THE END: AN EQUIVARIANT NEURAL DECODER FOR QUANTUM ERROR CORRECTION

Evgenii Egorov *
QUvA lab, University of Amsterdam
egorov.evgenyy@ya.ru

Roberto Bondesan *
Qualcomm AI Research, Netherlands
r.bondesan@gmail.com

Max Welling
QUvA lab, University of Amsterdam
m.welling@uva.nl

ABSTRACT

Quantum error correction is a critical component for scaling up quantum computing. Given a quantum code, an optimal decoder maps the measured code violations to the most likely error that occurred, but its cost scales exponentially with the system size. Neural network decoders are an appealing solution since they can learn from data an efficient approximation to such a mapping and can automatically adapt to the noise distribution. In this work, we introduce a data efficient neural decoder that exploits the symmetries of the problem. We characterize the symmetries of the optimal decoder for the toric code and propose a novel equivariant architecture that achieves state of the art accuracy compared to previous neural decoders.

1 INTRODUCTION

Quantum computers have exponential advantage compared to classical computers for quantum physics simulations and for breaking certain cryptosystems. They can also provide speed ups for optimization and searching problems. These advantages are guaranteed only for fault tolerant architectures and quantum error correction is a critical component to build a fault tolerant quantum computer.

The prototypical example of a quantum code is the toric code (Kitaev, 2003), where qubits are placed on the edges of a torus and the logical qubits are associated with operations along the non-contractible loops of the torus. This model (or rather its variant with open boundaries) has been implemented in current hardware (Krinner et al., 2022; Zhao et al., 2022; Google Quantum AI, 2022) and is a standard benchmark for developing new decoders.

The decoding problem aims at correcting the errors that occurred in a given time cycle. Exact optimal decoding is computationally intractable (Iyer & Poulin, 2013), and a standard approach in the literature is to devise handcrafted heuristics (Dennis et al., 2002; Delfosse & Nickerson, 2021) that give a good tradeoff between time and accuracy. The downside of these is however that they are tailored to a specific code or noise model. Neural decoders have been proposed to overcome these limitations, by learning from data how to adapt to experimental setups. Neural network decoders also benefit from quantization and dedicated hardware that allow them to meet the time requirements for decoders to be useful when deployed (Overwater et al., 2022). Several works therefore studied neural decoders for the toric code. Pure neural solutions are however limited to small system sizes (Krastanov & Jiang, 2017; Wagner et al., 2020) or low accuracy (Ni, 2020). Solutions that combine neural networks with classical heuristics can reach large systems but are limited in their accuracy by the underlying heuristics (Meinerz et al., 2021). We discuss related work in App. 4.1 and properties of decoders in App. 4.3.

In this work, we show how to improve the performance of neural decoders by designing an equivariant neural network that approximates the optimal decoder for the toric code. Our contributions are as follows:

*Equal contribution.

- We characterize the geometric symmetries of the optimal decoder for the toric code.
- We propose an equivariant neural decoder architecture. The key innovation is a novel twisted version of the global average pooling over the symmetry group.
- We benchmark a translation equivariant model against neural and non-neural decoders. Our model achieves state of the art accuracy compared to previous neural decoders.

2 SYMMETRIES OF THE TORIC CODE DECODER

We provide a short introduction to Toric code construction in App. 4.2.

2.1 MAXIMUM LIKELIHOOD DECODING

Let us denote by $p(E)$ the probability for an error E to occur, and assume that p is known. To correct an (unknown) error E we first measure its syndrome σ. This is a binary vector of size $2L^2$, whose i-th entry is 1 if E anticommutes with the i-th stabilizer and zero otherwise. The decoding problem is then to reconstruct the error given the syndrome. It is rather easy to produce an error that is compatible with a syndrome. In fact, note that syndromes always come in pairs at the end of the error paths, as shown in figure 1 by looking at the error paths a or c. Note that all operators on an error path, except the ones on the endpoints, intersect twice or zero times any member of stabilizer group, and thus commute with it. Thus a simple decoding strategy is to return error paths that join syndromes in pairs. Any such paths will produce an error which has the correct syndrome. However, there are many possible errors compatible with a syndrome since since both stabilizer and logical operators have trivial syndrome since they commute with all stabilizers. For example, the error c, d or a, b have the same syndromes in figure 1.

To understand what constitutes an optimal reconstruction we argue as follows. First, we note that stabilizer errors do not need to be corrected since by definition they act trivially on the logical qubits, and so two errors E and E' are equivalent if they differ by a stabilizer operator. However, logical operators do change the logical state, and the optimal decoding strategy is then to choose the most likely logical operator. The likelihood of a logical operator is to be computed by taking into account that any of the possible errors that are compatible with the syndrome and the logical operator content but differ by a stabilizer could have occurred.

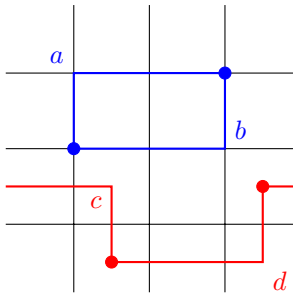


Figure 1: a and b are two possible errors (phase flip paths) that give rise to the same syndrome, here represented by blue dots. Similarly, c, d are two possible errors (bit flip paths) with the same syndrome, represented by red dots.

Formally, let us define the vector $L = (\bar{X}_1, \bar{X}_2, \bar{Z}_1, \bar{Z}_2)$. There are 16 possible logical operators corresponding to the 4 binary choices of acting or not with L_a, for $a = 1, \ldots, 4$. Similarly to the syndrome, we define the logical content of an error E as the four-dimensional binary vector $\omega(E, L)$, $L = (\bar{X}_1, \bar{X}_2, \bar{Z}_1, \bar{Z}_2)$. This allows us to detect whether any of the logical operators are part of E. (Note that one needs to swap the first two entries of γ with the last two entries to reconstruct the logical operator content of E due to the commutation relations. For example, $E = \bar{X}_1$, has $\omega(E, L) = (0, 0, 1, 0)$.) Then we consider the probability mass of all errors compatible with σ and γ:

$$p(\gamma, \sigma) = \sum_{E \in \mathcal{P}} p(E)\delta(\omega(E, S), \sigma)\delta(\omega(E, L), \gamma), \tag{1}$$

where $\mathcal{P}$ is the set of possible errors and S is a vector with all Z and X stabilizers. From the discussion above, the sum is effectively over all possible 2^{2L^2} stabilizer operators – all the possible products of plaquette and vertex operators. The most likely γ will then allow us to obtain the optimal reconstruction, so maximum likelihood decoding amounts to solving the following optimization problem:

$$\max_{\gamma \in \{0,1\}^4} p(\gamma|\sigma)\,. \tag{2}$$

In the following we shall consider the i.i.d. noise called depolarizing noise, which is a standard choice for benchmarking quantum error correction codes (Nielsen & Chuang, 2000):

$$p(E) = \prod_{e \in \mathcal{E}} \pi(E_e)\,, \quad \pi(E) = \begin{cases} 1-p & E = \mathbf{1} \\ p/3 & E \in \{X, Z, XZ\} \end{cases}\,. \tag{3}$$

with $\mathcal{E}$ the set of edges of the lattice. The number p is in $[0, 1]$ and we give the same probability $p/3$ to the events corresponding to the errors X, Z, XZ.

