Teaching small transformers to rewrite ZX diagrams

Richie Yeung
Quantinuum
Oxford University

Konstantinos Meichanetzidis
Quantinuum

Alexandre Krajenbrink
Quantinuum

François Charton
Meta

Abstract

ZX calculus is a graphical language for reasoning about linear maps. Maps are represented as graphs, and reasoning amounts to graph rewrites. The main applications of ZX calculus are in quantum computation. We train small transformers to simplify ZX graphs, i.e. perform resource optimisation of quantum circuits. Preliminary experiments show that transformers can be trained to simplify CNOT and Clifford circuits with high accuracy. These are the simplest kinds of ZX graphs, in the sense that there exists an efficient rewrite strategy. We also show evidence that transformers learn to simplify the more complex Clifford + T graphs, for which in general there does not exist an efficient simplification algorithm.

Transformers [20] have proven their worth on problems of symbolic mathematics, such as integration [16], theorem proving [18], symbolic regression [6] and SAT solving [19]. In this paper, we consider the optimization problem of quantum circuit compilation. Quantum computer programs are composed as sequences of quantum assembly instructions known as gates, which are often depicted using the circuit model. These circuits can be converted to *graph-like ZX diagrams* [8], which can be represented as a labelled graph. Thanks to the ZX calculus [5], which is equipped with rewrite rules that can transform between equivalent ZX diagrams, the optimization problem of minimizing resources on quantum circuits can be framed as a graph rewriting problem.

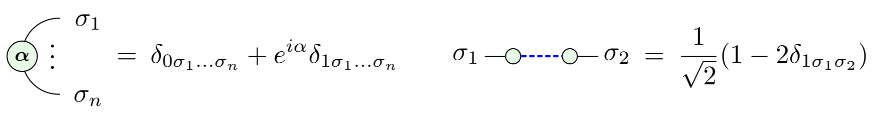
The optimal compilation of quantum circuits involves solving hard problems: checking whether two quantum circuits are exactly equivalent is QMA-complete [12] in the general case and the optimisation of the makespan (or duration) of a quantum circuit is NP-complete [4]. Meanwhile, there is a rich literature of using machine learning to tackle NP-hard problems [9]. In this paper, we train transformers to learn the normal form of two classes of quantum circuits, CNOT and Clifford circuits, that can be efficiently simulated in polynomial time [1]. Specifically, we learn the `full_reduce` procedure from the `pyzx` library, which converts a quantum circuit into a graph-state with local Cliffords, an intermediate representation that is also naturally amenable to measurement-based quantum computation. The procedure can also receive Clifford + T circuits, an approximately universal fragment of quantum circuits, though the output is not a normal form as the problem of reducing the number of T gates in a quantum circuit is hard [2, 11]. By learning this function on Clifford + T circuits, we effectively learn the “phase-teleportation" routine used by Kissinger and Wetering to perform T-count reduction [15].

This is a first step towards developing transformer-based compilers for quantum computers using methods inspired by machine translation.

37th Conference on Neural Information Processing Systems (NeurIPS 2023) Workshop on MATH-AI.

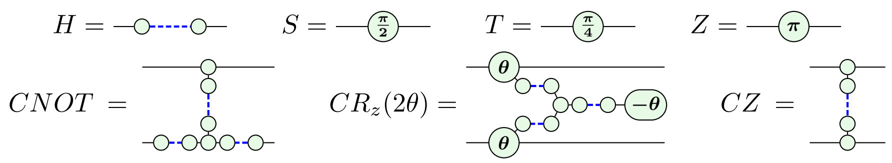
1 ZX Calculus

The circuit model of quantum computation enables the graphical expression of quantum states and unitary evolutions, which can be more verbosely represented as vectors in $\mathbb{C}^{2^n}$ and matrices in $\mathbb{C}^{2^n \times 2^n}$, respectively. They can be represented as ZX diagrams which are composed of tensors called the Z and X 'spiders'. X spiders can actually be decomposed into Z spiders with a Hadamard gate on each leg of the spider, so can be regarded as "syntactic sugar". In this work, we will present the *graph-like ZX calculus*. The semantics of the Z spider and the Hadamard gate using tensor index notation are:

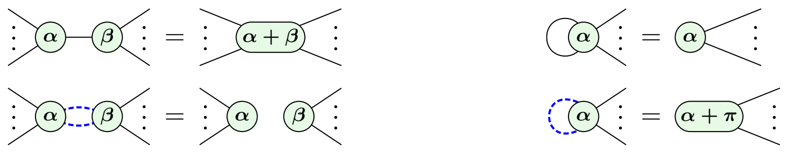


$$= \delta_{0\sigma_1\ldots\sigma_n} + e^{i\alpha}\delta_{1\sigma_1\ldots\sigma_n} \qquad = \frac{1}{\sqrt{2}}(1 - 2\delta_{1\sigma_1\sigma_2})$$

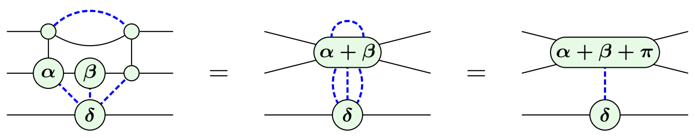
Spiders with 0 phase modulo 2π may be written without the phase label. Joining two wires corresponds to equating two indices and summing over them (tensor contraction). This allows us to express common quantum gates in terms of ZX graphs in a fine-grained way:



The ZX calculus comes equipped with a set of local rewrite rules allowing reasoning about ZX graphs, which allow one to transform, or *rewrite*, between ZX graphs that have an equal matrix representation. Given a rule $g \rightsquigarrow h$, rewriting entails identifying a subgraph g and replacing the subgraph with h. The rules are *sound* and *complete*, so graphs can be rewritten into one another if and only if they have the same underlying matrix representation. For example, the 'fusion' rule dictates that two Z spiders joined by a black wire can be combined into one Z spider whose phase is the sum of the phases of the two spiders. The rules are:



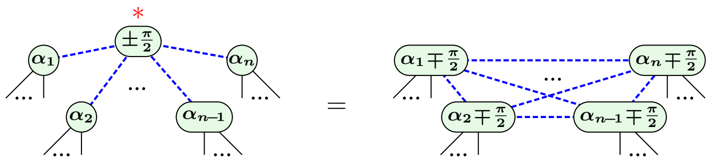
Using these rewrite rules, any graph-like ZX diagram can be converted to a *simple* graph containing no graph loops or parallel edges. Furthermore, all vertices would be Z spiders and all edges would be Hadamard edges (blue), except on the boundary where edges can be plain (black). Left and right boundary edges correspond to the columns and rows of the matrix represented by a ZX diagram. Here is an example of a ZX graph converted into a simple ZX graph:



2 Task Description

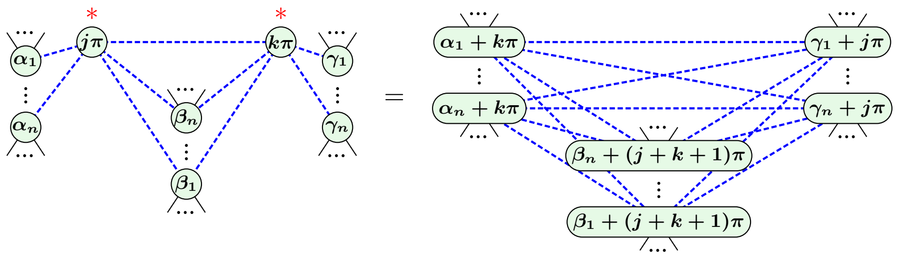
Duncan et al. [8] describe an efficient deterministic algorithm for rewriting ZX graphs, which was further developed by Kissinger and Van de Wetering [15] to an algorithm for reducing the number of T gates in a circuit. The key rewrite rules used in the algorithm are called 'local complementation' and 'pivoting'. The simplification algorithm exhaustively matches and applies the two rules within the graph. Since both rewrite rules strictly decrease the number of Z spiders, the algorithm is guaranteed to terminate in polynomial time in the size of the graph.

