

Graph Neural Networks on Quantum Computers: a Trainability and Classical Simulability Analysis *

Theodor Iosif^{12†} Paul San Sebastian³⁴ Tilen Gaetano Limbäck-Stokin¹ Kin Ian Lo¹
Yidong Liao⁵

¹*Quantum Learning Labs, Department of Computer Science, University College London, London, UK*

²*London Centre for Nanotechnology, London, UK*

³*Ikerlan Technology Research Centre, Basque Research and Technology Alliance, Mondragón, Spain*

⁴*University of the Basque Country/Euskal Herriko Unibertsitatea, Santsoena, Spain*

⁵*Laboratoire d’Informatique de Paris 6, CNRS, Sorbonne Université, Paris, France*

Abstract. Graph Neural Networks (GNNs) have shown promise in solving complex graph-based optimization problems, with implementations on both classical and quantum computers. Whilst promising better scalability with the problem size, quantum GNNs are prone to untrainability; in particular, the barren plateau problem. In this work, we provide a gradient analysis for a specific type of quantum GNN and indicate that it is efficiently trainable for a range of tasks. Our results are validated by simulated experiments on two benchmarks. Furthermore, we comment on the regimes of advantage of the model as well as conditions for its successful dequantization.

1 Introduction

Graph neural networks (GNNs) process structured, relational data efficiently, by analyzing the relations (edges) between different features (node labels) locally. When implemented, however, classical GNN models encounter scalability issues, such as computational blow-up [1], over-smoothing [2], or over-squashing [3].

For this reason, quantum architectures promise a significant computational advantage, by reshaping graph-based tasks onto Hilbert spaces [4–6]. Variational Quantum Algorithms (VQAs) are the leading candidates for realizing this advantage on near-term hardware. The major drawback is that they are usually exposed to the *barren plateau problem* [7–10] which hinders their trainability, making optimization exponentially slow. It is therefore crucial to check if (and when) a quantum model avoids this issue.

Several QGNN architectures have been devised, each of them bringing different insights towards the graph scalability problem. Hybrid models (egoQGNN [11], SQM-GNN [4]) employ classical graph decompositions into small subgraphs. Others (Q-AGNN [6]) implement the quantum circuits at a node-level, then classically performing the data hopping. For more geometry-oriented problems, the QGNN by Verdon et al. [12] tackle data compression to optimize graph-isomorphism tasks. Our work extends the quantum GNNs offered by Liao et al. in [13], which amplitude-encode the graph structure and features onto one global circuit.

In this project, we compute the cost gradient variance of the Quantum Graph Convolutional Networks (QGCNs) presented in [13]. Namely, we find approximate bounds and the expected asymptotic trend of the variance, and validate the results with experiments on two benchmarks. For the Simple Graph Convolution, we provide regimes of advantage for the quantum model as well as conditions for low-rank simulability.

2 Preliminaries: The QGNN Architecture

The QGCNs defined in [13] is split in two categories: Simple and Linear Graph Convolutions (SGCs and LGCs respectively). The problem is set as a node classification of a graph and for this, the circuit consists of two main registers: the node position (address) register, $Reg(i)$ and the data register $Reg(k)$. The former has each standard basis element associated to a node, meaning the size of the register is $n_i = \lceil \log N \rceil$, where N is the number of nodes. For $Reg(k)$, the node feature vectors (of size C) are amplitude encoded, leading to the qubit number $n_k = \lceil \log C \rceil$. A QGCN layer¹ is illustrated in Fig. 1. In the figure, two alternative blocks are passed to the registers, which are then compared using a Modified Hadamard Test using the topmost qubit [14, 15].

*Many thanks to Léo Monbroussou for helpful discussions, as well as observations pointed out in earlier drafts of this work. This work was supported by the Engineering and Physical Sciences Research Council [grant number EP/Y035046/1].

[†]theodor.iosif.24@ucl.ac.uk

¹An activation function is usually encoded as well, but is hardware inefficient and omitting it maintains competitive results.

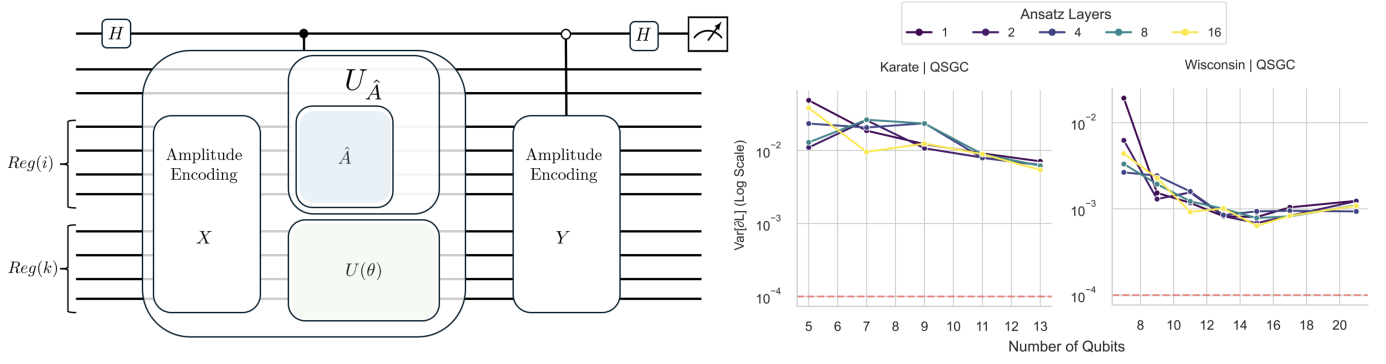


Figure 1: (a): The quantum circuit implementing a layer of the QGCN excluding the activation function (or equivalently, the QSGC). The cost function is evaluated by the Modified Hadamard test. (b): QSGC example for the cost gradient variance $\text{Var}(\partial L/\partial \theta_\mu)$ as a function of the number of qubits for the Karate Club [18] and Wisconsin [19]. A larger figure (Fig. 3) including the QLGC plots is available in Appendix E.

Unpacking the circuit, the first block performs three gates. The X gate amplitude encodes the node feature vectors in $\text{Reg}(k)$ and entangles them to the node positions from $\text{Reg}(i)$. Then, the gate $U_{\hat{A}}$ block encodes the graph’s *normalized adjacency matrix with self-loops*.² Finally, a variational quantum circuit (VQC), or ansatz, is applied to the data register, $U(\theta)$. The other block amplitude-encodes the target classes using gate Y which is functionally identical to X . On its own, the first block computes $|\psi_{out}\rangle \equiv (\hat{A} \otimes U(\theta)) |\psi_X\rangle$, where $|\psi_X\rangle$ is the input data state. The second block gives the target state, $|\psi_Y\rangle$. By performing the Modified Hadamard Test, the repeated measurement effectively gives the following cost function:

$$L_{QGCN} = -\text{Re}(\langle \psi_{out} | \psi_Y \rangle). \quad (1)$$

This widely used cost [16, 17] compares the output state $|\psi_{out}\rangle$ with some target $|\psi_Y\rangle$ via their inner product.

This circuit constitutes a single-layer QSGC. K -layer QSGCs repeat the block encoding above K times, and the LGCs replace it with a polynomial $P(\hat{A})$, effectively superimposing SGCs of different layers.

3 Trainability and Gradient Variance Bounds

Depending on the input embedding, architecture, and optimization process, ML models can be more or less efficiently trained. For gradient descent-based quantum models, the main impediment to efficient trainability is the presence of barren plateaus. This translates to a flattening of the loss landscape as the input size increases. To evaluate this, one turns to the *variance* of the cost function gradient [20], denoted as $\text{Var}(\partial L/\partial \theta_\mu)$. Here, L is the cost and $\partial/\partial \theta_\mu$ is the gradient component over parameter θ_μ .

Assumptions. There are several degrees of freedom in the trainability calculation. First is the graph structure, which we restrict to k -regular graphs, a choice that gives us a sensible quantitative analysis for typical GNN problems [21].³ Second, the choice of input and target (“i,t”) states $|\psi_X\rangle$ and $|\psi_Y\rangle$ can lead to a range of different variances. Here, we approximate their distribution as a spread $\Delta_{i,t}[\text{Var}(\partial_\mu L)]$ centered in the expected variance $\mathbb{E}_{i,t}[\text{Var}(\partial_\mu L)]$. This approximation is reasonable for discrete, classical data, as opposed to (Haar-)random quantum data which concentrates the loss around zero [25–27].

Results. The parameters of interest for our asymptotic behavior are the graph size N (number of nodes), the (average) node degree k , and the feature vector size, C . As such, we derive the following bounds:

$$\Omega(N^{-1}C^{-1}) \leq \mathbb{E}_{i,t}[\text{Var}(\partial_\mu L_{QGCN})] \leq O(C^{-1}), \quad (2)$$

where the upper bound holds for all choices of $U(\theta)$ and is attained by the Instantaneous Quantum Polynomial⁴ (IQP) ansatz [28, 29]. The lower bound is artificially increased by employing a trick that filters

²The matrix element $\hat{A}$ is defined in Eq. 6 in Appendix A. Hereafter, we may often call $\hat{A}$ simply the *adjacency matrix*.

³However, some types of problems deviate significantly from the regular graph approximation [22–24]. k is the node degree.

⁴This circuit uses commuting gates and thus is known to be trainable at the expense of expressivity.

out less trainable ansätze (shown in Appendix C.7.2). Whilst these bounds only tell us how well or how poorly different VQCs can perform, the *curse of dimensionality* reveals to us (in Appendix C.8) that the most likely asymptotic trend of arbitrary ansätze is

$$\mathbb{E}_{i,t}[\text{Var}(\partial_\mu L_{QSGC})]_{\text{expected}} \sim \Theta(C^{-2}) + \Theta(N^{-1}C^{-1}). \quad (3)$$

This does reach the flattening with N , albeit at a significantly worse dependence on C . We thus conclude that if $C \ll N$, the model is trainable, and validate this result with two QSGC experiments (Fig. 1). The gradient analysis for single-layer QSGCs is done in Appendix C. In Appendix D we generalize the results for graphs with weighted edges and use this to extend the results to any K -layer QSGC and QLGC model.

4 Note on Complexity and Dequantization

The QSGC complexity results in [13] suggest a trade-off between the algorithm qubit number and circuit depth. Can we exploit this for quantum advantage? The answer is mixed. First, the classical compilation space-time complexity for the amplitude and block encoding schemes is comparable to the Classical SGC complexity, thus losing all advantage by default (the *input problem* [30, 31]). Unfortunately, as of now the only way to bypass this is to assume an efficient oracle or QRAM. Second, we emphasize that efficient trainability translates to small query complexity — in our case, an $O(C^2)$ overhead.

Trade-off and advantage. For the QSGC (and any other QGCN for that matter), there is freedom in choosing the number of ancillary qubits. More qubits can perform the amplitude and block encoding with a significantly smaller depth, and vice-versa. This gives a trade-off between the overall space and time complexities, and effectively splits the advantage in three regimes: **(1)** By choosing $O(\text{polylog}(N))$ ancillary qubits, the QSGC gains an exponential space reduction and a polylogarithmic speed-up w.r.t. its classical counterpart. **(2)** Choosing a sublinear number of ancillas $O(N^\alpha)$ with $\alpha \in (0, 1)$ leads to polynomial advantage in both space and time complexities. **(3)** Lastly, by choosing $O(N/\text{polylog}(N))$ ancillas, we trade off the advantage in the first case — an exponential speed-up with a polylogarithmic space reduction.

Low-rank simulability. To even out the oracle assumption, Ewin Tang’s breakthrough algorithm [32, 33] samples values from r -rank approximations of $M \times N$ matrices in $O(\text{polylog}(MN)\text{poly}(r, 1/\varepsilon))$ with precision ε . This method is a baseline for dequantization of algorithms claiming quantum advantage. As derived in Appendix F.4, we apply it to find simulability conditions for the QSGC. If the rank of $\hat{A}$ is $r = O(N^{0.09})$, the model can be fully dequantized and if $r = O(\text{polylog}(N))$, the dequantization algorithm gains a further exponential advantage over the QSGC itself.

5 Discussion and Outlook

We studied the trainability and classical simulability for a type of amplitude-encoded QGCNs. As such, we established that most arbitrary VQCs can be efficiently trained on tasks where the classical impediment is graph size scalability. Furthermore, under a QRAM assumption, the base model showcases a tunable space-time advantage that can be exponential in either space or time, or polynomial in both. Lastly, we found that this model can be dequantized for low-rank adjacency matrices $\hat{A}$.