2.2 Symmetries of the code

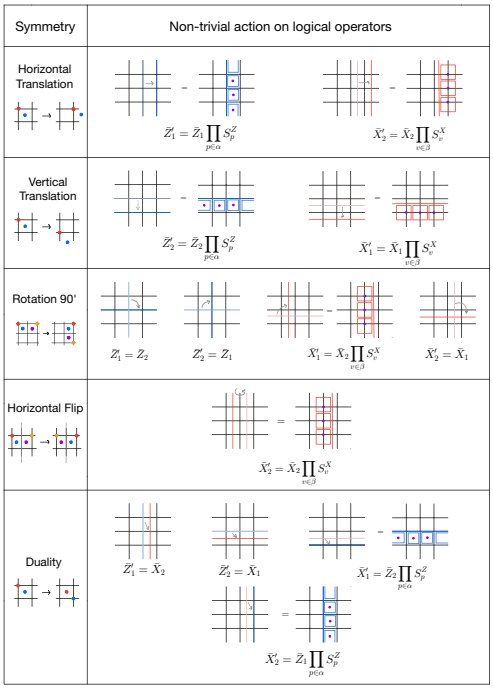


Figure 2: Left column: the list of symmetries of the toric code decoder and their action on the vertices and plaquettes. Right column: non-trivial action of those symmetries on the logical operators. Purple dots indicate the paths α, β in the formulas below the pictures. We assume odd L and rotations are performed around the center vertex of the lattice, while horizontal flips are done around the vertical middle line.

3 Machine learning approach

3.1 Data generation and loss function

We now set up the task of learning the logical error probabilities $p(\gamma|\sigma)$ introduced in section 2.1. The goal is to amortize the cost of maximum likelihood decoding via training a low complexity neural network. We are given a noise model equation 3 $p(E)$ from which we can sample errors $E^1, E^2, \ldots$ and compute syndrome $\sigma = \omega(E, S)$ and logical components $\gamma = \omega(E, L)$. The pairs (γ, σ)'s are distributed according to equation 1 and taken to be inputs and outputs of a supervised learning task. We thus aim at learning a map $\hat{p}$ that maps a syndrome $\sigma \in \{0, 1\}^{2L^2}$ to a probability distribution over a categorical variable with $2^4 = 16$ values. We learn this map by minimizing the cross entropy loss $\mathbb{E}_{\sigma \sim p(\sigma)} \mathbb{E}_{\gamma \sim p(\gamma|\sigma)} (-\log \hat{p}(\sigma)_\gamma)$ The minimizer of this loss function satisfies $\hat{p}(\sigma)_\gamma = p(\gamma|\sigma)$.

Decoder	L	p : 0.155	0.166	0.178	0.18
UFML	$(7, 63)$	$(0.5; 0.6)$	< 0.6	< 0.45	< 0.2
MWPM	17	0.55	0.43	0.31	0.29
(17, 32)	17	0.77	0.66	0.52	0.49
(17, 64)	17	0.82	0.72	0.57	0.55
(19, 128)	17	0.82	0.72	0.58	0.55
(17, 128)	17	0.85	0.75	0.61	0.59
MWPM	19	0.55	0.42	0.29	0.28
(17, 32)	19	0.75	0.63	0.47	0.45
(17, 64)	19	0.82	0.70	0.54	0.52
(19, 128)	19	0.84	0.72	0.57	0.55
(17, 128)	19	0.85	0.74	0.59	0.57
MWPM	21	0.55	0.41	0.28	0.26
(17, 32)	21	0.70	0.56	0.40	0.38
(17, 64)	21	0.77	0.63	0.46	0.44
(17, 128)	21	0.83	0.70	0.53	0.51
(19, 128)	21	0.83	0.71	0.55	0.53

Table 1: Logical accuracy of decoders over depolarising noise for $L : 17, 19, 21$ and noise levels around thresholds of competitive decoders. All *END* decoders were trained with noise probability 0.17 and denoted in the table by $(\mathrm{L}, \mathrm{ch})$: training lattice size and the number of channels in the first block of CNN body. All St.d. ≤ 0.002, sample size 10^6. UFML is the method of (Meinerz et al., 2021) which is the best neural decoder in the literature, and MPWM the minimum weight perfect matching decoder (Dennis et al., 2002).

3.2 General theory of equivariant architectures

We discuss the symmetry action introduced in theorem 4.1. Let us suppose that as in the theorem 4.1 we have a vector-valued function $\boldsymbol{f}(\sigma)$ and a symmetry action $(\rho_g \boldsymbol{f})(\sigma) = M_g(\sigma)\boldsymbol{f}(g^{-1} \cdot \sigma)$. For ρ to be a well defined symmetry action (group homorphism) we need $\rho_g \rho_h = \rho_{gh}$ for any g, h in the symmetry group G. As shown in App. 4.6, this leads to the following relations:

$$M_{gh}(\sigma) = M_g(\sigma) M_h(g^{-1} \cdot \sigma)\,. \tag{4}$$

The dependency of M on σ, the input to the function on which M acts on, makes the problem more complicated than those typically considered in the machine learning literature on equivariance (Weiler & Cesa, 2021). In fact, typically one considers functions $\boldsymbol{f} : V_{\text{in}} \to V_{\text{out}}$, with $V_{\text{in}}, V_{\text{out}}$ input and output linear representations of G.

In our case instead, the output representation matrix $M_g(\sigma)$ depends on the input σ, and therefore we cannot immediately use the standard theory of equivariant neural networks (Weiler & Cesa, 2021), which prescribes an alternation of layers with different linear representations of the group. Instead, we solve the problem of parametrizing the invariant function $\boldsymbol{p}$ of theorem 4.1 by projecting a general function onto the G-invariant subspace by symmetrizing over the group action.

Proposition 3.1. *Consider the group action $(\rho_g \boldsymbol{f})(\sigma) = M_g(\sigma)\boldsymbol{f}(g^{-1} \cdot \sigma)$ on a function $\boldsymbol{f} : \mathbb{R}^d \to \mathbb{R}^\ell$. If $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^{|G|} \otimes \mathbb{R}^\ell$ is G-equivariant, $\phi_{h,\gamma}(g^{-1} \cdot \sigma) = \phi_{gh,\gamma}(\sigma)$, then the following is invariant:*

$$\hat{\boldsymbol{f}}(\sigma) = \frac{1}{|G|} \sum_{h \in G} M_h(\sigma) \boldsymbol{\phi}_h(\sigma)\,. \tag{5}$$

The key innovation of our construction is to twist the sum by the matrix $M_g(\sigma)$ which ensures the right equivariance. App. 4.7 contains details of the implementation of $M_g(\sigma)$ for the translation group, and shows that we can compute the function $\hat{\boldsymbol{f}}$ defined in proposition 3.1 efficiently in $O(L^2)$.

3.3 Experiments (App. 4.8 contains details)

In Table 1 (3) we compare our method against the literature in terms of the logical accuracy. This is estimated as the fraction of correct reconstructions that a decoder produces over an ensemble of errors. We compared our method against MWPM (Dennis et al., 2002), the most popular decoder for the toric code, and (Meinerz et al., 2021), the best neural decoder in the literature.

REFERENCES

Bravyi, S., Terhal, B. M., and Leemhuis, B. Majorana fermion codes. *New Journal of Physics*, 12 (8):083039, Aug 2010. ISSN 1367-2630. doi: 10.1088/1367-2630/12/8/083039. URL http://dx.doi.org/10.1088/1367-2630/12/8/083039.

Bravyi, S., Suchara, M., and Vargo, A. Efficient algorithms for maximum likelihood decoding in the surface code. *Physical Review A*, 90(3), Sep 2014. ISSN 1094-1622. doi: 10.1103/physreva.90.032326. URL http://dx.doi.org/10.1103/PhysRevA.90.032326.

Chubb, C. T. General tensor network decoding of 2d pauli codes, 2021.

