


## Scalable Neural Decoder for Topological Surface Codes

Kai Meinerz<sup>1</sup>, Chae-Yeun Park, and Simon Trebst<sup>1</sup>

*Institute for Theoretical Physics, University of Cologne, 50937 Cologne, Germany*

 (Received 5 February 2021; revised 21 December 2021; accepted 2 February 2022; published 24 February 2022)

With the advent of noisy intermediate-scale quantum (NISQ) devices, practical quantum computing has seemingly come into reach. However, to go beyond proof-of-principle calculations, the current processing architectures will need to scale up to larger quantum circuits which will require fast and scalable algorithms for quantum error correction. Here, we present a neural network based decoder that, for a family of stabilizer codes subject to depolarizing noise and syndrome measurement errors, is scalable to tens of thousands of qubits (in contrast to other recent machine learning inspired decoders) and exhibits faster decoding times than the state-of-the-art union find decoder for a wide range of error rates (down to 1%). The key innovation is to autodecode error syndromes on small scales by shifting a preprocessing window over the underlying code, akin to a convolutional neural network in pattern recognition approaches. We show that such a preprocessing step allows to effectively reduce the error rate by up to 2 orders of magnitude in practical applications and, by detecting correlation effects, shifts the actual error threshold up to fifteen percent higher than the threshold of conventional error correction algorithms such as union find or minimum weight perfect matching, even in the presence of measurement errors. An *in situ* implementation of such a machine learning-assisted quantum error correction will be a decisive step to push the entanglement frontier beyond the NISQ horizon.

DOI: [10.1103/PhysRevLett.128.080505](https://doi.org/10.1103/PhysRevLett.128.080505)

**Introduction.**—In quantum computing, recent years have seen a paradigm shift which has pivoted experimental road maps from building devices of a few pristine qubits toward the realization of circuit architectures of 50–100 qubits but tolerating a significant level of imperfections—the advent of what has been termed noisy intermediate-scale quantum (NISQ) technology [1]. This move has enabled a fundamental success in the recent demonstration that such a NISQ quantum processor is capable of exhibiting a true “quantum advantage” over classical computing resources [2]. One of the leading NISQ platforms involves arrays of transmons, superconducting charge qubits [3], which by design are particularly resilient with regard to charge fluctuations. However, building larger quantum circuits from transmons comes with some intricate challenges [4,5] and will eventually mandate to incorporate quantum error correction (QEC) schemes [6]. Arguably the most promising approach here is the implementation of a surface code [7,8], which exploits topological properties of the system and, at the same time, remains experimentally feasible [9,10]. In practical settings, one downside of realizing such surface code architectures is the relatively slow decoding time of current quantum error correction codes.

The decoding step in quantum error correcting codes requires, at its core, a *classical* algorithm that efficiently infers the locations of errors from measured error syndromes [11]. The most widely adopted algorithm for this purpose is minimum weight perfect matching

(MWPM) [12], an algorithm which runs in polynomial time and is known to nearly achieve the optimal threshold for the independent noise model [13,14]. One of the drawbacks of the MWPM algorithm, however, is that its implementations are often simply too slow. To improve algorithmic scaling and to push error thresholds also for more general noise situations, a number of alternative decoding approaches have been suggested, of which the most notable might be the renormalization group (RG) [15–17] and union-find (UF) [18] decoders. The RG decoder runs, for a surface code in a two-dimensional (2D) geometry of linear size  $L$ , in  $O(L^2 \log L)$  time, often a significant improvement over the MWPM approach [which, in the worst case, scales cubic in the number of errors, i.e.,  $O(L^6)$  in code distance]. However, its threshold value of  $\sim 0.129$  for depolarizing noise [15] is lower than that of the MWPM algorithm ( $\sim 0.151$  [14]). The most efficient conventional algorithm is the UF decoder which runs in  $O[L^2 \alpha(L^2)]$ , i.e., almost linear in the number of qubits [19], with a threshold  $\sim 0.146$  for the depolarizing noise model (see below). In addition, the last two years have seen a flurry of activity to adopt machine learning (ML) techniques to best the decoding times and threshold values of these “conventional” algorithms [20–33]. As ML methods can be easily parallelized and generally offer a high degree of adaptability, one might easily accept their potential, but the first practical ML-based decoders typically delivered only on one of the two benchmarks—improving the error threshold at the expense of scalability or the other way round, providing good scalability but

leading to error thresholds which are sometimes even *below* those of the conventional algorithms [34].