Further work could follow a number of tracks. For instance, the assumptions on the trainability calculation could be loosened to find a more accurate gradient scaling. On the simulability side, our analysis only considers the classical/quantum SGC comparison and one dequantization scheme. This could be extended to LGCs as well as more sophisticated models delivering the same output in order to find more simulable regimes. The trainability and simulability calculation could also be generalized for the Attention and Message-Passing QGNN variants described in [13]. To this end, we believe that this work sets an example for finding how different types of graph and feature encodings impact the trainability and classical simulability of a quantum GNN. Therefore, in the quickly expanding zoo of QGNNs, it would be worth performing this analysis on other models to further improve their quality.

The results in this project are included in a thorough implementation and benchmarking of QGCNs.

References

- [1] Rui Xue et al. *Haste Makes Waste: A Simple Approach for Scaling Graph Neural Networks*. 2024. arXiv: [2410.05416](https://arxiv.org/abs/2410.05416) [cs.LG]. URL: <https://arxiv.org/abs/2410.05416>.
- [2] Xiang Li et al. *ScaleGNN: Towards Scalable Graph Neural Networks via Adaptive High-order Neighboring Feature Fusion*. 2026. arXiv: [2504.15920](https://arxiv.org/abs/2504.15920) [cs.LG]. URL: <https://arxiv.org/abs/2504.15920>.
- [3] Zinuo You, Jin Zheng, and John Cartlidge. “How Wide and How Deep? Mitigating Over-squashing of GNNs via Channel Capacity Constrained Estimation”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 40.33 (Mar. 2026), pp. 27890–27898. ISSN: 2159-5399. DOI: [10.1609/aaai.v40i33.40012](https://doi.org/10.1609/aaai.v40i33.40012). URL: <http://dx.doi.org/10.1609/aaai.v40i33.40012>.
- [4] Le Tung Giang et al. *Scalable Quantum Message Passing Graph Neural Networks for Next-Generation Wireless Communications: Architectures, Use Cases, and Future Directions*. 2026. arXiv: [2601.18198](https://arxiv.org/abs/2601.18198) [cs.IT]. URL: <https://arxiv.org/abs/2601.18198>.
- [5] Muhammad Kashif, Alberto Marchisio, and Muhammad Shafique. *Computational Advantage in Hybrid Quantum Neural Networks: Myth or Reality?* 2025. arXiv: [2412.04991](https://arxiv.org/abs/2412.04991) [quant-ph]. URL: <https://arxiv.org/abs/2412.04991>.
- [6] Devashish Chaudhary, Sutharshan Rajasegarar, and Shiva Raj Pokhrel. *Q-AGNN: Quantum-Enhanced Attentive Graph Neural Network for Intrusion Detection*. 2026. arXiv: [2603.22365](https://arxiv.org/abs/2603.22365) [cs.CR]. URL: <https://arxiv.org/abs/2603.22365>.
- [7] Martín Larocca et al. “Barren plateaus in variational quantum computing”. In: *Nature Reviews Physics* 7.4 (Mar. 2025), pp. 174–189. ISSN: 2522-5820. DOI: [10.1038/s42254-025-00813-9](https://doi.org/10.1038/s42254-025-00813-9). URL: <http://dx.doi.org/10.1038/s42254-025-00813-9>.
- [8] Muhammad Kashif et al. “Alleviating Barren Plateaus in Parameterized Quantum Machine Learning Circuits: Investigating Advanced Parameter Initialization Strategies”. In: *2024 Design, Automation, and Test in Europe Conference and Exhibition (DATE)*. IEEE, Mar. 2024, pp. 1–6. DOI: [10.23919/date58400.2024.10546644](https://doi.org/10.23919/date58400.2024.10546644). URL: <http://dx.doi.org/10.23919/DATE58400.2024.10546644>.
- [9] Jack Cunningham and Jun Zhuang. *Investigating and Mitigating Barren Plateaus in Variational Quantum Circuits: A Survey*. 2025. arXiv: [2407.17706](https://arxiv.org/abs/2407.17706) [quant-ph]. URL: <https://arxiv.org/abs/2407.17706>.
- [10] Michelle Gelman. *A Survey of Methods for Mitigating Barren Plateaus for Parameterized Quantum Circuits*. 2024. arXiv: [2406.14285](https://arxiv.org/abs/2406.14285) [quant-ph]. URL: <https://arxiv.org/abs/2406.14285>.
- [11] Xing Ai et al. *Towards Quantum Graph Neural Networks: An Ego-Graph Learning Approach*. 2024. arXiv: [2201.05158](https://arxiv.org/abs/2201.05158) [quant-ph]. URL: <https://arxiv.org/abs/2201.05158>.
- [12] Guillaume Verdon et al. “Quantum Graph Neural Networks”. In: arXiv:1909.12264 (2019). DOI: [10.48550/arXiv.1909.12264](https://doi.org/10.48550/arXiv.1909.12264). URL: <http://arxiv.org/abs/1909.12264>.
- [13] Yidong Liao, Xiao-Ming Zhang, and Chris Ferrie. *Graph Neural Networks on Quantum Computers*. arXiv:2405.17060 [quant-ph]. May 2024. DOI: [10.48550/arXiv.2405.17060](https://doi.org/10.48550/arXiv.2405.17060). URL: <http://arxiv.org/abs/2405.17060> (visited on 06/04/2024).
- [14] J. Knörzer, D. Malz, and J. I. Cirac. “Cross-platform verification in quantum networks”. In: *Physical Review A* 107.6 (June 30, 2023). Publisher: American Physical Society, p. 062424. DOI: [10.1103/PhysRevA.107.062424](https://doi.org/10.1103/PhysRevA.107.062424). URL: <https://link.aps.org/doi/10.1103/PhysRevA.107.062424> (visited on 04/11/2026).
- [15] Alessandro Luongo. *Quantum algorithms for data analysis*. 2022. URL: <https://quantumalgorithms.org/quantumalgorithms.pdf> (visited on 04/11/2026).
- [16] Carlos Bravo-Prieto et al. “Variational Quantum Linear Solver”. In: *Quantum* 7 (Nov. 2023), p. 1188. ISSN: 2521-327X. DOI: [10.22331/q-2023-11-22-1188](https://doi.org/10.22331/q-2023-11-22-1188). URL: <http://dx.doi.org/10.22331/q-2023-11-22-1188>.
- [17] Sumeet Khatri et al. “Quantum-assisted quantum compiling”. In: *Quantum* 3 (May 2019), p. 140. ISSN: 2521-327X. DOI: [10.22331/q-2019-05-13-140](https://doi.org/10.22331/q-2019-05-13-140). URL: <http://dx.doi.org/10.22331/q-2019-05-13-140>.

- [18] Wayne W. Zachary. “An Information Flow Model for Conflict and Fission in Small Groups”. In: *Journal of Anthropological Research* 33.4 (Dec. 1977). Publisher: The University of Chicago Press, pp. 452–473. ISSN: 0091-7710. DOI: [10.1086/jar.33.4.3629752](https://doi.org/10.1086/jar.33.4.3629752). URL: <https://www.journals.uchicago.edu/doi/abs/10.1086/jar.33.4.3629752> (visited on 03/23/2026).
- [19] Hongbin Pei et al. *Geom-GCN: Geometric Graph Convolutional Networks*. Feb. 14, 2020. DOI: [10.48550/arXiv.2002.05287](https://doi.org/10.48550/arXiv.2002.05287). arXiv: [2002.05287](https://arxiv.org/abs/2002.05287)[cs]. URL: <http://arxiv.org/abs/2002.05287> (visited on 04/17/2026).
- [20] Jarrod R. McClean et al. “Barren plateaus in quantum neural network training landscapes”. en. In: *Nature Communications* 9.1 (Nov. 2018), p. 4812. ISSN: 2041-1723. DOI: [10.1038/s41467-018-07090-4](https://doi.org/10.1038/s41467-018-07090-4).
- [21] Michael M. Bronstein et al. “Geometric Deep Learning: Going beyond Euclidean data”. In: *IEEE Signal Process. Mag.* 34 (2016), pp. 18–42. URL: <https://api.semanticscholar.org/CorpusID:15195762>.
- [22] Angélica Mata. “Complex Networks: a Mini-review”. In: *Brazilian Journal of Physics* 50 (July 2020). DOI: [10.1007/s13538-020-00772-9](https://doi.org/10.1007/s13538-020-00772-9).
- [23] Tiago Simas, Rion Brattig Correia, and Luis M Rocha. “The distance backbone of complex networks”. In: *Journal of Complex Networks* 9.6 (Dec. 2021), cnab021. ISSN: 2051-1329. DOI: [10.1093/comnet/cnab021](https://doi.org/10.1093/comnet/cnab021).
- [24] Florian Kühnlenz and Pedro H. J. Nardelli. “Dynamics of Complex Systems Built as Coupled Physical, Communication and Decision Layers”. en. In: *PLOS ONE* 11.1 (Jan. 2016), e0145135. ISSN: 1932-6203. DOI: [10.1371/journal.pone.0145135](https://doi.org/10.1371/journal.pone.0145135).
- [25] Supanut Thanasilp et al. “Subtleties in the trainability of quantum machine learning models”. In: *Quantum Machine Intelligence* 5.1 (May 2023). ISSN: 2524-4914. DOI: [10.1007/s42484-023-00103-6](https://doi.org/10.1007/s42484-023-00103-6). URL: <http://dx.doi.org/10.1007/s42484-023-00103-6>.
- [26] M. Cerezo et al. “Cost function dependent barren plateaus in shallow parametrized quantum circuits”. In: *Nature Communications* 12.1 (Mar. 2021). ISSN: 2041-1723. DOI: [10.1038/s41467-021-21728-w](https://doi.org/10.1038/s41467-021-21728-w). URL: <http://dx.doi.org/10.1038/s41467-021-21728-w>.
- [27] Zoë Holmes et al. “Connecting Ansatz Expressibility to Gradient Magnitudes and Barren Plateaus”. en. In: *PRX Quantum* 3.1 (Jan. 2022), p. 010313. ISSN: 2691-3399. DOI: [10.1103/PRXQuantum.3.010313](https://doi.org/10.1103/PRXQuantum.3.010313).
- [28] Dan Shepherd and Michael J. Bremner. “Instantaneous Quantum Computation”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 465.2105 (May 2009), pp. 1413–1439. ISSN: 1364-5021, 1471-2946. DOI: [10.1098/rspa.2008.0443](https://doi.org/10.1098/rspa.2008.0443).
- [29] Michael J. Bremner, Bin Cheng, and Zhengfeng Ji. “Instantaneous Quantum Polynomial-Time Sampling and Verifiable Quantum Advantage: Stabilizer Scheme and Classical Security”. In: *PRX Quantum* 6.2 (Apr. 2025). arXiv:2308.07152 [quant-ph], p. 020315. ISSN: 2691-3399. DOI: [10.1103/PRXQuantum.6.020315](https://doi.org/10.1103/PRXQuantum.6.020315).
- [30] Scott Aaronson. “Read the fine print”. en. In: *Nature Physics* 11.4 (Apr. 2015), pp. 291–293. ISSN: 1745-2481. DOI: [10.1038/nphys3272](https://doi.org/10.1038/nphys3272).
- [31] Jacob Biamonte et al. “Quantum machine learning”. en. In: *Nature* 549.7671 (2017), pp. 195–202. ISSN: 1476-4687. DOI: [10.1038/nature23474](https://doi.org/10.1038/nature23474).
- [32] Ewin Tang. “A quantum-inspired classical algorithm for recommendation systems”. en. In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. arXiv:1807.04271 [cs.IR]. 2019, pp. 217–228. DOI: [10.1145/3313276.3316310](https://doi.org/10.1145/3313276.3316310). URL: <http://arxiv.org/abs/1807.04271>.
- [33] Nai-Hui Chia et al. “Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning”. In: *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*. arXiv:1910.06151 [cs.DS]. 2020, pp. 387–400. DOI: [10.1145/3357713.3384314](https://doi.org/10.1145/3357713.3384314). URL: <http://arxiv.org/abs/1910.06151>.
- [34] Chris Godsil and Gordon Royle. *Algebraic Graph Theory*. en. 2001st ed. Graduate Texts in Mathematics. New York, NY: Springer, Apr. 2001.