Cohen, T. S. and Welling, M. Group equivariant convolutional networks. 2016. doi: 10.48550/ARXIV.1602.07576. URL https://arxiv.org/abs/1602.07576.

Delfosse, N. and Nickerson, N. H. Almost-linear time decoding algorithm for topological codes. *Quantum*, 5:595, Dec 2021. ISSN 2521-327X. doi: 10.22331/q-2021-12-02-595. URL http://dx.doi.org/10.22331/q-2021-12-02-595.

Dennis, E., Kitaev, A., Landahl, A., and Preskill, J. Topological quantum memory. *Journal of Mathematical Physics*, 43(9):4452–4505, Sep 2002. ISSN 1089-7658. doi: 10.1063/1.1499754. URL http://dx.doi.org/10.1063/1.1499754.

Duclos-Cianci, G. and Poulin, D. Fast decoders for topological quantum codes. *Physical Review Letters*, 104(5), Feb 2010. ISSN 1079-7114. doi: 10.1103/physrevlett.104.050504. URL http://dx.doi.org/10.1103/PhysRevLett.104.050504.

Fowler, A. G., Mariantoni, M., Martinis, J. M., and Cleland, A. N. Surface codes: Towards practical large-scale quantum computation. *Physical Review A*, 86(3), Sep 2012. ISSN 1094-1622. doi: 10.1103/physreva.86.032324. URL http://dx.doi.org/10.1103/PhysRevA.86.032324.

Google Quantum AI. Suppressing quantum errors by scaling a surface code logical qubit, 2022. URL https://arxiv.org/abs/2207.06431.

Higgott, O. PyMatching: A python package for decoding quantum codes with minimum-weight perfect matching. *arXiv preprint arXiv:2105.13082*, 2021.

Iyer, P. and Poulin, D. Hardness of decoding quantum stabilizer codes, 2013.

Kitaev, A. Fault-tolerant quantum computation by anyons. *Annals of Physics*, 303(1):2–30, Jan 2003. ISSN 0003-4916. doi: 10.1016/s0003-4916(02)00018-0. URL http://dx.doi.org/10.1016/S0003-4916(02)00018-0.

Krastanov, S. and Jiang, L. Deep neural network probabilistic decoder for stabilizer codes. *Scientific Reports*, 7(1), sep 2017. doi: 10.1038/s41598-017-11266-1. URL https://doi.org/10.1038%2Fs41598-017-11266-1.

Krinner, S., Lacroix, N., Remm, A., Di Paolo, A., Genois, E., Leroux, C., Hellings, C., Lazar, S., Swiadek, F., Herrmann, J., et al. Realizing repeated quantum error correction in a distance-three surface code. *Nature*, 605(7911):669–674, 2022.

Liu, Y.-H. and Poulin, D. Neural belief-propagation decoders for quantum error-correcting codes. *Physical Review Letters*, 122(20), May 2019. ISSN 1079-7114. doi: 10.1103/physrevlett.122.200501. URL http://dx.doi.org/10.1103/PhysRevLett.122.200501.

Meinerz, K., Park, C.-Y., and Trebst, S. Scalable neural decoder for topological surface codes, 2021.

Ni, X. Neural Network Decoders for Large-Distance 2D Toric Codes. *Quantum*, 4:310, August 2020. ISSN 2521-327X. doi: 10.22331/q-2020-08-24-310. URL https://doi.org/10.22331/q-2020-08-24-310.

Nielsen, M. and Chuang, I. *Quantum Computation and Quantum Information*. Cambridge Series on Information and the Natural Sciences. Cambridge University Press, 2000. ISBN 9780521635035. URL https://books.google.co.in/books?id=65FqEKQOfP8C.

Overwater, R. W. J., Babaie, M., and Sebastiano, F. Neural-network decoders for quantum error correction using surface codes: A space exploration of the hardware cost-performance tradeoffs. *IEEE Transactions on Quantum Engineering*, 3:1–19, 2022. doi: 10.1109/tqe.2022.3174017. URL https://doi.org/10.1109%2Ftqe.2022.3174017.

Satorras, V. G. and Welling, M. Neural enhanced belief propagation on factor graphs. In *International Conference on Artificial Intelligence and Statistics*, pp. 685–693. PMLR, 2021.

Wagner, T., Kampermann, H., and Bruß, D. Symmetries for a high-level neural decoder on the toric code. *Physical Review A*, 102(4):042411, 2020.

Weiler, M. and Cesa, G. General $e(2)$-equivariant steerable cnns, 2021.

Zhao, Y., Ye, Y., Huang, H.-L., Zhang, Y., Wu, D., Guan, H., Zhu, Q., Wei, Z., He, T., Cao, S., Chen, F., Chung, T.-H., Deng, H., Fan, D., Gong, M., Guo, C., Guo, S., Han, L., Li, N., Li, S., Li, Y., Liang, F., Lin, J., Qian, H., Rong, H., Su, H., Sun, L., Wang, S., Wu, Y., Xu, Y., Ying, C., Yu, J., Zha, C., Zhang, K., Huo, Y.-H., Lu, C.-Y., Peng, C.-Z., Zhu, X., and Pan, J.-W. Realization of an error-correcting surface code with superconducting qubits. *Phys. Rev. Lett.*, 129:030501, Jul 2022. doi: 10.1103/PhysRevLett.129.030501. URL https://link.aps.org/doi/10.1103/PhysRevLett.129.030501.

4 APPENDIX

4.1 RELATED WORK

Popular handcrafted decoders for the toric code are the minimum weight perfect matching (MWPM) decoder (Dennis et al., 2002) and the union find decoder (Delfosse & Nickerson, 2021). These decoders however treat independently bit and phase flip errors, and they do not count correctly degenerate errors. For these reasons they are practically fast but have limited accuracy. Not dealing with degenerate errors impacts also their equivariance as discussed in details in Appendix 4.3. Decoders based on tensor network contraction (Bravyi et al., 2010; Chubb, 2021) achieve the highest threshold for the toric code. Their runtime however increases quickly with the bond dimension that controls the accuracy of the approximation and they are difficult to parallelize compared to neural networks. Also, contrary to ML methods, they cannot adapt automatically to different noise models.

Several papers have investigated neural networks for quantum error correction, however none of them studies the problem from an equivariance lens. (Krastanov & Jiang, 2017) uses a fully connected architecture; (Wagner et al., 2020) imposes translation invariance by to zero-centering the syndrome and uses a fully connected layer on top; (Ni, 2020) uses a convolutional neural network which does not represent the right equivariance properties of the optimal decoder. Appendix 4.3 contains details of these architectures and the results obtained in these papers. (Meinerz et al., 2021) obtains the largest system size and threshold among neural decoders by combining a convolutional neural network backbone with a union find decoder. In our work we show that our model, which does not rely on a handcrafted decoder, achieves higher accuracy.

From the perspective of equivariant architectures (Cohen & Welling, 2016; Weiler & Cesa, 2021), our work studies a generalized form of equivariance, where the output representation depends on the values of the inputs to the neural network. To the best of our knowledge, this type of symmetry properties for a neural network have not been considered before.

Finally, neural decoders for classical error correction were discussed as a form of generalized belief propagation in (Satorras & Welling, 2021). However classical and quantum error correction are fundamentally different (Iyer & Poulin, 2013), and these results do not directly translate to the quantum case. See (Liu & Poulin, 2019) for an attempt which however does not achieve good accuracy for the toric code.