It is the purpose of this Letter to introduce a powerful two-step decoding algorithm that combines neural network based preprocessing and union-find decoding to simultaneously achieve (i) improved error thresholds for depolarizing noise (even in the presence of syndrome measurement errors), (ii) algorithmic scalability up to tens of thousands of qubits, and (iii) real-life wall-clock run times (i.e., the elapsed time passed to execute the decoding process) that, for a range of error rates, best even those of the bare union-find algorithm. Our main algorithmic idea can be described as a hierarchical approach [33] that employs an ML decoder to preprocess *local* error corrections and leave the unresolved longer-range errors to a conventional UF decoding. The preprocessing step shifts a 2D subsystem over a given stabilizer code (akin to the preprocessing in a convolutional neural network) and decodes local errors in these subsystems. After this step, the system still exhibits errors that require longer range corrections, for which we employ a conventional UF decoder. However, since the preprocessing reduces the effective error rate—up to 2 orders of magnitude depending on the original error rate—this second step is extremely performant as compared to, e.g., employing UF decoding to the original unprocessed error instances. Extensive wall-clock time measurements of our approach (the true performance indicator in many real-life applications) show that our algorithm outperforms the bare UF decoder in a noise regime from 1% (in which one might want to operate quantum computing devices) up to the 10% regime where our ML-assisted approach is found to push the error threshold by some 15 % above the value of the bare UF decoder, as summarized in Table I. Our approach bears some similarity to the “lazy UF decoder” [33], which employs hierarchical decoding with a strictly local, hard decision preprocessing step and has been shown to substantially improve UF decoding for ultralow error rates below the per mil range.

*Hierarchical QEC.*—Throughout the Letter, we apply our decoding algorithm to the toric code in the presence of depolarizing noise as well as a scenario with additional syndrome measurement errors. For the latter, we use a phenomenological noise model where ancilla qubits for measuring syndromes are also subject to depolarizing noise but propagation of errors between data and ancilla qubits is neglected. The toric code is defined on a square lattice of linear size  $L$  and the stabilizer operators around the vertices and plaquettes are given by  $X_v = \prod_{i \in v} X_i$  and  $Z_p = \prod_{i \in p} Z_i$ . The code space is then spanned by the basis vectors  $\{|\psi\rangle : X_v|\psi\rangle = 1 \forall v, Z_p|\psi\rangle = 1 \forall p\}$ , which, for periodic boundary conditions, is four dimensional (and thus encodes two qubits) and the distance of the code is  $L$ . Each  $Z$  ( $X$ ) error on a qubit flips the value of the nearby  $X_v$  ( $Z_p$ ) operators.

TABLE I. Overview of results. For multiple variants of our decoding algorithm we provide the error threshold  $p_{\text{th}}$  (second column) for depolarizing noise (upper panel) and additional syndrome measurement errors (lower panel) where ancillary qubits for measuring syndromes are also subject to depolarizing noise, as well as wall-clock time measurements (in milliseconds) of the decoding time for different error rates (averaged over  $10^6$  instances) for code distances  $L = 255$  and  $L = 31$ , respectively. The boldfaced entries identify the best performing algorithm when optimizing for error threshold or compute times. Comparisons are shown for the union-find (UF) and minimum weight perfect matching (MWPM) decoders, combined with either lazy [33] or machine learning (ML) assisted preprocessing using subsystems of size  $\ell = 3, 5$ , or  $7$  as indicated in brackets (see main text). We have used a custom implementation for the UF decoder [42] and PyMatching [43] for MWPM [44]. In the presence of additional syndrome errors, the pure MWPM calculation was optimized by combining the Blossom and Dijkstra algorithms and for the ML-assisted MWPM with precomputed shortest paths. Details of our hardware setup are given in Supplemental Material [34].