- [35] Jean Gallier. *Spectral Theory of Unsigned and Signed Graphs. Applications to Graph Clustering: a Survey*. 2016. arXiv: [1601.04692](https://arxiv.org/abs/1601.04692) [cs.LG]. URL: <https://arxiv.org/abs/1601.04692>.
- [36] Felix Wu et al. *Simplifying Graph Convolutional Networks*. June 20, 2019. DOI: [10.48550/arXiv.1902.07153](https://doi.org/10.48550/arXiv.1902.07153). arXiv: [1902.07153](https://arxiv.org/abs/1902.07153)[cs]. URL: <http://arxiv.org/abs/1902.07153> (visited on 04/11/2026).
- [37] Seiji Maekawa et al. “Beyond Real-world Benchmark Datasets: An Empirical Study of Node Classification with GNNs”. In: *Advances in Neural Information Processing Systems* 35 (Dec. 6, 2022), pp. 5562–5574. URL: https://proceedings.neurips.cc/paper_files/paper/2022/hash/24d6d158531508115e628188e2697f76-Abstract-Datasets_and_Benchmarks.html (visited on 04/11/2026).
- [38] Luca Pasa et al. *Simple Graph Convolutional Networks*. June 10, 2021. DOI: [10.48550/arXiv.2106.05809](https://doi.org/10.48550/arXiv.2106.05809). arXiv: [2106.05809](https://arxiv.org/abs/2106.05809)[cs]. URL: <http://arxiv.org/abs/2106.05809> (visited on 04/11/2026).
- [39] Naixu Guo, Kosuke Mitarai, and Keisuke Fujii. “Nonlinear transformation of complex amplitudes via quantum singular value transformation”. In: *Physical Review Research* 6.4 (Dec. 2, 2024). Publisher: American Physical Society, p. 043227. DOI: [10.1103/PhysRevResearch.6.043227](https://doi.org/10.1103/PhysRevResearch.6.043227). URL: <https://link.aps.org/doi/10.1103/PhysRevResearch.6.043227> (visited on 04/11/2026).
- [40] Arthur G. Rattew and Patrick Rebentrost. *Non-Linear Transformations of Quantum Amplitudes: Exponential Improvement, Generalization, and Applications*. Sept. 18, 2023. DOI: [10.48550/arXiv.2309.09839](https://doi.org/10.48550/arXiv.2309.09839). arXiv: [2309.09839](https://arxiv.org/abs/2309.09839)[quant-ph]. URL: <http://arxiv.org/abs/2309.09839> (visited on 04/11/2026).
- [41] Luca Pasa et al. “Empowering Simple Graph Convolutional Networks”. In: *IEEE Transactions on Neural Networks and Learning Systems* 35.4 (Apr. 2024), pp. 4385–4399. ISSN: 2162-2388. DOI: [10.1109/TNNLS.2022.3232291](https://doi.org/10.1109/TNNLS.2022.3232291). URL: <https://ieeexplore.ieee.org/abstract/document/10005046> (visited on 04/11/2026).
- [42] Xiao-Ming Zhang and Xiao Yuan. “Circuit complexity of quantum access models for encoding classical data”. In: *npj Quantum Information* 10.1 (Apr. 23, 2024). Publisher: Nature Publishing Group, p. 42. ISSN: 2056-6387. DOI: [10.1038/s41534-024-00835-8](https://doi.org/10.1038/s41534-024-00835-8). URL: <https://www.nature.com/articles/s41534-024-00835-8> (visited on 04/11/2026).
- [43] Alok Shukla and Prakash Vedula. “An efficient quantum algorithm for preparation of uniform quantum superposition states”. In: *Quantum Information Processing* 23.2 (Jan. 29, 2024), p. 38. ISSN: 1573-1332. DOI: [10.1007/s11128-024-04258-4](https://doi.org/10.1007/s11128-024-04258-4). URL: <https://doi.org/10.1007/s11128-024-04258-4> (visited on 09/12/2025).
- [44] Naixu Guo, Kosuke Mitarai, and Keisuke Fujii. *Nonlinear transformation of complex amplitudes via quantum singular value transformation*. en. 2021. URL: <https://arxiv.org/abs/2107.10764v2>.
- [45] M. Cerezo et al. “Does provable absence of barren plateaus imply classical simulability?” en. In: *Nature Communications* 16.1 (Aug. 2025), p. 7907. ISSN: 2041-1723. DOI: [10.1038/s41467-025-63099-6](https://doi.org/10.1038/s41467-025-63099-6).
- [46] Ville Bergholm et al. *PennyLane: Automatic differentiation of hybrid quantum-classical computations*. July 29, 2022. DOI: [10.48550/arXiv.1811.04968](https://doi.org/10.48550/arXiv.1811.04968). arXiv: [1811.04968](https://arxiv.org/abs/1811.04968)[quant-ph]. URL: <http://arxiv.org/abs/1811.04968> (visited on 01/10/2025).
- [47] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. 2018. URL: <http://github.com/jax-ml/jax>.
- [48] Maria Schuld et al. “Circuit-centric quantum classifiers”. In: *Physical Review A* 101.3 (Mar. 6, 2020), p. 032308. ISSN: 2469-9926, 2469-9934. DOI: [10.1103/PhysRevA.101.032308](https://doi.org/10.1103/PhysRevA.101.032308). arXiv: [1804.00633](https://arxiv.org/abs/1804.00633)[quant-ph]. URL: <http://arxiv.org/abs/1804.00633> (visited on 12/12/2023).
- [49] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. Jan. 29, 2017. DOI: [10.48550/arXiv.1412.6980](https://doi.org/10.48550/arXiv.1412.6980). arXiv: [1412.6980](https://arxiv.org/abs/1412.6980)[cs]. URL: <http://arxiv.org/abs/1412.6980> (visited on 12/12/2023).
- [50] K. Mitarai et al. “Quantum circuit learning”. en. In: *Physical Review A* 98.3 (2018), p. 032309. ISSN: 2469-9926, 2469-9934. DOI: [10.1103/PhysRevA.98.032309](https://doi.org/10.1103/PhysRevA.98.032309).

- [51] Leonardo Banchi, Dominic Branford, and Chetan Waghela. “Overshifted Parameter-Shift Rules: Optimizing Complex Quantum Systems with Few Measurements”. en. In: (2025). DOI: [10.48550/ARXIV.2510.05289](https://doi.org/10.48550/ARXIV.2510.05289). URL: <https://arxiv.org/abs/2510.05289>.
- [52] Meng Li and Ron Goldman. “Limits of sums for binomial and Eulerian numbers and their associated distributions”. In: *Discrete Mathematics* 343.7 (2020), p. 111870. ISSN: 0012-365X. DOI: <https://doi.org/10.1016/j.disc.2020.111870>.
- [53] G. Casella and R.L. Berger. *Statistical Inference*. Duxbury advanced series in statistics and decision sciences. Thomson Learning, 2002. ISBN: 978-0-534-24312-8. URL: https://books.google.co.uk/books?id=0x_vAAAAAAAJ.
- [54] Richard Ernest Bellman. *Dynamic Programming*. Princeton, NJ: Princeton University Press, 1957.
- [55] Lei Chen. “Curse of Dimensionality”. In: *Encyclopedia of Database Systems*. Ed. by LING LIU and M. TAMER ÖZSU. Boston, MA: Springer US, 2009, pp. 545–546. ISBN: 978-0-387-39940-9. DOI: [10.1007/978-0-387-39940-9_133](https://doi.org/10.1007/978-0-387-39940-9_133). URL: https://doi.org/10.1007/978-0-387-39940-9_133.
- [56] Naomi Altman and Martin Krzywinski. “The curse(s) of dimensionality”. en. In: *Nat. Methods* 15.6 (June 2018), pp. 399–400.
- [57] Zheyuan Hu et al. “Tackling the curse of dimensionality with physics-informed neural networks”. In: *Neural Networks* 176 (Aug. 2024), p. 106369. ISSN: 0893-6080. DOI: [10.1016/j.neunet.2024.106369](https://doi.org/10.1016/j.neunet.2024.106369). URL: <http://dx.doi.org/10.1016/j.neunet.2024.106369>.

APPENDIX

In this Appendix we provide the full trainability and classical simulability derivations. Specifically, in Appendix A we introduce the graph notation and some relevant results. In Appendix B we give a more thorough description of the QGCNs presented in [13]. Appendix C is concerned with the full trainability derivations for the single-layer QSGC, followed by a generalization to all other QSGC and QLGC models in Appendix D. The following section illustrates the trainability experiments in full. In Appendix F, we provide the advantage and simulability analysis for the QSGC, leaving the last section for helper lemmas.

A Graph preliminaries

A.1 Unweighted graphs

A graph is a collection of nodes $\mathcal{N}$ and edges $\mathcal{E}$ between the nodes. It usually describes structured information where different parts of the data are linked via some relations. In our work, we define addresses of the nodes and encode them into the register $Reg(i)$, thus called the address register. In order to encode the edges, we employ the *normalized adjacency matrix with self-loops*. To unwrap this notion, we define the adjacency matrix A as a symmetric matrix such that

$$A_{ij} = \begin{cases} 1 & \text{if nodes } i, j \text{ are connected,} \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

By definition, $A_{ii} = 0$. In a GNN, it is important to reinforce the information of a node onto itself, so we introduce self-loops in the graph (literally meaning that each node has an edge linking to itself). The self loops count as one extra edge for each node. We define this as $\tilde{A} \equiv A + I_N$, where N is the total number of nodes. Finally, for the purpose of encoding this matrix into a unitary gate, the matrix itself needs to have a sub-unitary spectral norm, meaning $\|A\|_2 \leq 1$. For this reason, $\tilde{A}$ must also be normalized. In order not to bias more connected nodes, we use the following normalization:

$$\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}, \quad (5)$$

where $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ is the degree matrix. This normalizes each edge by the degrees of the connected nodes as

$$\hat{A}_{ij} = \frac{A_{ij}}{\sqrt{\deg(n_i) \deg(n_j)}}. \quad (6)$$

Here, the degree $\deg(n_i) = \tilde{D}_{ii}$ of node n_i is the number of edges connected to it.

A.2 Weighted graphs

For a graph with *weighted edges*, each edge has an associated weight $w_{ij} > 0$. In this case the adjacency matrix is defined as

$$A_{ij} = \begin{cases} w_{ij} & \text{if nodes } i, j \text{ are connected,} \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

By adding the self loops, we now have extra freedom in choosing the weights of the self loops to be $w_{ii} > 0$. Therefore, the new adjacency matrix is also

$$\tilde{A}_{ij} = \begin{cases} A_{ij} & \text{if } i \neq j, \\ w_{ii} & \text{otherwise.} \end{cases} \quad (8)$$

The definitions for adding self-loops and the degrees are the same as above, therefore inheriting the changes from Eq. 7. We note that by node degree and degree matrix in this case we refer to weighted degrees, meaning that, in accordance with the definitions above, we have

$$\tilde{D}_{ii} = \sum_j \tilde{A}_{ij} = \sum_j w_{ij}, \quad (9)$$

with $\deg(n_i) := \tilde{D}_{ii}$. This means that node n_i is not restricted to having $\deg(n_i)$ edges connected to it, but instead it sums up the weights of all its connected edges.

Next, we will prove two important results that will help us generalize our one-layer QSGC results to both general QSGCs and QLGCs. The generalization is done in Appendix D. For further details we refer the reader to [34, 35].