4.2 THE TORIC CODE

In this section we review the necessary background on the toric code. Recall that a qubit $|\psi\rangle$ is a superposition of 0 and 1 bits, which are denoted by $|0\rangle$ and $|1\rangle$: $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$. A quantum error correction code aims at correcting two types of errors on qubits: bit flip errors X, and phase flip errors Z, which act as: $X(\alpha\,|0\rangle + \beta\,|1\rangle) = \beta\,|0\rangle + \alpha\,|1\rangle$ and $Z(\alpha\,|0\rangle + \beta\,|1\rangle) = \alpha\,|0\rangle - \beta\,|1\rangle$. We also recall that the space of n qubits is that of superpositions of the 2^n possible bit strings of n bits. We denote by E_i an error that acts only on the i-th qubit. E_i can take four values: $\mathbf{1}, X, Z, XZ$, corresponding to no-error, bit-flip, phase-flip, or combined phase and bit flip. It turns out that the ability to correct these discrete set of errors is enough to correct general errors. We refer the reader to (Nielsen & Chuang, 2000) for details on quantum error correction.

4.2.1 ERROR PATHS

The toric code protects against errors by encoding logical qubits in topological degrees of freedom related to the non-contractible cycles of a torus (Kitaev, 2003). This is done as follows. We start by placing physical qubits on the edges of a $L \times L$ square lattice embedded on a torus. Errors are then associated with paths that traverse the edges corresponding to the qubits affected by errors. For reasons that will become clear later, we associate Z errors to paths on the lattice, and X errors to paths on the dual lattice. This is illustrated in figure 3. Here a represents a Z error on the edges traversed by the paths, while b represents a X errors on the edges traversed by the dual path.

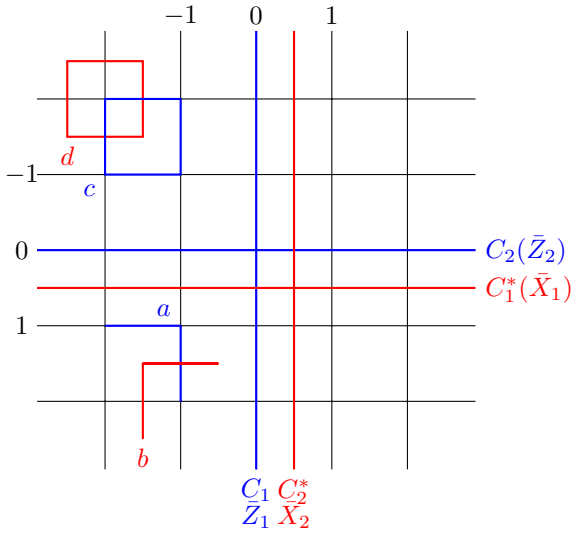


Figure 3: Toric code square lattice with periodic boundary conditions. Blue paths are Z errors, red paths on the dual lattice are X errors. c, d are X- and Z-stabilizers, while C_i, C_i^* are logical operators corresponding to non-contracbtle loops around the torus.

4.2.2 STABILIZERS AND CODE SPACE

We now consider certain combinations of bit and phase flips called X- and Z-stabilizers. For each plaquette of the lattice, we define a Z-stabilizers as the product of phase flips on the edges around the plaquette. Similarly, for each vertex of the lattice, a X-stabilizer is defined as the product of bit flips around a vertex. This is illustrated in figure 3 by the cycles c and d. Note that Z-stabilizers are not all independent. In fact if we take the product of two neighbouring plaquettes, the error on the shared edge disappears, since flipping twice is identical to no flipping: $Z^2 = \mathbf{1}$. If we take the product of Z-stabilizers over all the plaquettes, each edge is counted twice and so all errors disappear. This means that out of the L^2 Z-stabilizers, only $L^2 - 1$ are independent. Similarly, for X-stabilizers. The toric code is then defined as the subspace of the $2L^2$ qubits that is preserved by the stabilizer operators. Concreately, if $|\psi\rangle$ is a vector in the 2^{2L^2}-dimensional space of the physical qubits and S_i a stabilizer, the code subspace is defined by $S_i |\psi\rangle = |\psi\rangle$ for all i. Note that $S_i^2 = \mathbf{1}$ for each stabilizer, so S_i has ± 1 eigenvalues, and imposing the constraint $S_i |\psi\rangle = |\psi\rangle$ reduces the dimension of the space of the qubits by half. Since we have $2(L^2 - 1)$ independent stabilizers, the logical space has dimension $2^{2L^2}/2^{2(L^2-1)} = 2^2$, which means that the toric code encodes two logical qubits for any L. We thus see that an error-free code vector lives in a 4 dimensional vector space. If errors are introduced, this code vector will develop components in the complement of this code space. The goal of error correction is to find the most likely projection back onto the code subspace.

4.2.3 LOGICAL OPERATORS

We denote logical X and Z operators acting on the logical qubits by $\bar{X}_1, \bar{X}_2, \bar{Z}_1, \bar{Z}_2$. These operators are defined by the paths denoted by C_1^*, C_2^*, C_1, C_2 respectively in figure 3. To verify this statement, we need to check the commutation relations of these operators. First, we note that X and Z errors commute if they act on different qubits and anti-commute if they act on the same qubit: $XZ = -ZX$. Thus if we have a Z error string a and a X error string b, they commute if they cross an even number of times (so that we have an even number of -1's) and anti-commute if they cross an odd number of times (so that we have an odd number of -1's). For example, the errors represented by the paths a, b in figure 3 anticommute. We can then check that a X-stabilizer always commutes with a Z-stabilizer, since they always cross at either 0 or 2 edges. Similarly, we can check that logical operators commute with stabilizers for a similar reason, but are independent of them – i.e. they cannot be written as products of stabilizers – and thus preserve the logical space but act non-trivially on it, as required to logical operators. Also, we can check that $\bar{X}_i$ anti-commutes with $\bar{Z}_i$ for $i = 1, 2$ since they cross on

a single edge. We introduce the notation $\omega(E, E')$ to denote whether two errors E, E' anti-commute ($\omega(E, E') = 1$) or commute ($\omega(E, E') = 0$).

4.3 EQUIVARIANCE PROPERTY OF TORIC CODE DECODERS IN THE LITERATURE

4.3.1 CLASSICAL DECODERS

We first discuss classical, i.e. non-neural, decoders.

The maximum weight perfect matching decoder (MWPM) is the standard decoder for the toric code (Dennis et al., 2002). It treats X and Z syndromes independently and returns the minimum (Hamming) weight error consistent with the syndrome, a problem which can be solved using the Blossom algorithm in $O(n^3)$ time, but on average, it takes $O(n)$ time (Fowler et al., 2012). This decoder is popular because of its simplicity, but it has two main drawbacks: first, it treats X and Z error independently; second, it does not account for the degeneracy of errors due to the trivial action of the stabilizer (Duclos-Cianci & Poulin, 2010). Here we also show that it does not respect the equivariance properties of the maximum likelihood decoder under translations, see also (Wagner et al., 2020) where the authors point out that MWPM is not translation invariant. We note that the root cause for this failure is the ambiguity of minimum weight decoding for a string of syndromes, which is translation invariant, while the error string returned by the MWPM decoder is not, since it is obtained by breaking the ambiguity by an arbitrary choice, which is not modified after a translation. Note that degeneracy also can lead to a breaking of the symmetry in the maximum likelihood decoder. In fact, if two logical classes γ, γ' are such that $p(\gamma|\sigma) = p(\gamma'|\sigma)$ are this value is the largest logical probability, then it does not matter which one we return. This ambiguity can also lead to a non-translation equivariant result of the maximum likelihood decoder.