Algorithm	$p_{\text{th}}$	$t_{p=0.01}$	$t_{p=0.05}$	$t_{p=0.1}$	$t_{p=0.1461}$
Depolarizing noise ( $L = 255$ )					
ML(7) + UF	<b>0.167(0)</b>	10.5	25.1	43.4	78.6
ML(5) + UF	0.162(5)	<b>6.7</b>	<b>12.8</b>	<b>26.2</b>	<b>56.2</b>
Lazy + UF	0.131(9)	6.9	20.7	51.1	...
UF	0.146(1)	8.4	22.5	44.9	92.8
ML(7) + MWPM	0.167(1)	~210	~530	~650	~980
ML(5) + MWPM	0.163(8)	~270	~510	~650	~970
MWPM	0.154(2)	~560	~840	~1100	~1300
Algorithm	$p_{\text{th}}$	$t_{p=0.01}$	$t_{p=0.02}$	$t_{p=0.03}$	$t_{p=0.0378}$
Depolarizing noise + syndrome errors ( $L = 31$ )					
ML(3) + UF	0.043(4)	12.1	13.5	<b>15.4</b>	<b>17.8</b>
Lazy + UF	0.031(3)	<b>11.1</b>	<b>12.8</b>	16.6	...
UF	0.037(8)	11.5	13.4	15.7	18.9
ML(3) + MWPM*	<b>0.044(5)</b>	14.6	25.8	81.5	229
MWPM	0.043(7)	211	239	273	294

The decoding problem is then defined as identifying the error configuration for a given syndrome, i.e., a given measurement of the outcomes of all stabilizers  $X_v$  and  $Z_p$ . To do so, we employ a two-step hierarchical procedure. In the first stage—the ML-assisted preprocessing—we aim to remove those errors that can be inferred from *local* syndromes. To this end, we only consider qubits directly connected to so-called defects (identified by an odd syndrome measurement  $X_v = -1$  or  $Z_p = -1$ ), as they are the typical source of locally correctable errors. To infer which error is the most probable for a given qubit, our preprocessing step shifts through all qubits with a subsystem of size  $\ell \times \ell$  centered around an “examination qubit” located at its center (see the setup in Fig. 1). The local inference task for each such examination qubit is then assigned to a neural network, whose details we

discuss below. The results of the inference are collected and the respective corrections are applied in one shot at the end. The outcome of this step is that a large number of local errors are decoded and only a small fraction of nonlocal errors, manifest on scales beyond the range of our subsystem, remain.

The second stage of our algorithm is to process the remaining nonlocal errors. To do so, we employ a conventional UF decoder on the remaining syndrome. Doing so is significantly more efficient than employing the UF decoder on the bare decoding problem (without the preprocessing), as we will see that the “effective error rate” for this UF decoding step is up to 2 orders of magnitude smaller than the original error rate.

*Neural decoder.*—At the heart of our hierarchical QEC approach is a neural network that decodes error syndromes within a local subsystem, as illustrated in Fig. 1. We train this neural network to output the most probable error (among the four possible  $\{I, X, Y, Z\}$  errors) of the central qubit given  $2\ell^2$  nearby syndromes as an input (with the factor of 2 coming from the two types of  $X$  and  $Z$  measurements). In machine learning, this type of task is commonly known as a multiclass classification problem and exceedingly well studied in the context of supervised learning approaches. To adopt such a supervised learning approach to optimize our neural network, we do training with a labeled dataset, i.e., batches of error-syndrome pairs generated for a given error rate (and noise model), training separate networks for each error rates. In practice, we train our networks in  $10^6$  epochs, for which we create independent sets of 512 error-syndrome batches “on the fly,” which also reduces the chance of overfitting.