A.2.1 Multiplying normalized adjacency matrices

Here we show that multiplying two commuting normalized adjacency matrices with self-loops produces a new, valid normalized adjacency matrix with self-loops. This is done in Proposition 3, and to prove it, we will employ the following two lemmas:

Lemma 1 *For a graph whose normalized adjacency matrix with self-loops is $\hat{A}$ and the degree matrix is $\tilde{D}$, the vector $\tilde{D}^{1/2}\mathbf{1}$ is an eigenvector of $\hat{A}$, with eigenvalue 1. Here, $\mathbf{1}$ is a vector whose elements are ones.*

Proof. For the usual adjacency matrix with self-loops $\tilde{A}$, we notice that

$$(\tilde{A}\mathbf{1})_i = \sum_j \tilde{A}_{ij} \cdot 1 = \tilde{D}_{ii} \Rightarrow \tilde{A}\mathbf{1} = \tilde{D}\mathbf{1}. \quad (10)$$

Therefore, calculating

$$\hat{A} \cdot \tilde{D}^{1/2}\mathbf{1} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}\tilde{D}^{1/2}\mathbf{1} = \tilde{D}^{-1/2}\tilde{A}\mathbf{1} = \tilde{D}^{1/2}\mathbf{1}, \quad (11)$$

thus $\tilde{D}^{1/2}\mathbf{1}$ is an eigenvector of $\hat{A}$ corresponding to eigenvalue 1. $\square$

Lemma 2 *If the normalized adjacency matrices with self-loops of two graphs commute, then their degree matrices are proportional.*

Proof. For graphs $\mathcal{G}_A$ and $\mathcal{G}_B$ we define the normalized adjacency matrices with self-loops to be $\hat{A}$ and $\hat{B}$, respectively. We also denote the degree matrices as $\tilde{D}_A$ and $\tilde{D}_B$.

If $\hat{A}$ and $\hat{B}$ commute, then they share a mutual basis of eigenvectors. Furthermore, because they are symmetric, the basis is orthogonal.

Using Lemma 1, we know that $\tilde{D}_A^{1/2}\mathbf{1}$ and $\tilde{D}_B^{1/2}\mathbf{1}$ are in this basis. If they do not coincide, they should be orthogonal, which is impossible because their inner product is

$$\mathbf{1}^T (\tilde{D}_B^{1/2})^T \tilde{D}_A^{1/2} \mathbf{1} = \sum_i \sqrt{(\tilde{D}_A)_{ii}(\tilde{D}_B)_{ii}} > 0. \quad (12)$$

Therefore, they must be proportional, and so $(\tilde{D}_A)_{ii} \propto (\tilde{D}_B)_{ii}$, which completes the proof. $\square$

Proposition 3 *Let $\mathcal{G}_A$ and $\mathcal{G}_B$ be two graphs with weighted edges and the same number of nodes, such that their normalized adjacency matrices with self-loops are respectively $\hat{A}$ and $\hat{B}$. Then there exists a graph $\mathcal{G}_C$ whose normalized adjacency matrix with self-loops matches $\hat{C} = \hat{A}\hat{B}$ only if the two matrices $\hat{A}$ and $\hat{B}$ commute.*

Proof. We solve this from an algebraic point of view. The three conditions that $\hat{C}$ needs to meet are (1) symmetry, (2) non-negativity, and (3) the degree relationship given in Eq. 6, based on constructing a valid degree matrix $\tilde{D}_C$ associated to it.

The first condition is solved by the symmetry of each of the matrices $\hat{A}$ and $\hat{B}$, as well as the commutation relation:

$$\hat{C}^T = (\hat{A}\hat{B})^T = \hat{B}^T \hat{A}^T = \hat{B}\hat{A} = \hat{A}\hat{B} = \hat{C}. \quad (13)$$

The second condition means that all of the elements $\hat{C}_{ij} = \sum_k \hat{A}_{ik}\hat{B}_{kj}$ must be positive, which is true because all elements in $\hat{A}$ and $\hat{B}$ are also positive.

Finally, to address the third condition, we start constructing the desired graph $\mathcal{G}_C$. As we know from Lemma 2, the degree matrices of graphs $\mathcal{G}_A$ and $\mathcal{G}_B$ are proportional; let them be related as $\tilde{D}_B = b\tilde{D}_A$, where $\tilde{D}_i$ is the degree matrix of graph $\mathcal{G}_i$ and a is a proportionality constant.

Because $\hat{C} = \hat{A}\hat{B}$ commutes with both, if we want $\mathcal{G}_C$ to have $\hat{C}$ as the normalized adjacency matrix with self-loops, we need to further define the degree matrix $\tilde{D}_C \equiv c\tilde{D}_A$.

Finally, we check the relation in Eq. 6 for $\hat{C}$ and $\tilde{D}_C$:

$$\begin{aligned}\tilde{D}_C^{\frac{1}{2}}\hat{C}\tilde{D}_C^{\frac{1}{2}} &= \tilde{D}_C^{\frac{1}{2}}\hat{A}\hat{B}\tilde{D}_C^{\frac{1}{2}} \\ &= \tilde{D}_C^{\frac{1}{2}}\tilde{D}_A^{-\frac{1}{2}}\tilde{A}\tilde{D}_A^{-\frac{1}{2}}\tilde{D}_B^{-\frac{1}{2}}\hat{B}\tilde{D}_B^{-\frac{1}{2}}\tilde{D}_C^{\frac{1}{2}} \\ &= \frac{c}{b}\tilde{D}_A^{\frac{1}{2}}\tilde{D}_A^{-\frac{1}{2}}\tilde{A}\tilde{D}_A^{-\frac{1}{2}}\tilde{D}_A^{-\frac{1}{2}}\hat{B}\tilde{D}_A^{-\frac{1}{2}}\tilde{D}_A^{\frac{1}{2}} \\ &= \frac{c}{b}\tilde{A}\tilde{D}_A\hat{B} \equiv \tilde{C}.\end{aligned}\tag{14}$$

After simplifications, we establish that the weights of the desired graph should be

$$\tilde{C}_{ij} = \frac{c}{b} \sum_k \frac{\tilde{A}_{ik}\tilde{B}_{kj}}{(\tilde{D}_B)_{kk}}.\tag{15}$$

Assuming b is given, c is still a choice in our definition. Therefore, not one, but a whole family of graphs $\{\mathcal{G}_C^c\}_{c>0}$ is defined such that $\hat{A}\hat{B}$ is the (mutual) normalized adjacency matrix with self-loops.

To this end, we need to check that the relation $(\tilde{D}_C)_{ii} = \sum_j \tilde{C}_{ij}$ holds:

$$\sum_j \tilde{C}_{ij} = \frac{c}{b} \sum_k \frac{\tilde{A}_{ik} \sum_j \tilde{B}_{kj}}{(\tilde{D}_B)_{kk}} = c \sum_k \frac{\tilde{A}_{ik}}{(\tilde{D}_A)_{kk}} = c(\tilde{D}_A)_{ii} \underbrace{\sum_k \frac{\tilde{A}_{ik}}{(\tilde{D}_A)_{ii}(\tilde{D}_A)_{kk}}}_{\sum_k \hat{A}_{ik}=1} = (\tilde{D}_C)_{ii}.\tag{16}$$

Note that this check is equivalent to checking that Lemma 1 holds for $\mathcal{G}_C$. $\square$

For k -regular graphs we can also give the following corollary:

Corollary 4 *Following Proposition 3, if graphs $\mathcal{G}_A$ and $\mathcal{G}_B$ are weighted k -regular, then $\mathcal{G}_C$ can be chosen to be k -regular.*

Proof. For a regular graph, all nodes share the same degree value $k+1$ (including the self-loops). As such, the degree matrices are $\tilde{D}_A = \tilde{D}_B = k\mathbf{1}$ (so $b = 1$). As seen in the previous proof, $\tilde{D}_C = c\tilde{D}_A$, which makes the new degree matrix be

$$\tilde{D}_C = c(k+1)\mathbf{1}.\tag{17}$$

Therefore, to make the new graph k -regular we set $c = 1$. $\square$

A.3 Convex sums of normalized adjacency matrices

In this section we further extend the properties of commuting normalized adjacency matrices with self-loops by also showing that taking the convex sum of two such matrices leads to a new, valid normalized adjacency matrix with self-loops.

Proposition 5 *For two graphs $\mathcal{G}_A$ and $\mathcal{G}_B$ with commuting normalized adjacency matrices with self-loops $\hat{A}$ and $\hat{B}$, there exists a graph $\mathcal{G}_C$ such that its normalized adjacency matrix with self-loops is a convex sum $\hat{C} = \alpha\hat{A} + \beta\hat{B}$ for some non-negative numbers α and β such that $\alpha + \beta = 1$.*

Proof. As done in the previous Proposition, we confirm that a linear combination of two symmetric, non-negative matrices gives a symmetric, non-negative matrix as long as the coefficient α and β are non-negative.

By employing Lemma 2, because $\hat{A}$ and $\hat{B}$, the graphs' degree matrices are proportional, $\tilde{D}_B = b\tilde{D}_A$. If we want to construct a graph $\mathcal{G}_C$ whose normalized adjacency matrix with self-loops is $\hat{C}$, then this would also commute with $\hat{A}$ and $\hat{B}$, and so a necessary condition is that the degree matrix of $\mathcal{G}_C$ is proportional to that of $\mathcal{G}_A$, as $\tilde{D}_C = c\tilde{D}_A$ for some positive number c which we set.

Then, to complete the construction of $\mathcal{G}_C$, we calculate the weights given by the unnormalized adjacency matrix $\tilde{C}$ as follows:

$$\begin{aligned}\tilde{C} &= \tilde{D}_C^{\frac{1}{2}} \hat{C} \tilde{D}_C^{\frac{1}{2}} \\ &= c \tilde{D}_A^{\frac{1}{2}} (\alpha \hat{A} + \beta \hat{B}) \tilde{D}_A^{\frac{1}{2}} \\ &= c \tilde{D}_A^{\frac{1}{2}} \left(\alpha \tilde{D}_A^{-\frac{1}{2}} \tilde{A} \tilde{D}_A^{-\frac{1}{2}} + \beta \tilde{D}_B^{-\frac{1}{2}} \tilde{B} \tilde{D}_B^{-\frac{1}{2}} \right) \tilde{D}_A^{\frac{1}{2}} \\ &= c \left(\alpha \tilde{A} + \frac{\beta}{b} \tilde{B} \right).\end{aligned}\tag{18}$$

As done above, a last check that $(\tilde{D}_C)_{ii} = \sum_j \tilde{C}_{ij}$:

$$\sum_j \tilde{C}_{ij} = c \left(\alpha \sum_j \tilde{A}_{ij} + \frac{\beta}{b} \sum_j \tilde{B}_{ij} \right) = c(\alpha + \beta) (\tilde{D}_A)_{ii} = (\alpha + \beta)(\tilde{D}_C)_{ii} = (\tilde{D}_C)_{ii},\tag{19}$$

which is only valid because $\alpha + \beta = 1$. Here, everything except c is given and c can be set to be any positive number. $\square$

This can be extended to a convex sum over n normalized adjacency matrices with self-loops:

Corollary 6 *For graphs $\mathcal{G}_{A_p}$ with normalized adjacency matrices with self-loops $\hat{A}_p$ for p ranging from 1 to n , the convex sum $\hat{C} = \sum_p \alpha_p \hat{A}_p$ is a valid adjacency matrix with self-loops of some graph $\mathcal{G}_C$, for some α_p such that $\sum_p \alpha_p = 1$.*

Proof. By induction, the two-term case is done in the Proposition above. Let us assume it holds for n such graphs:

$$\hat{A} \equiv \sum_{p=1}^n \alpha_p \hat{A}_p, \quad \text{where} \quad \sum_{p=1}^n \alpha_p = 1.\tag{20}$$

In order to add a new graph, $\mathcal{G}_{A_{n+1}}$ with $\hat{A}_{n+1}$, we apply Proposition 5 again: For some α and β , the following is a valid normalized adjacency matrix with self-loops:

$$\hat{C} = \alpha \sum_{p=1}^n \alpha_p \hat{A}_p + \beta \hat{A}_{n+1}, \quad \text{where} \quad \alpha \sum_{p=1}^n \alpha_p + \beta = 1.\tag{21}$$

By relabeling $\alpha \alpha_p \equiv \alpha'_p$ and $\beta \equiv \alpha'_{n+1}$, the condition is met for the $n+1$ graphs as well. $\square$

Finally, a corollary on k -regularity:

Corollary 7 *Following Proposition 3, if graphs $\mathcal{G}_A$ and $\mathcal{G}_B$ are weighted k -regular, then $\mathcal{G}_C$ can be chosen to be k -regular.*

Proof. Similar to the proof of Corollary 4. Because $\tilde{D}_C = c\tilde{D}_A = c(k+1)\mathbf{1}$, then the free parameter c can be set to 1. $\square$

B QGCN model and circuit details

Here we dive into the details of the QGCN architecture. We first describe the QSGC in detail as it stands at the core of the other models.