Now we discuss the union find decoder (Delfosse & Nickerson, 2021). Like MWPM, union find also treats X and Z independently – which leads to suboptimal decisions – and is a based on a two-stage process: first, during the syndrom validation step errors are mapped onto erasure errors, namely losses that occur for example when a qubit is reinitialized into a maximally mixed state; then, one applies the erasure decoder. The latter simply grows a spanning forest to cover the erased edges starting from the leaves, and flips qubits if it encounters a vertex with a syndrome. The syndrome validation step creates a list of all odd clusters, namely clusters with an odd number of non-trivial syndromes. This is done by growing odd clusters until two meet so that their parity will be even. We note that the syndrome validation step respects the symmetries of the square lattice as does the erasure decoder. The union find decoder d thus returns a recovery E for a syndrome σ so that $d(T\sigma) = Td(\sigma)$ for a translation T, leading to the right equivariance expected from a maximum likelihood decoder. This decoder is also very fast, practically $O(L^2)$, but the heuristics used leads to a suboptimal performance w.r.t. the MWPM decoder.

The tensor network decoder achieves state of the art results for the threshold probability of the toric code (Bravyi et al., 2014; Chubb, 2021). It does so by approximating directly the intractable sum over the stabilizer group that is involved in computing the logical class probabilities. The runtime is $O(n \log n + n\chi^3)$ where $n = L^2$ and χ is the so-called bond dimension, which is the number of singular values kept when doing an approximate contraction of tensors. Near the threshold we expect this to grow with the system size, but in practice modest values (e.g. $\chi = 6$ for the surface code in (Bravyi et al., 2014) with $L = 25$) give good results over a range of noise probabilities. The symmetries of the decoder will depend on the approximate contraction procedure. Those used in (Bravyi et al., 2014; Chubb, 2021) create a one dimensional matrix product state along a sweep line on a planar embedding of the Tanner graph of the code. This procedure breaks the translational invariance of the decoder due to the finite χ, and in these works it was applied only to the surface code, namely the code with boundaries. We believe that an equivariant contraction procedure might lead to an even more efficient tensor network decoder.

4.3.2 NEURAL DECODERS

(Krastanov & Jiang, 2017) introduces the machine learning problem of predicting the error given a syndrome with a neural network for the toric code. The architecture used is a fully connected network that does not model any symmetries of the toric code. It obtains threshold 16.4 and studies lattices up to $L = 9$.

(Wagner et al., 2020) explicitly investigates the role of symmetries for neural decoders. It uses a high level decoder architecture, where an input syndrome σ is first passed to a low level decoder which is not learnable and returns a guess for the error, $f(\sigma)$, which will correspond to a given logical class. The syndrome is also passed to a neural decoder that as in our setting predicts the logical probability. This is then combined with the underlying decoder to make a prediction for the error. In formulas, called the neural prediction $\hat{p}(\gamma|\sigma)$, the logical probabilities returned by the high level decoder is

$$\hat{\gamma}(\sigma) = \arg\max_{\gamma} \hat{p}(\gamma|\sigma) + \omega(L, f(\sigma))\,. \tag{6}$$

To take into account symmetries, the authors modify this setup by introduce a preprocessing step to deal with translations and mirror symmetries. For translations for example they define equivalence classes of syndromes related by translations. For each class they define an algorithm that centers the syndrome σ to pick a representative, say $[\sigma]$ and pass that as input to both the low level decoder and neural network. By construction, denoted by T the translation operator, one has $[\sigma T] = [\sigma]$. Then the output of the low level decoder is obtained by undoing the translation on the output of the low level decoder on $[\sigma]$. Let us call this modified low level decoder $\tilde{f}(\sigma)$ and the translation applied to produce the representative $T_\sigma : [\sigma] = \sigma T_\sigma$. Then $\tilde{f}(\sigma) = f([\sigma])T_\sigma^{-1}$ and this means that $\tilde{f}(\sigma)$ is translationally equivariant by construction: $\tilde{f}(\sigma T) = f([\sigma])T_{\sigma T}^{-1} = f([\sigma])(T^{-1}T_\sigma)^{-1} = \tilde{f}(\sigma)T$. The neural network has input $[\sigma]$ and the modified high level decoder used in this paper is:

$$\hat{\gamma}(\sigma) = \arg\max_{\gamma} \hat{p}(\gamma|[\sigma]) + \omega(L, \tilde{f}(\sigma))\,. \tag{7}$$

Note that we get the correct behavior under translations, see 4.2: (Note that T^{-1} appears w.r.t. equation 23 since we are considering the equation written as $p(\gamma|\sigma') = p(\gamma''|\sigma)$)

$$\begin{aligned}
\hat{\gamma}(\sigma T) &= \arg\max_{\gamma} \hat{p}(\gamma|[\sigma T]) + \omega(L, \tilde{f}(\sigma)T) && (8)\\
&= \arg\max_{\gamma} \hat{p}(\gamma|[\sigma]) + \omega(LT^{-1}, \tilde{f}(\sigma)) && (9)\\
&= \arg\max_{\gamma} \hat{p}(\gamma|[\sigma]) + \omega(L, \tilde{f}(\sigma)) + \rho_{\text{stab}}(T^{-1})\omega(H, \tilde{f}(\sigma)) && (10)\\
&= \hat{\gamma}(\sigma T) + \rho_{\text{stab}}(T^{-1})\sigma\,. && (11)
\end{aligned}$$

While the pipeline proposed in this paper is manifestly equivariant under translations, it requires additional computational cost to preprocess the data, and uses a fully connected network. Further, the authors could only show improvements w. r. t. MWPM decoder for $L = 3, 5, 7$, when using as underlying decoder MWPM itself, which adds additional runtime.

(Ni, 2020) implements a neural decoder for large distance toric codes $L = 16, 64$. The decoder is only tested for bit flip noise, where it performs on par, or lower, to MWPM. Large distance is achieved by using convolutional layers to downscale the lattice, in a similar fashion to a renormalization group decoder. The architecture is a a stack of CNN blocks each downsampling by half the lattice size, till the system has size 2×2. Downsampling is done by a convolutional layer with filter size $[2, 2]$ and stride $[2, 2]$. The output marginal probabilities for logical classes are then produced by a dense layer on the outputs the CNN blocks: $p(\gamma_0|\sigma), p(\gamma_1|\sigma)$ where $\gamma_i \in \mathbb{F}_2$ corresponds to acting with $\bar{X}_i$ or not. Note that the marginal probabilities will have a transformation law inherited by that of the joint, namely for translations $\rho_{\text{logi}}(T) = \mathbf{1}$, we have $p(\gamma_i + \rho_{\text{stab}}(T)_{i:}\sigma|\sigma T) = p(\gamma_i|\sigma)$. The authors did not discuss whether the architecture they propose has this symmetry property. We conjecture that the architecture in this paper does not have the right symmetry under translations. In fact, we expect that a CNN – the architecture proposed is a CNN apart from the periodic boundary conditions in the convolutions – can approximate only a translation invariant function, in our case $p(\gamma_i|\sigma T) = p(\gamma_i|\sigma)$, and a function with the equivariance properties required by the actual logical probabilities.

(Meinerz et al., 2021) uses a CNN backbone which processes patches of the lattice to produce the probability that the central qubit of the patch has an error, and then adds on top a union find decoder to deal with correlations beyond the size of the patch that the neural network sees. Using a CNN (and assuming periodic padding), the system is equivariant under translations, and so is the union find decoder, so the whole procedure amounts to a decoder $d(T\sigma) = Td(\sigma)$ for a translation T, leading to the right equivariance expected from a maximum likelihood decoder. While relying on the union find decoder for long range correlations allows one to scale to large lattices (up to $L = 255$), it also limits its accuracy, which leads to a threshold probability of 0.167.