In designing the neural network (NN) architecture, there is an inherent trade-off between the two algorithmic layers of our hierarchical approach: If one opts for a small NN, its computation time remains low but its accuracy in resolving local syndromes drops, resulting in more computational

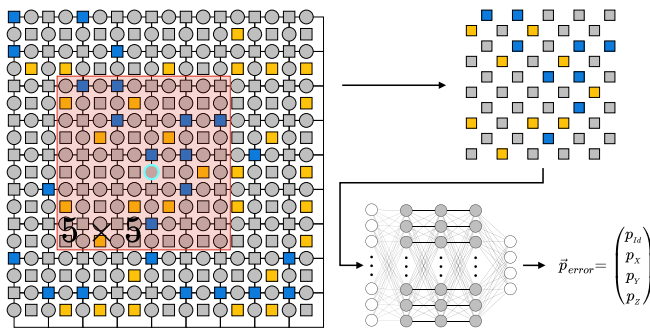


FIG. 1. Neural network setup. Syndromes in the immediate vicinity (red shading) of a reference qubit (cyan circle) are used as input, whereby measured syndromes (blue or yellow) are assigned the value  $+1/-1$  and no syndromes (gray) are assigned value 0, respectively. Passing the input through the feed forward network results in the error probabilities for the reference qubit.

load for the UF decoder on the higher algorithmic layer. If, on the other hand, one opts for a large NN, its accuracy in resolving syndromes goes up at the cost of larger compute times, while also alleviating the load of the higher-level UF decoder. Indeed, this trade-off leads to a sweet spot, i.e., an intermediate NN size that results, e.g., in minimal wall-clock run times *or* maximal error thresholds. To identify an optimal configuration, we have explored a multitude of different network architectures for the case of depolarizing noise, varying the size of the subsystem, the depth of the network, and the number of nodes per layer as main parameters (as detailed in Supplemental Material [34]). When optimizing for compute speed a  $5 \times 5$  subsystem turns out to be ideal, while pushing the error threshold one might want to go with a  $7 \times 7$  subsystem—see Table I. However, since the error threshold of the speed-optimized network is only 3% smaller than the threshold-optimized network, we consider the  $5 \times 5$  NN approach the best compromise in achieving fast decoding *and* high error thresholds for an algorithm that *also* delivers on high scalability.

*Benchmark results.*—In benchmarking our hierarchical QEC algorithm, we start in the high-noise regime and calculate the error threshold of our approach. Decoding  $10^6$  random instances of depolarizing noise for different error rates and linear system sizes in the range  $L = 7, \dots, 127$  we can readily deduce the error threshold from the finite-size scaling shown in Fig. 2. In comparison to the bare UF

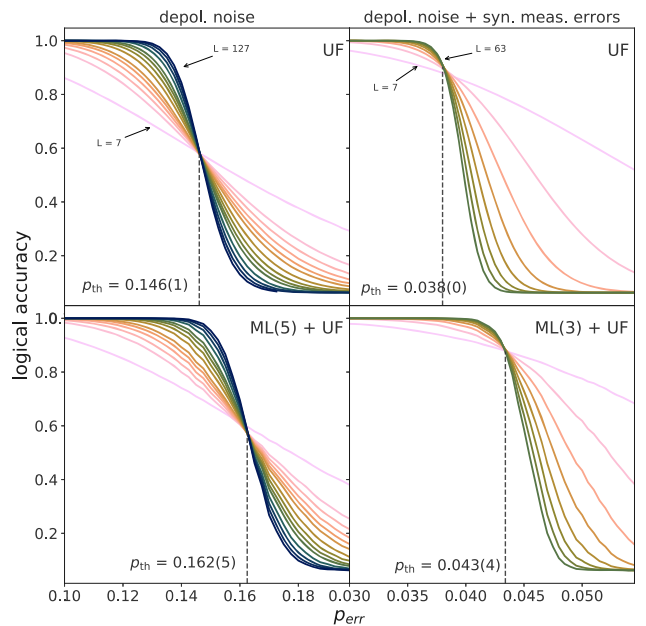


FIG. 2. Error threshold and scaling behavior for the conventional union find (UF) algorithm (upper row), and the machine learning assisted ML + UF algorithm (lower row) for depolarizing noise (left column) and additional syndrome errors (right column).

