which would benefit future studies in both areas.

The source code for this work can be found in Ref. [74].

We acknowledge insightful discussions with Lei Wang. Y. Z. and P.-L. Z. are supported by the start-up grant from Peking University. Y.-H. Z. and D.-L. D. acknowledge the start-up fund from Tsinghua University (Grant. No. 53330300320). D.-L. D. also acknowledges additional support from the Shanghai Qi Zhi Institute.

*frankzhangyi@gmail.com

†dldeng@tsinghua.edu.cn

- [1] M. A. Nielsen and I. Chuang, Quantum computation and quantum information, *Am. J. Phys.* **70**, 558 (2002).
- [2] I. M. Georgescu, S. Ashhab, and F. Nori, Quantum simulation, *Rev. Mod. Phys.* **86**, 153 (2014).
- [3] J. Preskill, Quantum computing in the NISQ era and beyond, *Quantum* **2**, 79 (2018).
- [4] C. M. Dawson and M. A. Nielsen, The solovay-kitaev algorithm, [arXiv:quant-ph/0505030](https://arxiv.org/abs/quant-ph/0505030).
- [5] A. Y. Kitaev, A. H. Shen, and M. N. Vyalyi, *Classical and Quantum Computation* (American Mathematical Society, Providence, 2002).
- [6] N. C. Jones, J. D. Whitfield, P. L. McMahon, M.-H. Yung, R. V. Meter, A. Aspuru-Guzik, and Y. Yamamoto, Faster quantum chemistry simulation on fault-tolerant quantum computers, *New J. Phys.* **14**, 115023 (2012).
- [7] A. G. Fowler, Constructing arbitrary steane code single logical qubit fault-tolerant gates, *Quantum Inf. Comput.* **11**, 867 (2011), <https://dl.acm.org/doi/10.5555/2230936.2230946>.
- [8] A. Bocharov and K. M. Svore, Resource-Optimal Single-Qubit Quantum Circuits, *Phys. Rev. Lett.* **109**, 190501 (2012).
- [9] A. Bocharov, Y. Gurevich, and K. M. Svore, Efficient decomposition of single-qubit gates into v basis circuits, *Phys. Rev. A* **88**, 012313 (2013).
- [10] T. T. Pham, R. Van Meter, and C. Horsman, Optimization of the Solovay-Kitaev algorithm, *Phys. Rev. A* **87**, 052332 (2013).
- [11] Y. Zhiyenbayev, V. M. Akulin, and A. Mandilara, Quantum compiling with diffusive sets of gates, *Phys. Rev. A* **98**, 012325 (2018).
- [12] V. Kliuchnikov, D. Maslov, and M. Mosca, Asymptotically Optimal Approximation of Single Qubit Unitaries by Clifford and t Circuits Using a Constant Number of Ancillary Qubits, *Phys. Rev. Lett.* **110**, 190502 (2013).
- [13] P. Selinger, Quantum circuits of t -depth one, *Phys. Rev. A* **87**, 042302 (2013).
- [14] D. Gosset, V. Kliuchnikov, M. Mosca, and V. Russo, An algorithm for the t -count, *Quantum Inf. Comput.* **14**, 1261 (2014), <https://dl.acm.org/doi/10.5555/2685179.2685180>.
- [15] R. N. J. and P. Selinger, Optimal ancilla-free clifford + t approximation of z -rotations, *Quantum Inf. Comput.* **16**, 901 (2016), <https://dl.acm.org/doi/10.5555/3179330.3179331>.
- [16] L. E. Heyfron and E. T. Campbell, An efficient quantum compiler that reduces t count, *Quantum Sci. Technol.* **4**, 015004 (2018).
- [17] M. S. Alam, Quantum logic gate synthesis as a Markov decision process, [arXiv:1912.12002](https://arxiv.org/abs/1912.12002).
- [18] See Supplemental Material at <http://link.aps.org/supplemental/10.1103/PhysRevLett.125.170501> for details on the structure of the neural networks, the grid search, the pseudocodes for the algorithm, and more numerical data, which includes Refs. [19–22].
- [19] A. L. Maas, A. Y. Hannun, and A. Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in *Proceedings of ICML* (2013), Vol. 30, p. 3, <https://api.semanticscholar.org/CorpusID:16489696>.