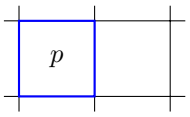
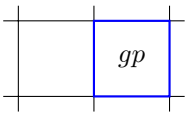
4.4 EQUIVARIANCE PROPERTIES

The goal of this section is to derive the equivariance properties of the toric code maximum likelihood decoder. To start, we define a symmetry of the code as the a transformation g that preserve the group of stabilizers, namely that acts as a permutation of the stabilizers. Since the logical subspace is defined by $S_i |\psi\rangle = |\psi\rangle$, $\forall i$, this definition is natural since the logical subspace does not change if we permute the stabilizers. We call the set of all code symmetries the automorphism group of the code.

If we denote with prime the transformed quantities, we have

$$S'_i = S_{gi} \,. \tag{12}$$

For example, if g is the horizontal translations of the lattice by one unit to the right, it acts on the Z stabilizers S^Z's as:

$S^Z_p =$  $\longrightarrow (S^Z)'_p =$  , (13)

and similarly for the X stabilizers S^X. We call the set of all code symmetries the automorphism group of the code.

The automorphism group of the toric code is generated by the symmetries of the square lattice, namely horizontal and vertical translations, $90°$ rotations and horizontal flips, together with the duality map which switches primal to dual lattice as well as X with Z. The left column of figure 4 shows the action of each of these symmetries on the vertices and plaquettes, defining the permutation of the associated stabilizers. Logical operators also need to be permuted among themselves up to stabilizers:

$$L'_a = L_{ga} \prod_{p \in \alpha^g_a} S^Z_p \prod_{v \in \beta^g_a} S^X_v \,, \tag{14}$$

where as above $L = (\bar{X}_1, \bar{X}_2, \bar{Z}_1, \bar{Z}_2)$, ga is a permutation of the four elements, and α^g_a and β^g_a are some g-dependent paths on the primal and dual lattice respectively. The right column of figure 4 shows the non-trivial action of the generators of the automorphism group of the toric code on the logical operators. For example, focusing on the rotation by $90°$ row, we see that ga acts as the permutation $(1234) \to (2143)$.

After discussing the symmetries of the toric code, we now consider the noise distribution. We call a transformation g a symmetry of the noise model if it leaves the noise distribution invariant: $p(E') = p(E)$. To present the equivariance result, we find it notationally convenient to see the probability $p(\gamma|\sigma)$ as the σ-dependent tensor $\boldsymbol{p}(\sigma)$ with 4 indices, $p(\sigma)_{\gamma_1,\gamma_2,\gamma_3,\gamma_4} = p(\gamma_1, \gamma_2, \gamma_3, \gamma_4|\sigma)$. The permutation part $a \to ga$ for $a = 1, 2, 3, 4$ of equation 14 acts on a tensor $t_{\gamma_1,\gamma_2,\gamma_3,\gamma_4}$ as the operator P_g:

$$(P_g \boldsymbol{t})_{\gamma_1,\gamma_2,\gamma_3,\gamma_4} = t_{\gamma_{g1},\gamma_{g2},\gamma_{g3},\gamma_{g4}} \,. \tag{15}$$

With α^g_a and β^g_a as in equation 14 we define the following quantity:

$$(\Delta_g \sigma)_a = \sum_{p \in \alpha^g_a} \sigma^Z_p + \sum_{v \in \beta^g_a} \sigma^X_v \tag{16}$$

with σ^Z (σ^X) the syndrome of S^Z (S^X). With these definitions, we are ready to enunciate the equivariance properties of the maximum likelihood decoder.

Theorem 4.1. *If g is a symmetry of the code and of the noise model, with action as in equation 12 and equation 14, then the logical probability tensor is invariant under the following action*

$$(\rho_g \boldsymbol{p})(\sigma) = M_g(\sigma) \boldsymbol{p}(g^{-1} \cdot \sigma) \,, \tag{17}$$

$$(g \cdot \sigma)_i = \sigma_{gi} \,, \quad M_g(\sigma) = P_g^{-1} R^g_1(\sigma) R^g_2(\sigma) R^g_3(\sigma) R^g_4(\sigma) \,, \tag{18}$$

where $R^g_a(\sigma)$ acts as identity if $\Delta_g(g^{-1} \cdot \sigma)_a = 0 \mod 2$ and as the flip $t_{\dots\gamma_a\dots} \mapsto t_{\dots(1-\gamma_a)\dots}$ if $\Delta_g(g^{-1} \cdot \sigma)_a = 1 \mod 2$. P_g and $\Delta_g(\sigma)$ are defined in equation 15 and equation 16.

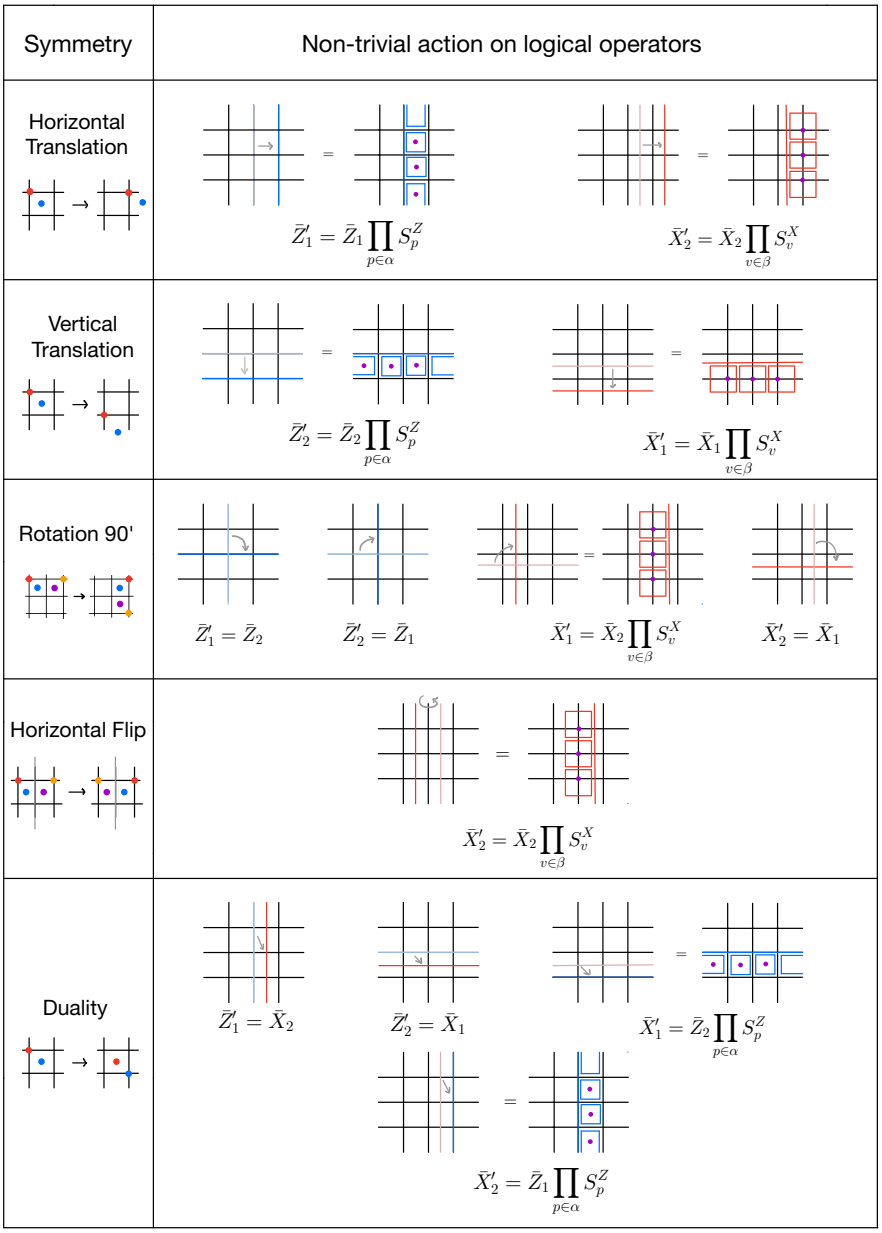


Figure 4: Left column: the list of symmetries of the toric code decoder and their action on the vertices and plaquettes. Right column: non-trivial action of those symmetries on the logical operators. Purple dots indicate the paths α, β in the formulas below the pictures. We assume odd L and rotations are performed around the center vertex of the lattice, while horizontal flips are done around the vertical middle line.