B.1 Quantum Simplified Graph Convolutions

In graph convolutional networks (GCNs), with each layer, the information is scrambled locally. This is done by employing a convolution that updates the data at node n_i of the graph by linearly mixing it with the data from its adjacent nodes. For simplicity, this convolution can be thought of as applying the normalized adjacency matrix $\hat{A}$ to the input data X . Afterwards, a trainable weight matrix $W^{(0)}$ is applied to the same data. Finally, an activation function σ is used to enhance the expressivity. This can be illustrated as

$$H^{(1)} = \sigma \left(\hat{A} X W^{(1)} \right). \quad (22)$$

By applying multiple layers, a telescopic expression is reached:

$$H^{(l)} = \sigma \left(\hat{A} \sigma \left(\hat{A} \sigma \left(\dots \hat{A} X W^{(1)} \right) \dots W^{(l-1)} \right) W^{(l)} \right). \quad (23)$$

Nonlinearities. In most classical learning models, expressivity is based on the presence of nonlinearities. However, in the case of GCNs, removing them leads to a well-known GCN variant called a *Simplified Graph Convolution* (SGC), which shows competitive results despite its low model complexity [36–38].

By dropping the nonlinearities, the application of l layers takes the form

$$H_{SGC}^{(l)} = \hat{A}^l X W, \quad (24)$$

where $W = \prod_l W^{(l)}$ is the combined weight matrix, which acts as a single layer weight matrix.

This model is of particular interest to us, because even though nonlinearities can be implemented on quantum circuits [39, 40], they bring a large overhead to the circuit depth, making it inefficient.

To implement a one-layer quantum SGC (or QSGC), we construct the circuit in Fig. 2. Here, the main registers are the address register $Reg(i)$ and the data register $Reg(k)$. The former stores the positions of the nodes, whilst the latter stores the features. There are two blocks of gates.

The one to the left (which is applied first) first loads the features through gate X ,⁵ then the normalized adjacency matrix $\hat{A}$ by block-encoding it into $U_{\hat{A}}$ — hence the extra ancillary qubits. Simultaneously, a trainable gate (ansatz) is applied on the data register.

The block to the right is a single gate that loads the labels used for a classification task. In order to compare them, one runs a Hadamard test between the two blocks, as we will explain in Section B.1.2. If one wants to implement l layers, it is enough to block-encode $\hat{A}^l$ instead of $\hat{A}$.

Classical and Quantum Linear Graph Convolutions. The Linear Graph Convolution (LGC) effectively acts as a superposition of SGCs with varying layers. In our case, this translates to replacing the block encoded adjacency matrix $\hat{A}$ with a polynomial $P(\hat{A}) = \sum_{p=1}^K a_p \hat{A}^p$. Here, the coefficients a_p sum up to 1. By allowing the coefficients a_p to be trainable, both classical LGCs can represent a much richer class of graph convolution filters compared to SGC. This increased expressivity enables LGC to capture more complex graph structures and long-range dependencies, leading to improved performance on certain downstream tasks.[41] In the quantum case, this is implemented using a QSVT routine, as derived in [13].

B.1.1 Data loading

In the usual, classical graph convolutional networks, the data is represented as a feature vector $\vec{X}_i \in \mathbb{R}^C$ assigned for each node i . Here, C is the number of features assigned per node (each feature slot being a *channel*). By stacking these vectors, we obtain the feature matrix $X \in \mathbb{R}^{N \times C}$, which we load (after normalization) as the following quantum state:

$$|\psi_X\rangle = \sum_{i=1}^N |i\rangle |\mathbf{x}_i\rangle \propto \sum_{i=1}^N \sum_{k=1}^C X_{ik} |i\rangle |k\rangle. \quad (25)$$

⁵We will see how X is loaded in Section B.1.1.

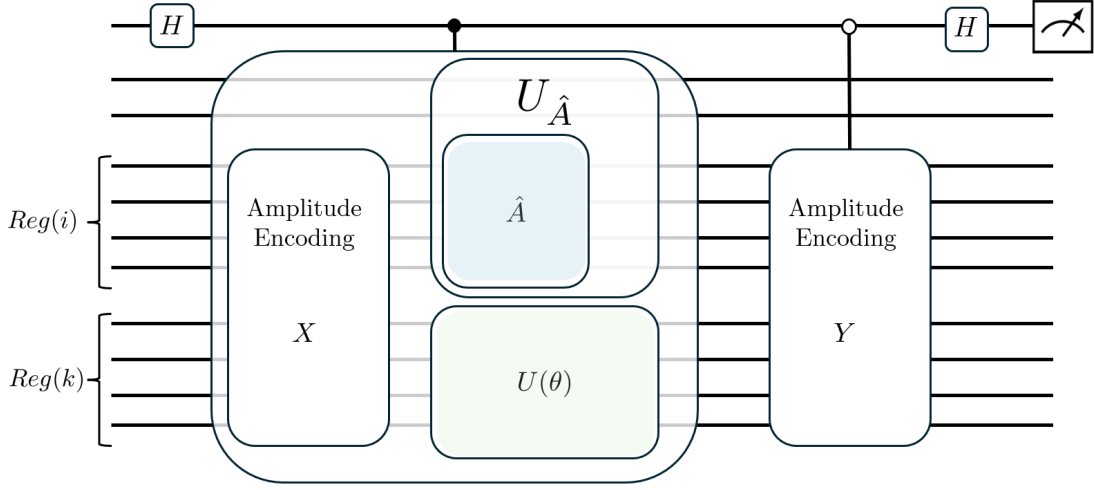


Figure 2: Quantum circuit implementing QSGC network. The cost function is evaluated by the Modified Hadamard test. This is a larger version of Fig. 1.

The entire state is prepared on two quantum registers hosting the channel index k and node index i , which are denoted as $Reg(k)$ and $Reg(i)$, respectively (see Fig. 2). Therefore, this encoding corresponds to an *amplitude encoding*, of the vectorized node feature matrix. In this work, the method from Ref. [42] is chosen, as their work provides a tunable trade-off between the number of ancillary qubits and the circuit depth for state preparation.

B.1.2 Cost function evaluation

In order to be able to compute a reasonable cost function directly on the quantum circuit, the following choice of cost⁶ is used (as given in Eq. 1):

$$L_{QGCN} = -\text{Re}(\langle \psi_{out} | \psi_Y \rangle). \quad (26)$$

Here, the state $|\psi_Y\rangle$ is a one-hot encoding of the target state:

$$|\psi_Y\rangle = \sum_{s \in Y_L} \sum_{f=1}^{F_K} Y_{sf} |s\rangle |f\rangle, \quad (27)$$

with $Y_{sf} = [\vec{Y}_s]_f$ reconstructing the target classes. The loading process is analogous to that of X . The state $|\psi_Y\rangle$ can be efficiently amplitude-encoded using the algorithm proposed in Ref. [43].

Finally, the cost function L_{QGCN} can be evaluated via the *Modified Hadamard Test* [14, 15]. As seen in Fig. 2, we add an extra qubit and controlling the gates generating $|\psi_{out}\rangle$ on the $|0\rangle$ state and $|\psi_Y\rangle$ on the $|1\rangle$ state. Then, we can measure the added qubit over several runs to form a probability distribution such that

$$p(0) = \frac{2 + 2\text{Re}[\langle \psi_{out} | \psi_Y \rangle]}{4}. \quad (28)$$

C Detailed trainability calculations

Here we describe in more detail the motivation behind the cost function as well as derivations for the results in Section 3. Furthermore, we work through a simplified version of the large-graph limit and give an example through the use of the IQP ansatz.

Why QSGC? Here, we focus on the single QSGC layer because it lies at the core of all other models. The generalization for any K -layer QSGC as well as QLGC models is done in Appendix D. Whilst the absence of nonlinearities (such as activation functions implemented using e.g. NTCA [44]) maintains a low expressivity, it protects the loss landscape from acquiring trivial barren plateaus.

⁶This cost is the same across all QGCN architectures defined in [13], including the QSGC.

C.1 Motivation of the cost function

A natural measure of the overlap between two states (the output and the target) is their square distance, which in this case is

$$L_{\text{distance}} := \| |\psi_{\text{out}}\rangle - |\psi_Y\rangle \|^2. \quad (29)$$

By expanding this definition, we reach

$$\begin{aligned} L_{\text{distance}} &= \langle \psi_{\text{out}} | \psi_{\text{out}} \rangle - \langle \psi_{\text{out}} | \psi_Y \rangle - \langle \psi_Y | \psi_{\text{out}} \rangle + \langle \psi_Y | \psi_Y \rangle \\ &= 2(1 - \text{Re} \langle \psi_{\text{out}} | \psi_Y \rangle) \\ &= 2(1 + L_{QGCN}). \end{aligned}$$

Therefore, the constant term as well as the prefactor of 2 can be dropped as they do not contribute to the dependence of the gradient variance on the input size. For future reference, we write the cost function $L_{QGCN} = (L_1 + L_1^*)/2$, with $L_1 = -\langle \psi_{\text{out}} | \psi_Y \rangle$.

C.2 Notation and explicit form of L_1

We give the notations for the index and data register qubit numbers $n_i \equiv \log_2(N_i)$, $n_k \equiv \log_2(N_k)$, so N_i and N_k are the dimensions of the respective Hilbert spaces. This means that the number of nodes N and the dimension of the feature vectors C are bounded by $N_i/2 < N \leq N_i$ and $N_k/2 < C \leq N_k$. Therefore, their orders match, $N_i = O(N)$ and $N_k = O(C)$.

Let us now calculate L_1 explicitly.

The normalized input and target states are

$$|\psi_X\rangle = \frac{1}{\sqrt{N_i}} \sum_{i=1}^{N_i} |i\rangle |\mathbf{x}_i\rangle, \quad (30)$$

$$|\psi_Y\rangle = \frac{1}{\sqrt{N_i}} \sum_{j=1}^{N_i} |j\rangle |\mathbf{y}_j\rangle. \quad (31)$$

Here, i, j label the nodes of the graph. Next is the output state:

$$|\psi_{\text{out}}\rangle = \left(\hat{A} \otimes U(\boldsymbol{\theta}) \right) |\psi_X\rangle, \quad (32)$$

where $\hat{A}$ is the normalized adjacency matrix with self loops as defined in Appendix A. This is not a unitary gate; it is implemented through block encoding in a larger circuit⁷, but can be used as such in our calculation.

This leads to the inner product:

$$L_1 = -\frac{1}{N_i} \sum_{i,j=1}^{N_i} \langle j | \hat{A} | i \rangle \langle \mathbf{y}_j | U(\boldsymbol{\theta}) | \mathbf{x}_i \rangle \quad (33)$$

$$= -\frac{1}{N_i} \sum_{i,j=1}^{N_i} \hat{A}_{ij} \langle \mathbf{y}_j | U(\boldsymbol{\theta}) | \mathbf{x}_i \rangle. \quad (34)$$

Here, the ansatz $U(\boldsymbol{\theta})$ is generic; however, with minimal loss of generality, we can assume that the ansatz takes the following form⁸:

$$U(\boldsymbol{\theta}) := \prod_{\eta=D}^1 \exp(-i\theta_\eta V_\eta) \cdot W_\eta. \quad (35)$$

⁷We note that this can be done as $\hat{A}$ is a subunitary, in the sense that its operator norm is $\|\hat{A}\|_{\text{op}} \leq 1$.

⁸We note that we could add a fixed gate W_{D+1} at the end of the circuit, as $U(\boldsymbol{\theta}) := W_{D+1} \cdot \prod_{\eta=D}^1 \exp(-i\theta_\eta V_\eta) \cdot W_\eta$, as is usual in many ansätze. The results are derived identically if one keep this gate into account.

The parameters $\boldsymbol{\theta} = (\theta_1, \theta_2, \dots, \theta_D)$ are trainable, D is thus the dimension of $\boldsymbol{\theta}$. This is not guaranteed to be equal to the depth of the ansatz, as usually some of the gates containing these parameters can be performed simultaneously. The number D is usually related to the size of the data register n_k only. Finally, the product ranges from D to 1 (descending) as the order of the parametrized gates is from right to left in order to be applied in ascending order to the state $|\mathbf{x}_i\rangle$.

The unitary operators V_η and W_η are not trainable, but ansatz-dependent. V_k is also Hermitian and squares to the identity, which leads to a useful formula:

$$\exp(-i\theta_\eta V_\eta) = \cos \theta_\eta \mathbb{1} - i \sin \theta_\eta V_\eta. \quad (36)$$

This can be seen by taking the Taylor expansion of the LHS and separating by even and odd terms.