- [20] S. Ioffe and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, [arXiv:1502.03167](#).
- [21] B. Kraus and J. I. Cirac, Optimal creation of entanglement using a two-qubit gate, *Phys. Rev. A* **63**, 062309 (2001).
- [22] R. R. Tucci, An introduction to Cartan's Kak decomposition for qc programmers, [arXiv:quant-ph/0507171](#).
- [23] T. Rokicki, H. Kociemba, M. Davidson, and J. Dethridge, The diameter of the rubik's cube group is twenty, *SIAM Rev.* **56**, 645 (2014).
- [24] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, *Nature (London)* **521**, 436 (2015).
- [25] M. Jordan and T. Mitchell, Machine learning: Trends, perspectives, and prospects, *Science* **349**, 255 (2015).
- [26] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, *Nature (London)* **549**, 195 (2017).
- [27] S. D. Sarma, D.-L. Deng, and L.-M. Duan, Machine learning meets quantum physics, *Phys. Today* **3**, No. **72**, 48 (2019).
- [28] V. Dunjko and H. J. Briegel, Machine learning & artificial intelligence in the quantum domain: A review of recent progress, *Rep. Prog. Phys.* **81**, 074001 (2018).
- [29] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, Machine learning and the physical sciences, *Rev. Mod. Phys.* **91**, 045002 (2019).
- [30] G. Carleo and M. Troyer, Solving the quantum many-body problem with artificial neural networks, *Science* **355**, 602 (2017).
- [31] X. Gao, Z. Zhang, and L.-M. Duan, An efficient quantum algorithm for generative machine learning, [arXiv:1711.02038](#).
- [32] G. Torlai, G. Mazzola, J. Carrasquilla, M. Troyer, R. Melko, and G. Carleo, Neural-network quantum state tomography, *Nat. Phys.* **14**, 447 (2018).
- [33] J. Carrasquilla, G. Torlai, R. G. Melko, and L. Aolita, Reconstructing quantum states with generative models, *Nat. Mach. Intell.* **1**, 155 (2019).
- [34] D.-L. Deng, Machine Learning Detection of Bell Non-locality in Quantum Many-Body Systems, *Phys. Rev. Lett.* **120**, 240402 (2018).
- [35] Y. Zhang and E.-A. Kim, Quantum Loop Topography for Machine Learning, *Phys. Rev. Lett.* **118**, 216401 (2017).
- [36] J. Carrasquilla and R. G. Melko, Machine learning phases of matter, *Nat. Phys.* **13**, 431 (2017).
- [37] E. P. van Nieuwenburg, Y.-H. Liu, and S. D. Huber, Learning phase transitions by confusion, *Nat. Phys.* **13**, 435 (2017).
- [38] L. Wang, Discovering phase transitions with unsupervised learning, *Phys. Rev. B* **94**, 195105 (2016).
- [39] P. Broecker, J. Carrasquilla, R. G. Melko, and S. Trebst, Machine learning quantum phases of matter beyond the fermion sign problem, *Sci. Rep.* **7**, 8823 (2017).
- [40] K. Ch'ng, J. Carrasquilla, R. G. Melko, and E. Khatami, Machine Learning Phases of Strongly Correlated Fermions, *Phys. Rev. X* **7**, 031038 (2017).
- [41] Y. Zhang, R. G. Melko, and E.-A. Kim, Machine learning z_2 quantum spin liquids with quasiparticle statistics, *Phys. Rev. B* **96**, 245119 (2017).
- [42] S. J. Wetzel, Unsupervised learning of phase transitions: From principal component analysis to variational auto-encoders, *Phys. Rev. E* **96**, 022140 (2017).
- [43] W. Hu, R. R. P. Singh, and R. T. Scalettar, Discovering phases, phase transitions, and crossovers through unsupervised machine learning: A critical examination, *Phys. Rev. E* **95**, 062122 (2017).
- [44] Y. Zhang, A. Mesaros, K. Fujita, S. Edkins, M. Hamidian, K. Ch'ng, H. Eisaki, S. Uchida, J. S. Davis, E. Khatami *et al.*, Machine learning in electronic-quantum-matter imaging experiments, *Nature (London)* **570**, 484 (2019).
- [45] W. Lian, S.-T. Wang, S. Lu, Y. Huang, F. Wang, X. Yuan, W. Zhang, X. Ouyang, X. Wang, X. Huang, L. He, X. Chang, D.-L. Deng, and L. Duan, Machine Learning Topological Phases with a Solid-State Quantum Simulator, *Phys. Rev. Lett.* **122**, 210503 (2019).
- [46] M. Sewak, *Deep Reinforcement Learning: Frontiers of Artificial Intelligence* (Springer, New York, 2019).
- [47] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, Mastering the game of go with deep neural networks and tree search, *Nature (London)* **529**, 484 (2016).
- [48] F. Agostinelli, S. McAleer, A. Shmakov, and P. Baldi, Solving the rubiks cube with deep reinforcement learning and search, *Nat. Mach. Intell.* **1**, 356 (2019).
- [49] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, Human-level control through deep reinforcement learning, *Nature (London)* **518**, 529 (2015).
- [50] M. L. Puterman and M. C. Shin, Modified policy iteration algorithms for discounted Markov decision problems, *Manage. Sci.* **24**, 1127 (1978).
- [51] P. E. Hart, N. J. Nilsson, and B. Raphael, A formal basis for the heuristic determination of minimum cost paths, *IEEE Trans. Syst. Sci. Cybern.* **4**, 100 (1968).
- [52] I. Pohl, Heuristic search viewed as path finding in a graph, *Artif. Intell.* **1**, 193 (1970).
- [53] C. Nayak, S. H. Simon, A. Stern, M. Freedman, and S. D. Sarma, Non-Abelian anyons and topological quantum computation, *Rev. Mod. Phys.* **80**, 1083 (2008).
- [54] S. D. Sarma, M. Freedman, and C. Nayak, Majorana zero modes and topological quantum computation, *npj Quantum Inf.* **1**, 15001 (2015).
- [55] T. Karzig, Y. Oreg, G. Refael, and M. H. Freedman, Universal Geometric Path to a Robust Majorana Magic Gate, *Phys. Rev. X* **6**, 031019 (2016).
- [56] A. Y. Kitaev, Fault-tolerant quantum computation by anyons, *Ann. Phys. (Amsterdam)* **303**, 2 (2003).
- [57] A. Kitaev, Anyons in an exactly solved model and beyond, *Ann. Phys. (Amsterdam)* **321**, 2 (2006).
- [58] M. H. Freedman, M. Larsen, and Z. Wang, A modular functor which is universal for quantum computation, *Commun. Math. Phys.* **227**, 605 (2002).
- [59] J. S. Xia, W. Pan, C. L. Vicente, E. D. Adams, N. S. Sullivan, H. L. Stormer, D. C. Tsui, L. N. Pfeiffer, K. W. Baldwin, and K. W. West, Electron Correlation in the Second Landau Level: A Competition between Many