Proof. See Appendix 4.5 □

As a corollary of theorem 4.1, we see that the symmetries of the toric code discussed above (translations, rotations, mirrors and duality) are also symmetries of the maximum likelihood decoder when we have the depolarizing noise of equation 3.

Example 4.1. For concreteness, we here give the explicit formulas for the transformation $M_g(\sigma)$ in the case of translations. Referring then to figure 4, if g is the horizontal translation by one unit to the right, then P_g, R_1^g, R_4^g act as identity – recall that R_1^g, R_4^g are associated to $\bar{X}_1, \bar{Z}_2$ which do not change. Let us now introduce coordinates on the lattice such that $v = (0, 0)$ is the middle vertex (assuming L odd for simplicity), and label other vertices with numbers increasing to the right and bottom, as in figure 3. We also label the plaquette neighboring a vertex (i, j) to its bottom-right as $(i + \frac{1}{2}, j + \frac{1}{2})$. Then we have explicitly,

$$\Delta_g(g^{-1} \cdot \sigma) = \Big(0, \sum_{i=0}^{L-1} \sigma^X_{i,0}, \sum_{i=0}^{L-1} \sigma^Z_{i+\frac{1}{2},-\frac{1}{2}}, 0\Big) . \tag{19}$$

where the coordinates are understood modulo L. Then R_a^g acts as $t_{\dots\gamma_a\dots} \mapsto t_{\dots(1-\gamma_a)\dots}$ if $\Delta_g(g^{-1} \cdot \sigma)_a = 1 \mod 2$ and as identity $\Delta_g(g^{-1} \cdot \sigma)_a = 0 \mod 2$. Similarly, if g is the vertical translation by one unit to the bottom, we have that P_g is identity and the action of R_a^g is read off from:

$$\Delta_g(g^{-1} \cdot \sigma) = \Big(\sum_{j=0}^{L-1} \sigma^X_{0,j}, 0, 0, \sum_{j=0}^{L-1} \sigma^Z_{-\frac{1}{2},j+\frac{1}{2}}\Big) . \tag{20}$$

Still referring to figure 4, it is also clear that translations by more than one unit will involve sums over syndromes associated to more than one row or column. For example, if g is the vertical translation by two units to the bottom,

$$\Delta_g(g^{-1} \cdot \sigma) = \Big(\sum_{i=-1}^{0} \sum_{j=0}^{L-1} \sigma^X_{i,j}, 0, 0, \sum_{i=-1}^{0} \sum_{j=0}^{L-1} \sigma^Z_{i-\frac{1}{2},j+\frac{1}{2}}\Big) . \tag{21}$$

Translating by L units to the bottom or to the right is the same as no translations. In our formalism this follows from the fact that there exists an error E such that:

$$\sum_{i,j=0}^{L-1} \sigma^X_{ij} = \sum_{i,j=0}^{L-1} \omega(E, S^X_{ij}) = \omega\Big(E, \prod_{ij} S^X_{ij}\Big) = 0 . \tag{22}$$

The first equality is the definition of syndrome, the second uses the fact that $\omega(E, FG) = \omega(E, F) + \omega(E, G) \mod 2$, and the third uses that the product of X stabilizers across all vertices is the identity, as remarked in section 4.2.2. The same argument applies to σ^Z and Z stabilizers.

4.5 Proof of theorem 4.1

To prove theorem 4.1, we shall first establish the following proposition which shows the transformation of the components of the maximum likelihood decoder.

Proposition 4.2. *If g is a symmetry of the code and noise model, then for all $\gamma \in \mathbb{F}_2^{2k}, \sigma \in \mathbb{F}_2^{n-k}$, we have $p(\gamma|\sigma) = p(\gamma'|\sigma')$, with*

$$\sigma' = g^{-1} \cdot \sigma \tag{23}$$

$$\gamma' = \rho^{-1}_{logi}(g)(\gamma + \Delta_g(\sigma') \mod 2) , \tag{24}$$

where ρ_{logi} is the permutation representation of the logical operators in equation 14 and Δ_g is defined in equation 16.

Proof. If we denote by $\pi(g)$ the action of a symmetry on the error E, since $\omega(E, FG) = \omega(E, F) + \omega(E, G) \mod 2$, and $p(E) = p(\pi(g)E)$ by assumption, we have $\omega(\pi(g)E, \pi(g)F) = \omega(E, F)$,

so:

$$\begin{aligned} p(\gamma,\sigma) &= \sum_{E\in\mathcal{P}} p(E)\,\delta(\omega(E,S),\sigma)\delta(\omega(E,L),\gamma) \\ &= \sum_{E\in\mathcal{P}} p(\pi(g)E)\,\delta(\omega(\pi(g)E,\pi(g)S),\sigma)\delta(\omega(\pi(g)E,\pi(g)L),\gamma) \\ &= \sum_{E\in\mathcal{P}} p(E)\,\delta(\omega(E,\pi(g)S),\sigma)\delta(\omega(E,\pi(g)L),\gamma) \\ &= \sum_{E\in\mathcal{P}} p(E)\,\delta(\omega(E,S),g^{-1}\cdot\sigma)\delta(\rho_{\text{logi}}(g)\omega(E,L)+\Delta_g(\omega(E,S)),\gamma) \\ &= p(\gamma',\sigma')\,. \end{aligned}$$

In the third to last equality we relabeled $\pi(g)E$ with E since $\pi(g)$ is an invertible transformation on the set of Pauli operators and thus acts as a permutation of the Pauli errors. In the second to last equality we used the transformation laws of S and L, equation 12 and equation 14.