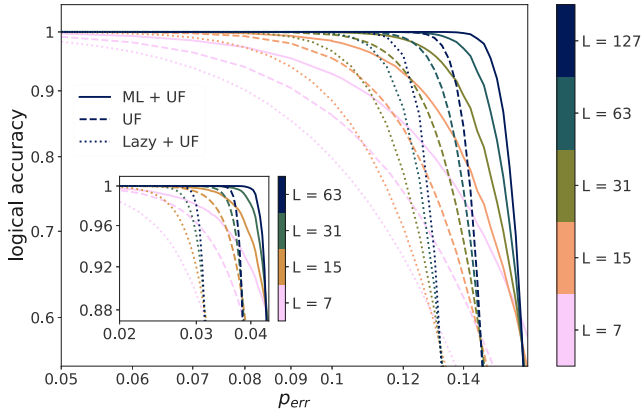


FIG. 3. Logical accuracy of the conventional UF decoder and combined with lazy or ML-assisted preprocessing for depolarizing noise. The inset shows the case of additional syndrome measurement errors. The ML + UF decoder increases the logical accuracy, independent of system size, for all error rates shown.

algorithm (top panel), which exhibits an error threshold of  $p_{\text{th}}^{\text{UF}} = 0.146(1)$ , our algorithm yields a 10% higher value of  $0.162(5)$  and an increase of more than 20% compared to the lazy UF decoder’s threshold of  $0.131(9)$  [45]. This notable increase of the error threshold indicates that our ML-assisted approach is capable of identifying and resolving *correlated* errors in the depolarizing noise, which the bare UF decoder cannot handle. The strength of the ML-assisted decoder in the dense error regime can also be exemplified by the logical accuracy near the threshold plotted in Fig. 3, which shows a higher logical accuracy for the ML + UF decoder in this regime, independent of system size. It should further be noted that our threshold values are higher than the one of the bare RG decoder [15] with  $p_{\text{th}}^{\text{RG}} = 0.153$  and comparable to those found for a combination of RG and sparse decoders [17], or the best ML-based decoders using deep neural networks, for which error thresholds of  $p_{\text{th}}^{\text{ML}} \approx 0.165$  are reported [22,28] for depolarizing noise. However, our result is still significantly below the optimal theoretical value [46] of  $p_{\text{opt}} = 0.189(3)$ . Performing a similar analysis for a scenario with additional syndrome measurement errors, we come to analogous conclusions with a spread of the error threshold between  $p_{\text{th}} = 0.031(3)$  for the lazy UF decoder and  $0.044(5)$  obtained for ML-assisted MWPM decoding (lower panel of Table I).

One measure to illustrate the inner workings of our hierarchical approach is an “effective error rate,” i.e., the reduction of errors obtained after performing the first ML-assisted step of our algorithm. Shown in Fig. 4, this effective error rate reveals that preprocessing is particularly powerful at low error rates, i.e., in the regime where few long-range errors occur. Here, one can reduce the initial error rate by more than 2 orders of magnitude (see the right panel of Fig. 4), thereby significantly speeding up the

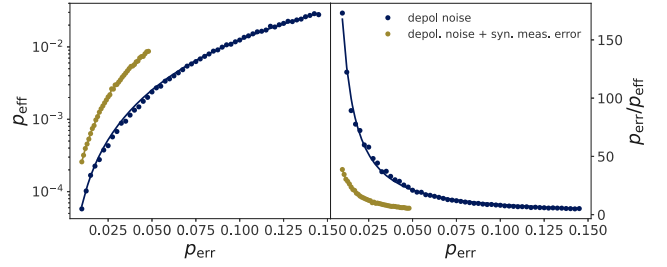


FIG. 4. Effective error reduction attained by the ML preprocessing step. Left: The effective error rate  $p_{\text{eff}}$  as a function of the original error probability  $p_{\text{err}}$ . The effective error rate is calculated from the number of remaining syndromes  $p_{\text{eff}} = \sum S_i / ((4/3) \times 2 \times 2L^2)$ . Right: The ratio of original error probability and effective error rate.

subsequent UF decoding step (as compared to a direct application to the original syndrome).