Lemma 1 *(Local Complementation) The following rule holds in the ZX-calculus:*


*
$\pm\frac{\pi}{2}$
α_1
α_2
α_{n-1}
α_n
=
$\alpha_1 \mp \frac{\pi}{2}$
$\alpha_n \mp \frac{\pi}{2}$
$\alpha_2 \mp \frac{\pi}{2}$
$\alpha_{n-1} \mp \frac{\pi}{2}$

where the RHS is obtained from the LHS by toggling all of the connections between the neighbouring vertices of the marked vertex, removing the marked vertex, and updating the phases as shown.

Lemma 2 *(Pivoting) The following rule holds in the ZX-calculus:*



where the RHS is obtained from the LHS by first pairwise-toggling the connections between the exclusive neighbourhood the first marked vertex $\{\alpha_i\}$, the exclusive neighbourhood the second marked vertex $\{\gamma_i\}$, and the shared neighbourhood of the two marked vertices $\{\beta_i\}$, then removing the marked vertices and updating the phases as shown.

In this work, we focus on the task of simplifying quantum circuits from three families: CNOT circuits (generated from $\{CNOT\}$), Clifford circuits (generated from $\{CNOT, H, S\}$), and Clifford + T circuits (generated from $\{CNOT, H, T\}$). Note that the fusion of two T gates leads to a S gate. When represented as ZX graphs, the spiders of CNOT circuits are decorated with phases that are multiples of π. For Clifford and Clifford + T circuits, the phases are multiples of $\frac{\pi}{2}$ and $\frac{\pi}{4}$ respectively.

While CNOT and Clifford circuits can be efficiently simplified using local complementation and pivoting [8], the task of optimally compiling these circuit classes is still important for near-term quantum processors, as the noisiest gates are the 2-qubit entangling CNOT gates. The Clifford + T gateset is approximately universal for quantum computation, and so it is not expected that there exists an efficient algorithm to simplify any given Clifford + T ZX graph; if there were such an algorithm, that would imply that it would be possible to efficiently and exactly simulate any quantum computation, which is not believed to be the case [3, 7]. Still, simplification of the Clifford part of Clifford + T ZX graphs can lead to cancellation of T-gates and hence in T-count reduction [15], which is important for the fault-tolerant regime of quantum processors, where the T-gates dominate the cost of quantum resources [10, 17].

3 Model settings

Generating datasets. All datasets are generated using the Python `pyzx` library [14]. Random circuits C are generated (either CNOT, Clifford, or Clifford + T), for a given depth d and number of qubits q (typically, $d = 15$ and $q = 10$), and reduced into circuit R using the function `full_reduce`, which implements the simplification strategy using local complementation and pivoting. The model is then trained to reduce C into R (or an equivalent reduced circuit).

Encoding circuits. A circuit C is encoded by enumerating the number of nodes and edges, and then listing the type and phase of all nodes, and the type and nodes of all edges. There are 2 possible edge types, 4 possible node types, 8 possible phases. Our total vocabulary has no more than 100 tokens. Random circuits C with 10 qubits and depth 15 are encoded as sequences of up to 360 tokens, and their reduced form R as sequence of up to 210 tokens. Table A in Appendix A presents the maximal length of the sequences representing C and R, for different values of q and d.

Models. We train sequence-to-sequence transformers [20] to minimize the cross-entropy between model predictions of R and the correct solutions. We use models with 6 or 8 layers in the encoder and decoder, 512 dimensions and 8 attention heads. The optimizer is Adam [13] with a learning rate of 10^{-4}. For models with 8 layers or more, we use linear warmup over the first 10,000 optimization steps, and inverse square root scheduling of the learning rate. Note that we are attempting to learn a procedure with $O(|V|^3)$ time complexity using a model which has $O(d \cdot n^2 + n \cdot d^2)$ time complexity, where $n \approx |V|$ is the sequence length and d is the embedding size. In the worst case, the sequence length can be quadratic in the number of vertices for a fully connected graph.