C.3 Derivation of the gradient variance

To derive the gradient variance for the Quantum Simple Graph Convolution, L_{QSGC} , we start by decomposing it in terms of L_1 , as

$$\text{Var}(\partial_\mu L_{QSGC}) = \frac{1}{4} (\text{Var}(\partial_\mu L_1) + \text{Var}(\partial_\mu L_1^*) + 2\text{Cov}(\partial_\mu L_1, \partial_\mu L_1^*)). \quad (37)$$

Here, $\partial_\mu \equiv \partial/\partial\theta_\mu$ is the μ -th direction of the gradient. Computing the building block of Eq. 37:

$$\partial_\mu L_1 = -\frac{\partial}{\partial\theta_\mu} \frac{1}{N_i} \sum_{i,j=1}^{N_i} \hat{A}_{ij} \langle \mathbf{y}_j | U(\boldsymbol{\theta}) | \mathbf{x}_i \rangle \quad (38)$$

$$= -\frac{1}{N_i} \sum_{i,j=1}^{N_i} \hat{A}_{ij} \langle \mathbf{y}_j | \partial_\mu U(\boldsymbol{\theta}) | \mathbf{x}_i \rangle. \quad (39)$$

For the ansatz, we have

$$\partial_\mu U(\boldsymbol{\theta}) = \partial_\mu \prod_{\eta=D}^1 \exp(-i\theta_\eta V_\eta) \cdot W_\eta \quad (40)$$

$$= \underbrace{\left(\prod_{\eta=D}^{\mu+1} \exp(-i\theta_\eta V_\eta) \cdot W_\eta \right)}_{U_L} \partial_\mu \exp(-i\theta_\mu V_\mu) \cdot W_\mu \underbrace{\left(\prod_{\eta=\mu-1}^1 \exp(-i\theta_\eta V_\eta) \cdot W_\eta \right)}_{U_R} \quad (41)$$

$$= U_L (-iV_\mu \exp(-i\theta_\mu V_\mu) \cdot W_\mu) U_R. \quad (42)$$

Expected value is zero. By taking the expected value of $\partial_\mu L_1$:

$$\begin{aligned} \mathbb{E}_{\boldsymbol{\theta}}[\partial_\mu L_1] &= \frac{1}{(2\pi)^D} \int d\boldsymbol{\theta} (\partial_\mu L_1) \\ &= \frac{1}{(2\pi)^D} \int d\boldsymbol{\theta}' \int d\theta_\mu \frac{1}{N_i} \sum_{i,j=1}^{N_i} \hat{A}_{ij} \langle \mathbf{y}_j | \partial_\mu U(\boldsymbol{\theta}) | \mathbf{x}_i \rangle \\ &= \frac{1}{(2\pi)^D N_i} \sum_{i,j=1}^{N_i} \hat{A}_{ij} \int d\boldsymbol{\theta}' \langle \mathbf{y}_j | U(\boldsymbol{\theta}', \theta_\mu = \pi) - U(\boldsymbol{\theta}', \theta_\mu = -\pi) | \mathbf{x}_i \rangle. \end{aligned} \quad (43)$$

Here, all weights θ_η range from $-\pi$ to π . The full integral over $\boldsymbol{\theta}$ was separated into an integral over θ_μ and the integral over all other parameters, denoted $\boldsymbol{\theta}'$. We notice that thanks to Eq. 36, $\exp(-i(\pm\pi)V_\mu) = \mathbb{1}$ and thus the two terms in the bracket above cancel out. By a similar calculation, this extends to the whole cost gradient, so $\mathbb{E}_{\boldsymbol{\theta}}[L_{QSGC}] = 0$.

Now we can calculate the gradient variance considering the range of all parameters to be $[-\pi, \pi] \ni \theta_\eta$. When integrating over the parameters, this range is assumed.

$$\text{Var}(\partial_\mu L_1) = \frac{1}{(2\pi)^D} \int d\boldsymbol{\theta} (\partial_\mu L_1)^2 \quad (44)$$

$$= \frac{1}{(2\pi)^D} \int d\boldsymbol{\theta}' \int d\theta_\mu \frac{1}{N_i^2} \sum_{i,j=1}^{N_i} \hat{A}_{ij} \langle \mathbf{y}_j | \partial_\mu U(\boldsymbol{\theta}) | \mathbf{x}_i \rangle \sum_{k,l=1}^{N_i} \hat{A}_{kl} \langle \mathbf{y}_l | \partial_\mu U(\boldsymbol{\theta}) | \mathbf{x}_k \rangle \quad (45)$$

$$= \frac{1}{(2\pi)^D N_i^2} \sum_{i,j,k,l=1}^{N_i} \hat{A}_{ij} \hat{A}_{kl} \int d\boldsymbol{\theta}' \int d\theta_\mu \langle \mathbf{y}_j | \partial_\mu U(\boldsymbol{\theta}) | \mathbf{x}_i \rangle \langle \mathbf{y}_l | \partial_\mu U(\boldsymbol{\theta}) | \mathbf{x}_k \rangle. \quad (46)$$

The same separation into θ_μ and $\boldsymbol{\theta}'$ was applied as for the expected value calculation.

Let us start with the term (i, j, k, l) and integrate it over θ_μ , using Eqs. 36 and 42. We denote this by I_μ :

$$I_\mu = \int d\theta_\mu \langle \mathbf{y}_j | \partial_\mu U(\boldsymbol{\theta}) | \mathbf{x}_i \rangle \langle \mathbf{y}_l | \partial_\mu U(\boldsymbol{\theta}) | \mathbf{x}_k \rangle \quad (47)$$

$$= \int d\theta_\mu \langle \mathbf{y}_j | U_L (-iV_\mu \exp(-i\theta_\mu V_\mu) \cdot W_\mu) U_R | \mathbf{x}_i \rangle \langle \mathbf{y}_l | U_L (-iV_\mu \exp(-i\theta_\mu V_\mu) \cdot W_\mu) U_R | \mathbf{x}_k \rangle \quad (48)$$

$$= - \int d\theta_\mu \langle \mathbf{y}_j | U_L V_\mu (\cos \theta_\mu \mathbb{1} - i \sin \theta_\mu V_\mu) W_\mu U_R | \mathbf{x}_i \rangle \cdot \langle \mathbf{y}_l | U_L V_\mu (\cos \theta_\mu \mathbb{1} - i \sin \theta_\mu V_\mu) W_\mu U_R | \mathbf{x}_k \rangle \quad (49)$$

$$\begin{aligned} &= - \int d\theta_\mu (\cos \theta_\mu)^2 \langle \mathbf{y}_j | U_L V_\mu W_\mu U_R | \mathbf{x}_i \rangle \langle \mathbf{y}_l | U_L V_\mu W_\mu U_R | \mathbf{x}_k \rangle \\ &\quad + \int d\theta_\mu (\sin \theta_\mu)^2 \langle \mathbf{y}_j | U_L V_\mu^2 W_\mu U_R | \mathbf{x}_i \rangle \langle \mathbf{y}_l | U_L V_\mu^2 W_\mu U_R | \mathbf{x}_k \rangle \\ &\quad - i \int d\theta_\mu \cos \theta_\mu \sin \theta_\mu \langle \mathbf{y}_j | U_L V_\mu W_\mu U_R | \mathbf{x}_i \rangle \langle \mathbf{y}_l | U_L V_\mu^2 W_\mu U_R | \mathbf{x}_k \rangle \\ &\quad - i \int d\theta_\mu \cos \theta_\mu \sin \theta_\mu \langle \mathbf{y}_j | U_L V_\mu^2 W_\mu U_R | \mathbf{x}_i \rangle \langle \mathbf{y}_l | U_L V_\mu W_\mu U_R | \mathbf{x}_k \rangle. \end{aligned} \quad (50)$$

The second equality comes from Eq. 36, the third from Eq. 42, which we then separate by decomposing the product of brackets. One can now check that the first two trigonometric integrals are equal to π and the last two are 0. Considering also that $V_\mu^2 = \mathbb{1}$, we reach the final relation

$$I_\mu = \pi (\langle \mathbf{y}_j | U_L W_\mu U_R | \mathbf{x}_i \rangle \langle \mathbf{y}_l | U_L W_\mu U_R | \mathbf{x}_k \rangle - \langle \mathbf{y}_j | U_L V_\mu W_\mu U_R | \mathbf{x}_i \rangle \langle \mathbf{y}_l | U_L V_\mu W_\mu U_R | \mathbf{x}_k \rangle). \quad (51)$$

Fortunately, after this tedious calculation, the other integrals follow. The only difference is that in this one extra V_μ factors were present, which lead to a sign cancellation only as the resulting terms were swapped. Therefore, each integral (over e.g. θ_λ) takes the current terms and splits each into two: one product of brackets that do not contain the V_λ minus one product of brackets that do. By counting all of these terms with a D -bit counter $\vec{\alpha}$, we obtain the desired result for the variance of $\partial_\mu L_1$:

$$\text{Var}(\partial_\mu L_1) = \frac{1}{2^D N_i^2} \sum_{i,j,k,l=1}^{N_i} \hat{A}_{ij} \hat{A}_{kl} \sum_{\vec{\alpha} \in \{0,1\}^D} (-1)^{w(\vec{\alpha})} \langle \mathbf{y}_j | \prod_{\eta=D}^1 V_\eta^{\alpha_\eta} W_\eta | \mathbf{x}_i \rangle \langle \mathbf{y}_l | \prod_{\eta=D}^1 V_\eta^{\alpha_\eta} W_\eta | \mathbf{x}_k \rangle. \quad (52)$$

The string $\vec{\alpha}$ tracks the branch we took to follow this term as we computed the integrals, and the number of minuses factored comes from the numbers of 1s present in the string, thus the Hamming weight $w(\vec{\alpha})$. Reducing it to a simpler form:

$$\text{Var}(\partial_\mu L_1) = \frac{1}{2^D N_i^2} \sum_{\vec{\alpha} \in \{0,1\}^D} (-1)^{w(\vec{\alpha})} \left(\sum_{i,j=1}^{N_i} \hat{A}_{ij} \langle \mathbf{y}_j | \prod_{\eta=D}^1 V_\eta^{\alpha_\eta} W_\eta | \mathbf{x}_i \rangle \right)^2. \quad (53)$$

We can do a similar calculation for $\text{Var}(\partial_\mu L_1^*)$, but first we note that this is not trivially the complex conjugate of Eq. 52. The derivative of L_1^* is

$$\partial_\mu L_1^* = \partial_\mu \left[\frac{1}{N_i} \sum_{i,j=1}^{N_i} \langle \mathbf{x}_i | \prod_{\eta=1}^D W_\eta^\dagger \exp(+i\theta_\eta V_\eta) | \mathbf{y}_j \rangle \right], \quad (54)$$

thus instead of extracting a $-iV_\mu$ factor from the exponential as done previously, now a plus sign appears. Therefore,

$$\partial_\mu L_1^* = -(\partial_\mu L_1)^*. \quad (55)$$

When computing the variance, the minus sign above is squared and vanishes, hence

$$\text{Var}(\partial_\mu L_1^*) = \frac{1}{2^D N_i^2} \sum_{\vec{\alpha} \in \{0,1\}^D} (-1)^{w(\vec{\alpha})} \left(\sum_{i,j=1}^{N_i} \hat{A}_{ij} \langle \mathbf{y}_j | \prod_{\eta=1}^D W_\eta^\dagger V_\eta^{\alpha_\eta} | \mathbf{x}_i \rangle \right)^2. \quad (56)$$

Finally, the calculation for the covariance is very similar, except the decomposition in line 49. Here, one of the brackets is complex conjugated, as

$$\langle \mathbf{y}_j | U_L V_\mu (\cos \theta_\mu \mathbb{1} - i \sin \theta_\mu V_\mu) W_\mu U_R | \mathbf{x}_i \rangle \langle \mathbf{x}_k | U_L W_\mu^\dagger V_\mu (\cos \theta_\mu \mathbb{1} + i \sin \theta_\mu V_\mu) U_R | \mathbf{y}_l \rangle, \quad (57)$$

and the plus means that when separating the terms, both the $\cos^2$ and the $\sin^2$ have a positive sign, hence we drop the $(-1)^{w(\vec{\alpha})}$ in the answer for the covariance:

$$\text{Cov}(\partial_\mu L_1, \partial_\mu L_1^*) = \frac{1}{2^D N_i^2} \sum_{\vec{\alpha} \in \{0,1\}^D} \left| \sum_{i,j=1}^{N_i} \hat{A}_{ij} \langle \mathbf{y}_j | \prod_{\eta=D}^1 V_\eta^{\alpha_\eta} W_\eta | \mathbf{x}_i \rangle \right|^2. \quad (58)$$