- Nearly Degenerate Quantum Phases, *Phys. Rev. Lett.* **93**, 176809 (2004).
- [60] N. R. Cooper, N. K. Wilkin, and J. M. F. Gunn, Quantum Phases of Vortices in Rotating Bose-Einstein Condensates, *Phys. Rev. Lett.* **87**, 120405 (2001).
- [61] M. Freedman, C. Nayak, K. Shtengel, K. Walker, and Z. Wang, A class of p,t-invariant topological phases of interacting electrons, *Ann. Phys. (Amsterdam)* **310**, 428 (2004).
- [62] P. Fendley and E. Fradkin, Realizing non-Abelian statistics in time-reversal-invariant systems, *Phys. Rev. B* **72**, 024412 (2005).
- [63] N. E. Bonesteel, L. Hormozi, G. Zikos, and S. H. Simon, Braid Topologies for Quantum Computation, *Phys. Rev. Lett.* **95**, 140503 (2005).
- [64] H. Xu and X. Wan, Constructing functional braids for low-leakage topological quantum computing, *Phys. Rev. A* **78**, 042325 (2008).
- [65] L. Hormozi, G. Zikos, N. E. Bonesteel, and S. H. Simon, Topological quantum compiling, *Phys. Rev. B* **75**, 165310 (2007).
- [66] D.-L. Deng, C. Wu, J.-L. Chen, and C. H. Oh, Fault-Tolerant Greenberger-Horne-Zeilinger Paradox Based on Non-Abelian Anyons, *Phys. Rev. Lett.* **105**, 060402 (2010).
- [67] M. Burrello, H. Xu, G. Mussardo, and X. Wan, Topological Quantum Hashing with the Icosahedral Group, *Phys. Rev. Lett.* **104**, 160502 (2010).
- [68] V. Kliuchnikov, A. Bocharov, and K. M. Svore, Asymptotically Optimal Topological Quantum Compiling, *Phys. Rev. Lett.* **112**, 140504 (2014).
- [69] C. Carnahan, D. Zeuch, and N. E. Bonesteel, Systematically generated two-qubit anyon braids, *Phys. Rev. A* **93**, 052328 (2016).
- [70] D. Q. Huynh, Metrics for 3d rotations: Comparison and analysis, *J. Math. Imaging Vision* **35**, 155 (2009).
- [71] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, New York, 2016), pp. 770–778.
- [72] F. Vatan and C. Williams, Optimal quantum circuits for general two-qubit gates, *Phys. Rev. A* **69**, 032315 (2004).
- [73] D. Dong and I. R. Petersen, Quantum control theory and applications: a survey, *IET Control Theor. Appl.* **4**, 2651 (2010).
- [74] https://github.com/yuanhangzhang98/ml_quantum_compiling.