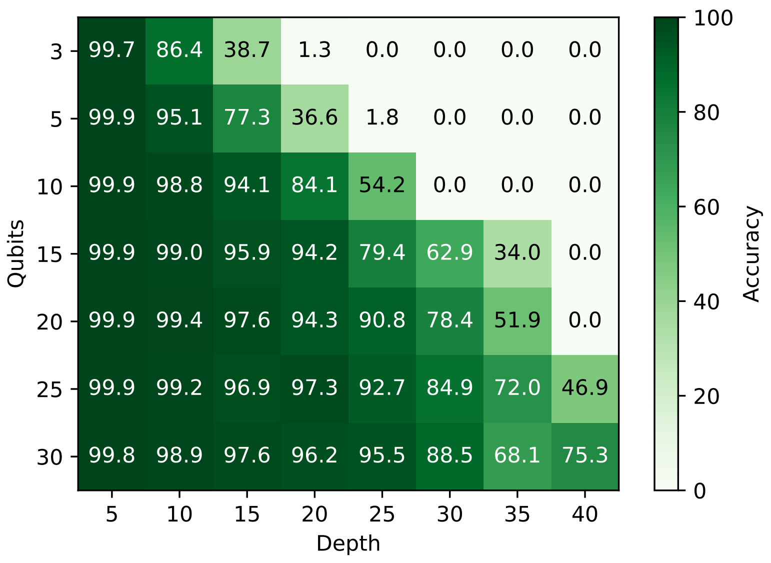
Evaluation. Trained models are evaluated on a held-out test set, generated with the same techniques and parameters as the training set, at the end of every epoch (300,000 examples). Whereas the mathematical signification of model input and output is not considered during training (only the correctness of predicted tokens matters), during evalutation the input and output sequences are decoded, as circuits I and O. The *syntactic correctness* of O (that it is a valid graph) is verified. Then we verify that the graph $I + \text{Adjoint}(O)$ reduces to identity, i.e. that O is a reduction of I, and that O has less nodes than the full reduction of I (as computed by `full_reduce`, and found in the test set).

4 Results

In a first set of experiments, we investigate the capability of Transformers to reduce Clifford circuits. On this category of graphs, a polynomial time solution exists, and `pyzx` `full_reduce` returns a minimal graph. A 6-layer transformer, with 512 dimensions and 8 attention heads, trained on circuits with 10 qubits and a depth of 15 achieves 96.4% accuracy after 770 epochs (231 million examples). In 88.4% of test cases, the circuit predicted is the output of `full_reduce`. In 8%, it is an equivalent full reduction, with the same number of nodes. Model accuracy goes down as circuits become deeper: 89.1% for circuits of depth 20, 63.2% for depth 25, and 29.5% for depth 30.

Larger models bring little gain in accuracy: 8-layer transformers with 640 dimensions and 10 attention heads achieve 95.4, 81.1 and 49.6% accuracy on circuits with 10 qubits and depth 15, 20 and 25. On the other hand, 4-layer models only achieve 92.4, 71.9 and 35.7% accuracy. Table 1 presents results for different depths and number of qubits (for 6 and 8-layer models). As a rule, models perform better for circuits with large number of qubits, and worse as depth increases.

Table 1: **Model accuracy for different depths and number of qubits.** Best of three 6 and 8-layer models.