As such one might naively expect the biggest computing gain of our algorithm in the low-noise regime. For practical implementations this is, however, not true as becomes apparent when performing run-time measurements of our decoder. Such measurements are illustrated in Fig. 5 where the decoding time (again averaged over  $10^6$  error instances) is plotted versus the linear system size for different error rates. The top panel nicely demonstrates that, for large system sizes, we find near linear scaling for both the UF and our hierarchical UF + ML decoder, independent of the error rate. Note that our ML-assisted decoder easily scales up to  $2 \times 255 \times 255 \approx 130\,000$  qubits where the decoding time per instance is still a fraction of a second—this should be contrasted to other ML-based decoders reported in the literature, which could not be scaled beyond a hundred qubits (see the overview in Supplemental Material [34] Table IV).

If we look at the scaling of our algorithm for small to moderate system sizes (highlighted in the lower panels of Fig. 5), a breakdown of the linear scaling of the ML-assisted decoder becomes evident. There is a considerable “lag” in our implementation, which arises from using an external graphics processing unit (GPU) to perform the preprocessing step (see Supplemental Material [34] for hardware specifications). Doing so readily implies another inherent trade-off: initializing the neural network and loading the syndrome data to the GPU has almost constant overhead, which explains the plateau in our scaling plots for small system sizes where the advantage of GPU processing of the neural network is not compensating this overhead (as it does for large system sizes). We have measured this “kernel start-up” time to subtract this overhead—which would not exist in a dedicated or *in situ* device in a practical implementation of QEC in the lab—to arrive at the “kernel adjusted” scaling of Fig. 5. The point at which the ML-assisted decoder outperforms the bare UF decoder comes down to code distances of  $L \approx 31$ , but we

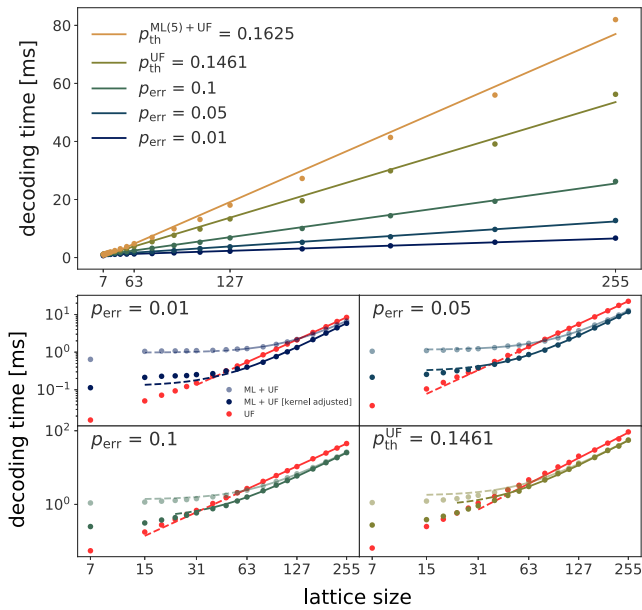


FIG. 5. Algorithmic scaling of our hierarchical decoder for various rates of depolarizing noise (top panel). The bottom panels show comparisons with the bare UF decoder on a log-log scale. Shown is the average decoding time measured in wall clock time averaged over  $10^6$  error instances. The “kernel adjusted” time in the lower panels is the ML + UF decoding time subtracted by a constant offset, to compensate kernel launch times (see main text).

expect even smaller code distances to benefit from the ML-assisted approach when going for an *in situ* implementation [35], using field-programmable gate arrays or tensor processing units [36].

In summary, we have demonstrated that the combination of machine-learning assisted preprocessing in conjunction with conventional decoders in a newly devised hierarchical approach results in a vastly scalable algorithm. Our practical implementation shows that one can increase logical accuracy and push the error threshold by resolving correlated errors, while also reducing the actual decoding times (to a few milliseconds in our hardware setup) particularly in the dense error regime. As such our approach nicely complements the lazy UF decoder [33] which excels in the opposite regime of ultralow error rates. Taken together, one might argue that one should always combine the UF decoder with some sort of preprocessing step—which one to go for depends on the expected noise level and code distances.

We thank M. Kastoryano and T. Wagner for insightful discussions, as well as O. Higgott for comments on optimizing the PyMatching results for MWPM decoding the phenomenological noise model. This project was funded by the Deutsche Forschungsgemeinschaft under Germany’s Excellence Strategy—Cluster of Excellence Matter and Light for Quantum Computing (ML4Q) EXC 2004/1–390534769 and within the CRC network TR 183 (Project Grant No. 277101999) as part of project B01.