C.4 Results for regular graphs

For a k -regular graph, each node has a degree of $k+1$ (including self-connections). By applying Eq. 6, $\hat{A}_{ij} = 1/\sqrt{\deg(i)\deg(j)} = 1/(k+1)$ if i and j are connected by an edge, and 0 otherwise. Here we noted the degree of node i by $\deg(i)$. It is easier to work with the separate terms derived above, namely Eqs. 53 and 58. The former equation becomes

$$\text{Var}(\partial_\mu L_1) = \frac{1}{2^D N_i^2 (k+1)^2} \sum_{\vec{\alpha} \in \{0,1\}^D} \left(\sum_{\langle i,j \rangle} \langle \mathbf{y}_j | \prod_{\eta=D}^1 V_\eta^{\alpha_\eta} W_\eta | \mathbf{x}_i \rangle \right)^2, \quad (59)$$

whilst the latter is simply

$$\text{Cov}(\partial_\mu L_1, \partial_\mu L_1^*) = \frac{1}{2^D N_i^2 (k+1)^2} \sum_{\vec{\alpha} \in \{0,1\}^D} \left| \sum_{\langle i,j \rangle} \langle \mathbf{y}_j | \underbrace{\prod_{\eta=D}^1 V_\eta^{\alpha_\eta} W_\eta}_{\equiv D_{\vec{\alpha}}} | \mathbf{x}_i \rangle \right|^2. \quad (60)$$

By $\langle i, j \rangle$ we mean that nodes i and j must be connected (this includes the self loops $\langle i, i \rangle$). For further reference we use the $D_{\vec{\alpha}}$ for the ansatz terms.

C.5 Expected values over the input and target states

The expected value of the cost gradient variance is done by considering the distributions of input and target feature vectors respectively, and averaging each node feature over these distributions.

Firstly, we take a look at the encoding of the input and target states, respectively $|\mathbf{x}_i\rangle$ and $|\mathbf{y}_i\rangle$, for node i . The target states are one-hot encoded onto some basis; let us call it $\{|\mathbf{e}_a\rangle\}_{a \in [1, N_k]}$. The input states are

multi-hot encoded, meaning that each state $|\mathbf{x}_i\rangle$ is equal to some equal superposition of a subset of basis states, based on the encoded input bit string, call it $\vec{\gamma}$.⁹ We write this as

$$\begin{aligned} |\mathbf{y}_i\rangle &= |\mathbf{e}_a\rangle, \\ |\mathbf{x}_i\rangle &= \frac{1}{\sqrt{w(\vec{\gamma})}} \sum_{b=1}^{N_k} \gamma_b |\mathbf{e}_b\rangle. \end{aligned} \quad (61)$$

The total number of possible target states is N_k , equal to the that of basis states. The number of possible input states is that of all non-zero N_k -bit strings, which is $2^{N_k} - 1$.

To this end, the input and target states are considered independent random variables¹⁰, and are independent across the nodes of the graph. Therefore, the expected value operator $\mathbb{E}[\cdot]$ acts on any function f of the states $|\mathbf{x}_i\rangle$ and $|\mathbf{y}_i\rangle$ as

$$\begin{aligned} \mathbb{E}[f(|\mathbf{y}_i\rangle)] &= \frac{1}{N_k} \sum_{a=1}^{N_k} f(|\mathbf{e}_a\rangle), \\ \mathbb{E}[f(|\mathbf{x}_i\rangle)] &= \frac{1}{2^{N_k} - 1} \sum_{\substack{\vec{\gamma} \in \{0,1\}^{N_k} \\ \vec{\gamma} \neq \vec{0}}} f\left(\frac{1}{\sqrt{w(\vec{\gamma})}} \sum_{b=1}^{N_k} \gamma_b |\mathbf{e}_b\rangle\right). \end{aligned} \quad (62)$$

Note that we use states, rather than density operators. This is because the distribution actually runs over decisions for the (classical) inputs and targets, that are then implemented onto the QGNN. This is a classical sampling process where each of the runs has a determined input and target, rather than running any of them with a probability and “forgetting” which decision was taken.

In order to find the expected value of the variance, $\mathbb{E}[\text{Var}(\partial_\mu L_{QSGC})]$, we will average the three separate terms using Eq. 37 first:

$$\mathbb{E}[\text{Var}(\partial_\mu L_1)] = \frac{1}{2^D N_i^2 (k+1)^2} \sum_{\vec{\alpha} \in \{0,1\}^D} (-1)^{w(\vec{\alpha})} \mathbb{E} \left[\left(\sum_{\langle i,j \rangle} \langle \mathbf{y}_j | D_{\vec{\alpha}} | \mathbf{x}_i \rangle \right)^2 \right], \quad (63)$$

$$\mathbb{E}[\text{Var}(\partial_\mu L_1^*)] = \mathbb{E}[\text{Var}(\partial_\mu L_1)]^*, \quad (64)$$

$$\mathbb{E}[\text{Cov}(\partial_\mu L_1, \partial_\mu L_1^*)] = \frac{1}{2^D N_i^2 (k+1)^2} \sum_{\vec{\alpha} \in \{0,1\}^D} \mathbb{E} \left[\left| \sum_{\langle i,j \rangle} \langle \mathbf{y}_j | D_{\vec{\alpha}} | \mathbf{x}_i \rangle \right|^2 \right]. \quad (65)$$

By expanding the square terms from Eqs. 63 and 65,

$$\mathbb{E} \left[\sum_{\substack{\langle i,j \rangle \\ \langle k,l \rangle}} \langle \mathbf{y}_j | D_{\vec{\alpha}} | \mathbf{x}_i \rangle \langle \mathbf{y}_l | D_{\vec{\alpha}} | \mathbf{x}_k \rangle \right] \equiv S_{\vec{\alpha}}^v, \quad \text{and} \quad \mathbb{E} \left[\sum_{\substack{\langle i,j \rangle \\ \langle k,l \rangle}} \langle \mathbf{y}_j | D_{\vec{\alpha}} | \mathbf{x}_i \rangle \langle \mathbf{x}_k | D_{\vec{\alpha}}^\dagger | \mathbf{y}_l \rangle \right] \equiv S_{\vec{\alpha}}^c, \quad (66)$$

we observe that the averages are taken over pairs of nodes (i, j) and (k, l) . As such, one needs to take into account whether the (directed) edges are disconnected, or have one mutual node, or both, as each of these cases requires a different distribution over the input and target random variables. For simplicity, we denote the expressions by $S_{\vec{\alpha}}^v$ and $S_{\vec{\alpha}}^c$ (which will be used for the variance and covariance terms respectively).

C.5.1 Dependence of the variance with the graph size N_i

When considering the distribution over $|\mathbf{x}_i\rangle$, $|\mathbf{y}_j\rangle$, $|\mathbf{x}_k\rangle$, and $|\mathbf{y}_l\rangle$, we need to separate the following cases based on whether $i = k$ or $j = l$ or both, or neither.

⁹Notice in Eq. 61 that because of the equal superposition of basis states, the normalisation factor depends on the Hamming weight of the bit string $\vec{\gamma}$.

¹⁰In practice, the input and target will have hidden correlations; otherwise, the model will not learn any useful information. However, for our purposes, we take them as independent for the sake of estimating the expected value.

Table 1: Counting the number of possible combinations of pairs of edges based on a constraint (in the first column). The values are denoted by $R_{||}$, $R_{\wedge}$, $R_{\vee}$, and $R_{|}$, respectively.

Constraint	Independent variables	Number of cases	Denoted as
$i \neq k, j \neq l$	$ \mathbf{x}_i\rangle, \mathbf{x}_k\rangle, \mathbf{y}_j\rangle, \mathbf{y}_l\rangle$	$N_i(N_i - 1)(k + 1)^2 - N_i k(k + 1)$	$R_{ }$
$i = k, j \neq l$	$ \mathbf{x}_i\rangle, \mathbf{y}_j\rangle, \mathbf{y}_l\rangle$	$N_i k(k + 1)$	$R_{\wedge}$
$i \neq k, j = l$	$ \mathbf{x}_i\rangle, \mathbf{x}_k\rangle, \mathbf{y}_j\rangle$	$N_i k(k + 1)$	$R_{\vee}$
$i = k, j = l$	$ \mathbf{x}_i\rangle, \mathbf{y}_j\rangle$	$N_i(k + 1)$	$R_{ }$

In Table 1, we count all the pairs of edges categorized by the constraint on whether input or target feature vectors coincide or not and denote them using the R -notation given in the last column¹¹. Rewriting the expected values from Eq. 66:

$$S_{\vec{\alpha}}^v = R_{||} \mathbb{E} [\langle \mathbf{y}_j | D_{\vec{\alpha}} | \mathbf{x}_i \rangle \langle \mathbf{y}_l | D_{\vec{\alpha}} | \mathbf{x}_k \rangle] + R_{\wedge} \mathbb{E} [\langle \mathbf{y}_j | D_{\vec{\alpha}} | \mathbf{x}_i \rangle \langle \mathbf{y}_l | D_{\vec{\alpha}} | \mathbf{x}_i \rangle] \\ + R_{\vee} \mathbb{E} [\langle \mathbf{y}_j | D_{\vec{\alpha}} | \mathbf{x}_i \rangle \langle \mathbf{y}_j | D_{\vec{\alpha}} | \mathbf{x}_k \rangle] + R_{|} \mathbb{E} [\langle \mathbf{y}_j | D_{\vec{\alpha}} | \mathbf{x}_i \rangle \langle \mathbf{y}_j | D_{\vec{\alpha}} | \mathbf{x}_i \rangle], \quad (67)$$

and

$$S_{\vec{\alpha}}^c = R_{||} \mathbb{E} [\langle \mathbf{y}_j | D_{\vec{\alpha}} | \mathbf{x}_i \rangle \langle \mathbf{x}_k | D_{\vec{\alpha}} | \mathbf{y}_l \rangle] + R_{\wedge} \mathbb{E} [\langle \mathbf{y}_j | D_{\vec{\alpha}} | \mathbf{x}_i \rangle \langle \mathbf{x}_i | D_{\vec{\alpha}} | \mathbf{y}_l \rangle] \\ + R_{\vee} \mathbb{E} [\langle \mathbf{y}_j | D_{\vec{\alpha}} | \mathbf{x}_i \rangle \langle \mathbf{x}_k | D_{\vec{\alpha}} | \mathbf{y}_j \rangle] + R_{|} \mathbb{E} [\langle \mathbf{y}_j | D_{\vec{\alpha}} | \mathbf{x}_i \rangle \langle \mathbf{x}_i | D_{\vec{\alpha}} | \mathbf{y}_j \rangle]. \quad (68)$$

Simple asymptotic analysis. In order to calculate the asymptotic behavior of the gradient variance with respect to the graph size (quantified by N_i and k), one can notice that the expected values of different node pairs are only dependent on the size of the distributions, thus on N_k , and the terms $D_{\vec{\alpha}}$ derived from the ansatz. Whilst our analysis considers N_i and N_k as independent parameters, all standard ansätze used depend strictly on the data register size, hence also on the logarithm of N_k . It is thus safe to assume that all of the edge pair expected values are equal to $O_{\text{gr}}(1)$ in both the node number N_i and the average connectivity k .¹² As such, both of the summations above have the order of the leading term, $S_{\vec{\alpha}}^v, S_{\vec{\alpha}}^c = O_{\text{gr}}(R_{||}(N_i, k))$. This implies that, when keeping N_k constant,

$$\mathbb{E}[\text{Var}(\partial_{\mu} L_{QSGC})] = \frac{1}{4} (\mathbb{E}[\text{Var}(\partial_{\mu} L)] + \mathbb{E}[\text{Var}(\partial_{\mu} L)^*] + 2\mathbb{E}[\text{Cov}(\partial_{\mu} L_1, \partial_{\mu} L_1^*)]) \\ = \frac{1}{2^D N_i^2 (k + 1)^2} \sum_{\vec{\alpha} \in \{0,1\}^D} \left[(-1)^{w(\vec{\alpha})} ((S_{\vec{\alpha}}^v) + (S_{\vec{\alpha}}^v)^*) + 2(S_{\vec{\alpha}}^c) \right] \\ = \frac{2}{2^D N_i^2 (k + 1)^2} \sum_{\vec{\alpha} \in \{0,1\}^D} \left[(-1)^{w(\vec{\alpha})} \text{Re}(S_{\vec{\alpha}}^v) + S_{\vec{\alpha}}^c \right] \quad (69) \\ = \frac{2}{2^D N_i^2 (k + 1)^2} \sum_{\vec{\alpha} \in \{0,1\}^D} \left[(-1)^{w(\vec{\alpha})} O_{\text{gr}}(R_{||}) + O_{\text{gr}}(R_{||}) \right] \\ = \frac{2}{2^D N_i^2 (k + 1)^2} O_{\text{gr}}(N_i^2 (k + 1)^2) \\ = O_{\text{gr}}(1). \quad (70)$$

Therefore, to leading order, the normalization of the adjacency matrix exactly cancels the summation of the different terms. Unless both the expected values $\mathbb{E} [\langle \mathbf{y}_j | D_{\vec{\alpha}} | \mathbf{x}_i \rangle \langle \mathbf{y}_l | D_{\vec{\alpha}} | \mathbf{x}_k \rangle]$ and $\mathbb{E} [\langle \mathbf{y}_j | D_{\vec{\alpha}} | \mathbf{x}_i \rangle \langle \mathbf{x}_k | D_{\vec{\alpha}}^{\dagger} | \mathbf{y}_l \rangle]$ vanish¹³, the bound is strict as function of the graph size and connectivity, $\mathbb{E}[\text{Var}(\partial_{\mu} L_{QSGC})] = \Theta_{\text{gr}}(1)$.