Qubits \ Depth	5	10	15	20	25	30	35	40
3	99.7	86.4	38.7	1.3	0.0	0.0	0.0	0.0
5	99.9	95.1	77.3	36.6	1.8	0.0	0.0	0.0
10	99.9	98.8	94.1	84.1	54.2	0.0	0.0	0.0
15	99.9	99.0	95.9	94.2	79.4	62.9	34.0	0.0
20	99.9	99.4	97.6	94.3	90.8	78.4	51.9	0.0
25	99.9	99.2	96.9	97.3	92.7	84.9	72.0	46.9
30	99.8	98.9	97.6	96.2	95.5	88.5	68.1	75.3

Qubits
Depth
Accuracy
100
80
60
40
20
0

In a second set of experiments, we compare 8-layer transformers performing `full_reduce` on three classes of circuits: CNOT, Clifford, and Clifford + T. Performance is better for CNOT than Clifford (unsurprisingly: CNOTs are included in Cliffords). For Clifford + T our models achieve over 90% accuracy for all depths. These results suggest that transformers can deal with complex ZX circuits.

Table 2: **Model accuracy for different depths and circuit types.** 8-layer models. 10 qubit circuits.

Depth	CNOT	Clifford	Clifford + T
15	98.8	94.0	98.6
20	95.8	79.3	95.1
25	90.1	53.7	90.1

5 Discussion and future work

These initial experiments show that ZX graphs rewriting can be learned by transformers trained from generated data, in the two simple cases (CNOT and Clifford) where an algorithm for calculating the optimal solution exists (`full_reduce`). For the Clifford + T graphs, such an algorithm does not exist, but our models can learn a "partial" reduction from a random graph into an equivalent, simpler, graph. This suggests that the same techniques can be applied in the general case, where no polynomial reduction algorithm is known.

In the general case, models would be trained to perform partial reductions, and trained models would be called iteratively, or used in a reinforcement learning framework. Training sets for partial reduction can be built by applying reduction rules, or their inverses, to a random graph, in a process reminiscent of a random walk over equivalent ZX graphs. Transformers would then learn to translate the largest graph into the smallest. We leave these experiments for future work.

References

[1] Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70(5):052328, 2004.

[2] Matthew Amy and Michele Mosca. T-count optimization and reed–muller codes. *IEEE Transactions on Information Theory*, 65(8):4771–4784, 2019.

[3] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26(5):1411–1473, 1997.

[4] Adi Botea, Akihiro Kishimoto, and Radu Marinescu. On the complexity of quantum circuit compilation. In *Proceedings of the International Symposium on Combinatorial Search*, volume 9, pages 138–142, 2018.

[5] Bob Coecke and Ross Duncan. Interacting quantum observables: categorical algebra and diagrammatics. *New Journal of Physics*, 13(4):043016, 2011.

[6] Stéphane d'Ascoli, Pierre-Alexandre Kamienny, Guillaume Lample, and François Charton. Deep symbolic regression for recurrent sequences, 2022.

[7] M. Van den Nest. Classical simulation of quantum computation, the gottesman-knill theorem, and slightly beyond, 2009.

[8] Ross Duncan, Aleks Kissinger, Simon Perdrix, and John Van De Wetering. Graph-theoretic simplification of quantum circuits with the zx-calculus. *Quantum*, 4:279, 2020.

[9] Alhussein Fawzi, Mateusz Malinowski, Hamza Fawzi, and Omar Fawzi. Learning dynamic polynomial proofs. *Advances in Neural Information Processing Systems*, 32, 2019.

[10] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Phys. Rev. A*, 86:032324, Sep 2012.

[11] Luke E Heyfron and Earl T Campbell. An efficient quantum compiler that reduces t count. *Quantum Science and Technology*, 4(1):015004, 2018.

[12] Dominik Janzing, Pawel Wocjan, and Thomas Beth. " non-identity-check" is qma-complete. *International Journal of Quantum Information*, 3(03):463–473, 2005.

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] Aleks Kissinger and John van de Wetering. Pyzx: Large scale automated diagrammatic reasoning. *arXiv preprint arXiv:1904.04735*, 2019.

[15] Aleks Kissinger and John van de Wetering. Reducing t-count with the zx-calculus. *arXiv preprint arXiv:1903.10477*, 2019.

[16] Guillaume Lample and François Charton. Deep learning for symbolic mathematics. In *International Conference on Learning Representations*, 2020.

[17] Daniel Litinski. A game of surface codes: Large-scale quantum computing with lattice surgery. *Quantum*, 3:128, mar 2019.