- [1] J. Preskill, Quantum Computing in the NISQ era and beyond, *Quantum* **2**, 79 (2018).
- [2] F. Arute *et al.*, Quantum supremacy using a programmable superconducting processor, *Nature (London)* **574**, 505 (2019).
- [3] J. Koch, T. M. Yu, J. Gambetta, A. A. Houck, D. I. Schuster, J. Majer, A. Blais, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf, Charge-insensitive qubit design derived from the Cooper pair box, *Phys. Rev. A* **76**, 042319 (2007).
- [4] D. Willsch, M. Nocon, F. Jin, H. De Raedt, and K. Michielsen, Gate-error analysis in simulations of quantum computers with transmon qubits, *Phys. Rev. A* **96**, 062302 (2017).
- [5] C. Berke, E. Varvelis, S. Trebst, A. Altland, and D. P. DiVincenzo, Transmon platform for quantum computing challenged by chaotic fluctuations, [arXiv:2012.05923](https://arxiv.org/abs/2012.05923).
- [6] D. Gottesman, An introduction to quantum error correction and fault-tolerant quantum computation, *Proc. Symp. Appl. Math.* **68**, 13 (2010).
- [7] A. Y. Kitaev, Fault-tolerant quantum computation by anyons, *Ann. Phys. (Amsterdam)* **303**, 2 (2003).
- [8] S. B. Bravyi and A. Y. Kitaev, Quantum codes on a lattice with boundary, [arXiv:quant-ph/9811052](https://arxiv.org/abs/quant-ph/9811052).
- [9] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Surface codes: Toward practical large-scale quantum computation, *Phys. Rev. A* **86**, 032324 (2012).
- [10] C. K. Andersen, A. Remm, S. Lazar, S. Krinner, N. Lacroix, G. J. Norris, M. Gabureac, C. Eichler, and A. Wallraff, Repeated quantum error detection in a surface code, *Nat. Phys.* **16**, 875 (2020).
- [11] D. Gottesman, Stabilizer codes and quantum error correction, Ph.D. thesis, California Institute of Technology, 1997.
- [12] J. Edmonds, Path, trees, and flowers, *Can. J. Math.* **17**, 449 (1965).
- [13] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Topological quantum memory, *J. Math. Phys. (N.Y.)* **43**, 4452 (2002).
- [14] J. W. Harrington, Analysis of quantum error-correcting codes: Symplectic lattice codes and toric codes, Ph.D. thesis, California Institute of Technology, 2004.
- [15] G. Duclos-Cianci and D. Poulin, Fast Decoders for Topological Quantum Codes, *Phys. Rev. Lett.* **104**, 050504 (2010).
- [16] G. Duclos-Cianci and D. Poulin, in *2010 IEEE Information Theory Workshop (IEEE, Dublin, 2010)*, pp. 1–5.
- [17] G. Duclos-Cianci and D. Poulin, Fault-tolerant renormalization group decoder for abelian topological codes, *Quantum Inf. Comput.* **14**, 721 (2014).
- [18] N. Delfosse and N. H. Nickerson, Almost-linear time decoding algorithm for topological codes, *Quantum* **5**, 595 (2021).
- [19] The prefactor  $\alpha(L^2)$  is the inverse of Ackermann’s function whose value is  $< 3$  for all practical purposes.
- [20] S. Varsamopoulos, B. Criger, and K. Bertels, Decoding small surface codes with feedforward neural networks, *Quantum Sci. Technol.* **3**, 015004 (2018).
- [21] G. Torlai and R. G. Melko, Neural Decoder for Topological Codes, *Phys. Rev. Lett.* **119**, 030501 (2017).
- [22] S. Krastanov and L. Jiang, Deep neural network probabilistic decoder for stabilizer codes, *Sci. Rep.* **7**, 11003 (2017).