The probability $p(\gamma|\sigma)$ has the same symmetry since $p(\gamma|\sigma) = p(\gamma,\sigma)/p(\sigma)$ and the denominator $p(\sigma)$ is invariant: $p(\sigma) = \sum_\gamma p(\gamma,\sigma) = \sum_\gamma p(\gamma',\sigma') = \sum_{\gamma'} p(\gamma',\sigma') = p(\sigma')$. □

The theorem 4.1 follows by noting that the map $p(\gamma,\sigma) \mapsto p(\gamma',\sigma')$ can be written as the operator $P_g^{-1}R_1R_2R_3R_4$ defined in theorem 4.1 acting on the tensor $\boldsymbol{p}(\sigma)$. Indeed the map $p(\gamma_1,\gamma_2,\gamma_3,\gamma_4) \mapsto p(\rho_{\text{logi}}(g)\gamma) = p(\gamma_{g1},\gamma_{g2},\gamma_{g3},\gamma_{g4})$ can be written as P_g acting on the tensor $\boldsymbol{p}$, with P_g explicitly in Dirac notation:

$$P_g = \sum_\gamma |\gamma_1,\gamma_2,\gamma_3,\gamma_4\rangle\langle\gamma_{g1},\gamma_{g2},\gamma_{g3},\gamma_{g4}| = \sum_\gamma \left|\gamma_{g^{-1}1},\gamma_{g^{-1}2},\gamma_{g^{-1}3},\gamma_{g^{-1}4}\right\rangle\langle\gamma_1,\gamma_2,\gamma_3,\gamma_4|\,. \tag{25}$$

This is the same object introduced in equation 15. It is a representation of the symmetric group of 4 elements: $P_gP_h = P_{gh}$. The map $p(\gamma) \mapsto p(\gamma + \Delta_g(\sigma) \mod 2)$ can be written as the following operator acting on tensor $\boldsymbol{p}$:

$$\bigotimes_{a=1}^{4}\left[\delta_{\Delta_g(\sigma)_a,0}\mathbf{1}_2 + \delta_{\Delta_g(\sigma)_a,1}X\right]\,, \tag{26}$$

with X the Pauli X. This proves the form of the operator $P_g^{-1}R_1\cdots R_4$ introduced in 4.1.

4.6 Group homomorphism property of the representation ρ

If we denote $\boldsymbol{f}'(\sigma) = (\rho_h\boldsymbol{f})(\sigma) = M_h(\sigma)\boldsymbol{f}(h^{-1}\cdot\sigma)$, the condition $\rho_g\rho_h = \rho_{gh}$, means that we have:

$$(\rho_g\rho_h\boldsymbol{f})(\sigma) = M_g(\sigma)\boldsymbol{f}'(g^{-1}\cdot\sigma) = M_g(\sigma)M_h(g^{-1}\cdot\sigma)\boldsymbol{f}(h^{-1}g^{-1}\cdot\sigma)\,, \tag{27}$$

which needs to equal $M_{gh}(\sigma)\boldsymbol{f}((gh)^{-1}\cdot\sigma)$, that is

$$M_{gh}(\sigma) = M_g(\sigma)M_h(g^{-1}\cdot\sigma)\,. \tag{28}$$

This is a necessary condition for ρ to be a well defined action.

4.7 Implementation details for the translation group

We shall now discuss some details of the construction of proposition 3.1 for the translation group. We index elements of the translation group $\mathbb{Z}_L^{\times 2}$ as $g = (i,j)$ indicating a translation to the right by i and to the bottom by j. $\boldsymbol{\phi}$ is then a standard translation-equivariant convolutional neural network:

$$\phi((-i,-j)\cdot\sigma)_{k,l,\gamma} = \phi(\sigma)_{k+i,l+j,\gamma}\,. \tag{29}$$

From equation 4 with $g = (i,0)$, $h = (0,j)$, we have

$$M_{(i,j)}(\sigma) = M_{(i,0)}(\sigma)M_{(0,j)}((-i,0)\cdot\sigma) \tag{30}$$

$$= M_{(i,0)}(\sigma)M_{(0,j)}(\sigma) \tag{31}$$

where the second equality follows from the fact that $M_{(0,j)}(\sigma)$ depends on σ only through sums along rows which are invariant under horizontal translations. We can then consider the horizontal and vertical translations separately. Setting $g = (i-1, 0)$ and $h = (1, 0)$ in equation 4 we get a recursion relation

$$M_{(i,0)}(\sigma) = M_{(i-1,0)}(\sigma) M_{(1,0)}((-i+1, 0) \cdot \sigma) \,. \tag{32}$$

We discussed explicitly $M_{(1,0)}(\sigma)$ in example 4.1. $M_{(1,0)}((i,0)\cdot\sigma)$ involves the sum of the syndrome over the $i+1$-th column of vertices or plaquettes – when starting counting from the middle, as in figure 3 – and can be precomputed for all i by summing along the columns of the matrices σ^X and σ^Z. Therefore we can compute $M_{(i,0)}(\sigma)$ from $M_{(i-1,0)}(\sigma)$ in $O(1)$ time. A similar procedure allows us to compute $M_{(0,j)}(\sigma)$ so that the summation in equation 5 can be computed efficiently in $O(L^2)$.

Since our experiments focus on the translation symmetry, we refrain from discussing here details of the implementation of the other symmetries of section 4.4.

4.8 Experiments details

Ablation	L	p : 0.155	0.166	0.178
(7,32,AP)	7	0.13(0.01)	-	-
(7,32,FC)	7	0.62(0.05)	0.51(0.05)	-
(15,64,FC)	15	-	0.21(0.02)	-
(17,64,FC)	17	-	0.06(0.02)	-
(19,64,FC)	17	-	0.1(0.03)	-

Table 2: Ablation study for *END* decoder. We used same body architecture and training procedure, but the Fully Connected (FC) or Average pooling (AP) projection from feature to the space of logits instead of the twisted global average pooling introduced in section 3.2.

Architecture We adapt the wide-resnet (WR) architecture zagoruyko2016wide: each convolution is defined to have periodic boundaries. WR consists of 3 blocks, where the depth of each block was 3 and fixed across all experiments. We vary the number of channels in the blocks: $(\text{ch}, 64, 64)$, $\text{ch} \in \{32, 64, 128\}$. Inside each block we used the GeLU hendrycks2016gaussian activation function. As initialization we used kaiming for leaky ReLU. As normalisation layer, we used standard batch-norm.

Sampling noise channel For performance tests of neural decoders we used standard NumPy random generator. During training we used Quasi-Monte Carlo generator based on Sobolev Sequence. This does not provide any gain in terms of performance overall, but we found it to stabilise training. Both for training and performance evaluation batches were generated on the fly.

Training hyperparameters We used AdamW optimiser loshchilovdecoupled for all experiments. In order to avoid manual tuning of schedule and learning rate, we used 1cycle approach smith2019super. Typical maximal learning rate was 0.01 for batch 512 and 0.03 for batch 2048. For the ablation studies we also tried reduce on plateau and cosine annealing, however this doesn't produce consistent effects for lattice size bigger than 7.

4.9 Threshold plot

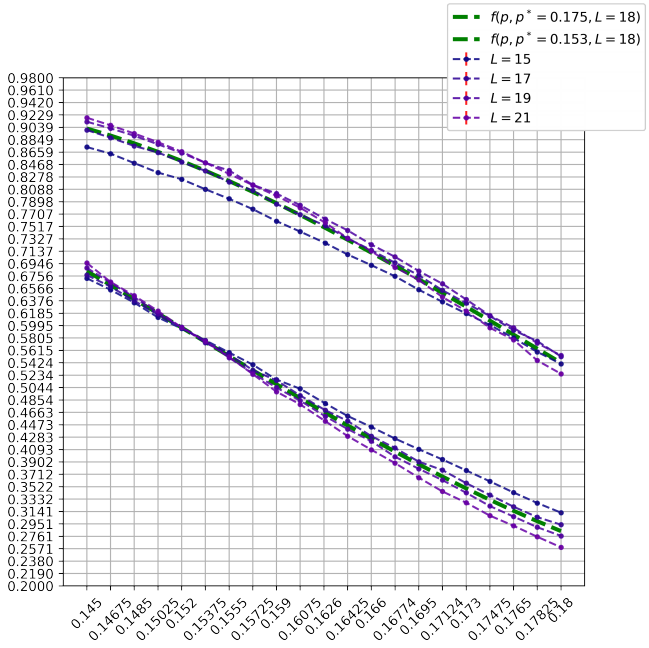


Figure 5: Logical accuracy of MWPM and *END* decoder. *END* decoder has threshold ≈ 0.17.