[18] Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving, 2020.

[19] Feng Shi, Chonghan Lee, Mohammad Khairul Bashar, Nikhil Shukla, Song-Chun Zhu, and Vijaykrishnan Narayanan. Transformer-based Machine Learning for Fast SAT Solvers and Logic Synthesis. *arXiv preprint arXiv:2107.07116*, 2021.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

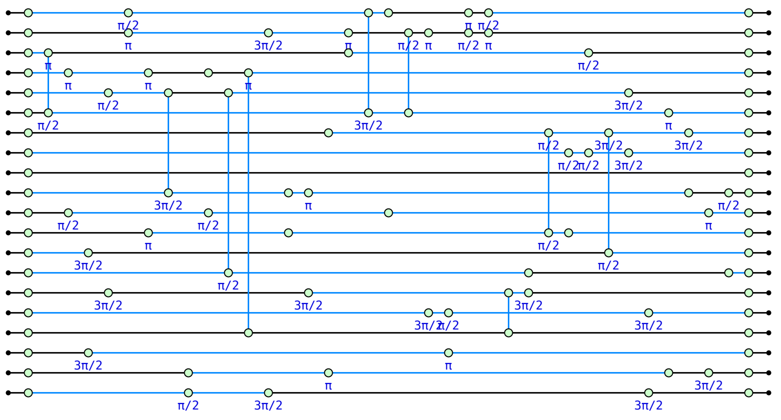
A Encoding length

	depth							
qubits	5	10	15	20	25	30	35	40
3	120 / 75	180 / 85	240 / 85	300 / 85	355 / 85	415 / 85	470 / 85	530 / 85
5	155 / 100	215 / 125	275 / 150	330 / 160	390 / 170	450 / 170	505 / 170	565 / 170
10	240 / 140	300 / 180	360 / 210	415 / 245	475 / 285	535 / 325	590 / 360	650 / 400
15	325 / 175	385 / 220	445 / 260	500 / 290	560 / 330	620 / 360	675 / 410	735 / 465
20	410 / 215	490 / 260	530 / 300	590 / 335	645 / 365	700 / 400	760 / 435	820 / 485
25	495 / 250	555 / 300	615 / 340	690 / 375	730 / 410	790 / 445	845 / 475	905 / 510
30	580 / 285	640 / 335	700 / 380	760 / 415	815 / 455	875 / 485	930 / 520	990 / 550

Table 3: **Graph encoding sequence lengths for different qubits and depths.** Maximum input (random) / output (reduced) lengths, 99-th percentile of all graphs generated.

B `full_reduce` Example

Before `full_reduce` (150 vertices, 139 edges)



After `full_reduce` (73 vertices, 71 edges)

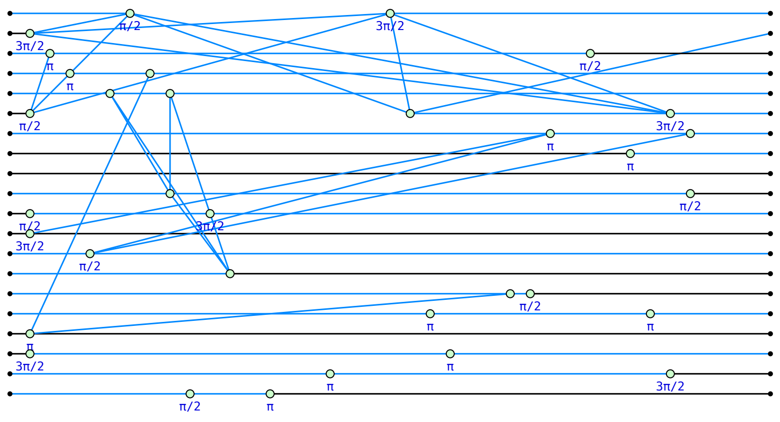


Figure 1: A randomly generated 20 qubit, 35 depth Clifford circuit, before and after the `full_reduce` procedure.