- [23] C. Chamberland and P. Ronagh, Deep neural decoders for near term fault-tolerant experiments, *Quantum Sci. Technol.* **3**, 044002 (2018).
- [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] Y.-H. Liu and D. Poulin, Neural Belief-Propagation Decoders for Quantum Error-Correcting Codes, *Phys. Rev. Lett.* **122**, 200501 (2019).
- [26] P. Andreasson, J. Johansson, S. Liljestrand, and M. Granath, Quantum error correction for the toric code using deep reinforcement learning, *Quantum* **3**, 183 (2019).
- [27] T. Wagner, H. Kampermann, and D. Bruß, Symmetries for a high-level neural decoder on the toric code, *Phys. Rev. A* **102**, 042411 (2020).
- [28] D. Fitzek, M. Eliasson, A. F. Kockum, and M. Granath, Deep  $Q$ -learning decoder for depolarizing noise on the toric code, *Phys. Rev. Research* **2**, 023230 (2020).
- [29] L. Domingo Colomer, M. Skotiniotis, and R. Muñoz-Tapia, Reinforcement learning for optimal error correction of toric codes, *Phys. Lett. A* **384**, 126353 (2020).
- [30] X. Ni, Neural network decoders for large-distance 2D toric codes, *Quantum* **4**, 310 (2020).
- [31] S. Varona and M. A. Martin-Delgado, Determination of the semion code threshold using neural decoders, *Phys. Rev. A* **102**, 032411 (2020).
- [32] R. Sweke, M. S. Kesselring, E. P. L. van Nieuwenburg, and J. Eisert, Reinforcement learning decoders for fault-tolerant quantum computation, *Mach. Learn.* **2**, 025005 (2021).
- [33] N. Delfosse, Hierarchical decoding to reduce hardware requirements for quantum computing, [arXiv:2001.11427](https://arxiv.org/abs/2001.11427).
- [34] See Supplemental Material at <http://link.aps.org/supplemental/10.1103/PhysRevLett.128.080505>, which includes Refs. [35–41], for a detailed overview of ML-based decoders, a more detailed description of the neural network, MWPM, and UF, as well as a comparison to other decoding algorithms.
- [35] P. Das, C. A. Pattison, S. Manne, D. Carmean, K. Svore, M. Qureshi, and N. Delfosse, A scalable decoder micro-architecture for fault-tolerant quantum computing, [arXiv:2001.06598](https://arxiv.org/abs/2001.06598).
- [36] N. P. Jouppi *et al.*, in *Proceedings of the 2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)* (Association for Computing Machinery, New York, 2017), pp. 1–12.
- [37] TensorFlow, <https://github.com/tensorflow/tensorflow>.
- [38] H. N. Gabow, Implementation of algorithms for maximum matching on nonbipartite graphs, Ph. D. thesis, Stanford University, 1974.
- [39] E. L. Lawler, *Combinatorial Optimization: Networks and Matroids* (Holt, Rinehart, and Winston, 1976).
- [40] A. G. Fowler, A. C. Whiteside, and L. C. L. Hollenberg, Toward Practical Classical Processing for the Surface Code, *Phys. Rev. Lett.* **108**, 180501 (2012).
- [41] N. Delfosse and G. Zémor, Linear-time maximum likelihood decoding of surface codes over the quantum erasure channel, *Phys. Rev. Research* **2**, 033042 (2020).
- [42] C.-Y. Park and K. Meinerz, Open-source C++ implementation of the Union-Find decoder, <https://github.com/chaeyeun-park/UnionFind>.
- [43] O. Higgott and N. P. Breuckmann, PyMatching, <https://pymatching.readthedocs.io/>.
- [44] O. Higgott and N. P. Breuckmann, Subsystem Codes with High Thresholds by Gauge Fixing and Reduced Qubit Overhead, *Phys. Rev. X* **11**, 031039 (2021).
- [45] One can push the threshold value even further up (at the expense of additional compute time) by employing a larger  $7 \times 7$  subsystem, which gives  $p_{\text{th}} = 0.167(0)$  (see also Table I). Going to even larger subsystems has not resulted in any further notable improvement of the threshold value.
- [46] H. Bombin, R. S. Andrist, M. Ohzeki, H. G. Katzgraber, and M. A. Martin-Delgado, Strong Resilience of Topological Codes to Depolarization, *Phys. Rev. X* **2**, 021004 (2012).