¹¹The indices correspond to whether the edges have zero common nodes (“||”), or the $|\mathbf{x}\rangle$ or $|\mathbf{y}\rangle$ in common (“ $\wedge$ ” or “ $\vee$ ”), or both (“|”).

¹²Here we use the “graph” subscript O_{gr} (and Θ_{gr} etc.) to restrict the result to large N_i or k asymptotes, for fixed N_k .

¹³We will see later that this requires the ansatz itself to be very restrictive.

We remind the reader that this is only an evaluation of the average variance over all possible inputs and targets. One can also find approximate (asymptotic) bounds of the full variance by also considering the spread $\Delta_{i,t}[\text{Var}(\partial_\mu L)]$. One option is to approximate it further with a standard deviation, taking the variance $\sqrt{\text{Var}_{i,t}(\text{Var}(\partial_\mu L_{QSGC}))}$, but this may be tricky as it assumes a symmetric spread which may not be true. We leave this task for future work.

C.6 Calculation of S_α^v and S_α^c and dependence on N_k

Here, we calculate the explicit forms of the terms S_α^v and S_α^c , by computing each of the terms separately. We begin with the “||” term in Eq. 67.

$$\begin{aligned} \mathbb{E} [\langle \mathbf{y}_j | D_{\bar{\alpha}} | \mathbf{x}_i \rangle \langle \mathbf{y}_l | D_{\bar{\alpha}} | \mathbf{x}_k \rangle] &= \\ &= \frac{1}{N_k^2 (2^{N_k} - 1)^2} \left(\sum_{a=1}^{N_k} \langle e_a | \right) D_{\bar{\alpha}} \left(\sum_{\vec{\gamma} \neq 0} \frac{1}{\sqrt{w(\vec{\gamma})}} \sum_{b=1}^{N_k} \gamma_b | e_b \rangle \right) \left(\sum_{c=1}^{N_k} \langle e_c | \right) D_{\bar{\alpha}} \left(\sum_{\vec{\delta} \neq 0} \frac{1}{\sqrt{w(\vec{\delta})}} \sum_{d=1}^{N_k} \delta_d | e_d \rangle \right). \end{aligned}$$

Here, we used expressions given in Eq. 62, where a, b, c , and d label the basis indices, and $\vec{\gamma}$ and $\vec{\delta}$ the bit string labels ranging from 1 to 2^{N_k} . Rearranging the summations, the expression becomes

$$\frac{1}{N_k^2 (2^{N_k} - 1)^2} \left[\sum_{a,b,c,d} \langle e_a | D_{\bar{\alpha}} | e_b \rangle \langle e_c | D_{\bar{\alpha}} | e_d \rangle \underbrace{\left(\sum_{\vec{\gamma} \neq 0} \frac{\gamma_b}{\sqrt{w(\vec{\gamma})}} \right)}_{H'_1} \underbrace{\left(\sum_{\vec{\delta} \neq 0} \frac{\delta_d}{\sqrt{w(\vec{\delta})}} \right)}_{H'_1} \right]. \quad (71)$$

The last two summations can be done independently (using Lemma 8 in Appendix G). For now, we simply denote the sums by H'_1 , and will plug them in at the end. The expression for the desired term becomes

$$\mathbb{E} [\langle \mathbf{y}_j | D_{\bar{\alpha}} | \mathbf{x}_i \rangle \langle \mathbf{y}_l | D_{\bar{\alpha}} | \mathbf{x}_k \rangle] = \frac{(H'_1)^2}{N_k^2 (2^{N_k} - 1)^2} \left[\sum_{a,b,c,d} \langle e_a | D_{\bar{\alpha}} | e_b \rangle \langle e_c | D_{\bar{\alpha}} | e_d \rangle \right]. \quad (72)$$

Next, by considering the following relation

$$\frac{1}{\sqrt{N_k}} \sum_{a=1}^{N_k} | e_a \rangle = |\vec{1}\rangle =: |+\rangle^{\otimes n_k} = H^{\otimes n_k} |0\rangle^{\otimes n_k}, \quad (73)$$

the expression in Eq. 72 reduces to

$$\mathbb{E} [\langle \mathbf{y}_j | D_{\bar{\alpha}} | \mathbf{x}_i \rangle \langle \mathbf{y}_l | D_{\bar{\alpha}} | \mathbf{x}_k \rangle] = \frac{(H'_1)^2}{(2^{N_k} - 1)^2} \left(\langle \vec{1} | D_{\bar{\alpha}} | \vec{1} \rangle \right)^2. \quad (74)$$

For simplicity, we continue with the “v” term in Eq. 67,

$$\begin{aligned} \mathbb{E} [\langle \mathbf{y}_j | D_{\bar{\alpha}} | \mathbf{x}_i \rangle \langle \mathbf{y}_j | D_{\bar{\alpha}} | \mathbf{x}_k \rangle] &= \frac{1}{N_k (2^{N_k} - 1)^2} \sum_{a,b,d} \langle e_a | D_{\bar{\alpha}} | e_b \rangle \langle e_a | D_{\bar{\alpha}} | e_d \rangle \sum_{\vec{\gamma} \neq 0} \frac{\gamma_b}{\sqrt{w(\vec{\gamma})}} \sum_{\vec{\delta} \neq 0} \frac{\delta_d}{\sqrt{w(\vec{\delta})}} \\ &= \frac{(H'_1)^2}{N_k (2^{N_k} - 1)^2} \sum_{a,b,d} [D_{\bar{\alpha}}]_{ab} [D_{\bar{\alpha}}]_{ad} \\ &= \frac{(H'_1)^2}{(2^{N_k} - 1)^2} \langle \vec{1} | D_{\bar{\alpha}}^T D_{\bar{\alpha}} | \vec{1} \rangle, \end{aligned} \quad (75)$$

where in the last two lines we wrote the matrix elements of $D_{\bar{\alpha}}$ explicitly and then contracted using the state $|\vec{1}\rangle$.

The “ $\wedge$ ” term is calculated as

$$\mathbb{E}[\langle \mathbf{y}_j | D_{\bar{\alpha}} | \mathbf{x}_i \rangle \langle \mathbf{y}_l | D_{\bar{\alpha}} | \mathbf{x}_i \rangle] = \frac{1}{N_k^2(2^{N_k} - 1)} \sum_{a,b,c,d} \langle e_a | D_{\bar{\alpha}} | e_b \rangle \langle e_c | D_{\bar{\alpha}} | e_d \rangle \sum_{\vec{\gamma}} \frac{\gamma_b \gamma_d}{w(\vec{\gamma})}. \quad (76)$$

Here, the summation over $\vec{\gamma}$ branches into two cases, as can be seen in Eq. 182 in Lemma 8. We name these values H_1 and H'_2 , respectively for the cases where $b = d$ and $b \neq d$. The sum then reduces to

$$\frac{1}{N_k^2(2^{N_k} - 1)} \left[H'_2 \sum_{\substack{a,b,c,d \\ b \neq d}} \langle e_a | D_{\bar{\alpha}} | e_b \rangle \langle e_c | D_{\bar{\alpha}} | e_d \rangle + H_1 \underbrace{\sum_{a,b,c} \langle e_a | D_{\bar{\alpha}} | e_b \rangle \langle e_c | D_{\bar{\alpha}} | e_b \rangle}_{=N_k \langle \vec{1} | D_{\bar{\alpha}} D_{\bar{\alpha}}^T | \vec{1} \rangle} \right], \quad (77)$$

where the last sum was simplified in a similar manner to the previous term (Eq. 75). We can then artificially add and subtract a copy of the second sum into the first one, to get the following expression:

$$\begin{aligned} \mathbb{E}[\langle \mathbf{y}_j | D_{\bar{\alpha}} | \mathbf{x}_i \rangle \langle \mathbf{y}_l | D_{\bar{\alpha}} | \mathbf{x}_i \rangle] &= \frac{1}{N_k^2(2^{N_k} - 1)} \left[H'_2 \left(N_k^2 (\langle \vec{1} | D_{\bar{\alpha}} | \vec{1} \rangle)^2 - N_k \langle \vec{1} | D_{\bar{\alpha}} D_{\bar{\alpha}}^T | \vec{1} \rangle \right) \right. \\ &\quad \left. + H_1 N_k \langle \vec{1} | D_{\bar{\alpha}} D_{\bar{\alpha}}^T | \vec{1} \rangle \right] \\ &= \frac{H'_2}{2^{N_k} - 1} \left[(\langle \vec{1} | D_{\bar{\alpha}} | \vec{1} \rangle)^2 - \frac{1}{N_k} \langle \vec{1} | D_{\bar{\alpha}} D_{\bar{\alpha}}^T | \vec{1} \rangle \right] \\ &\quad + \frac{H_1}{N_k(2^{N_k} - 1)} \langle \vec{1} | D_{\bar{\alpha}} D_{\bar{\alpha}}^T | \vec{1} \rangle. \end{aligned} \quad (78)$$

To get this expression we wrote the summations over a, b, c , and d the same way as for the previous terms.

The final, “ $\vee$ ” term in Eq. 67 undergoes the same branching over two cases as done above; the only difference is that now, the bra vectors are entangled, rather than independent:

$$\begin{aligned} \mathbb{E}[\langle \mathbf{y}_j | D_{\bar{\alpha}} | \mathbf{x}_i \rangle \langle \mathbf{y}_j | D_{\bar{\alpha}} | \mathbf{x}_i \rangle] &= \frac{1}{N_k(2^{N_k} - 1)} \sum_{a,b,d} \langle e_a | D_{\bar{\alpha}} | e_b \rangle \langle e_a | D_{\bar{\alpha}} | e_d \rangle \sum_{\vec{\gamma} \neq 0} \frac{\gamma_b \gamma_d}{w(\vec{\gamma})} \\ &= \frac{H'_2}{2^{N_k} - 1} \left[\langle \vec{1} | D_{\bar{\alpha}}^T D_{\bar{\alpha}} | \vec{1} \rangle - \frac{1}{N_k} \text{Tr}(D_{\bar{\alpha}} D_{\bar{\alpha}}^T) \right] \\ &\quad + \frac{H_1}{N_k(2^{N_k} - 1)} \text{Tr}(D_{\bar{\alpha}} D_{\bar{\alpha}}^T), \end{aligned} \quad (79)$$

where the traces arise from the summation of the form $\sum_{a,b} (\langle e_a | D_{\bar{\alpha}} | e_b \rangle)^2$.

To this end, we can now add up the terms to reconstruct $S_{\bar{\alpha}}^v$ using Eq. 67, as well as the earlier calculations for each term.

$$\begin{aligned} S_{\bar{\alpha}}^v &= \frac{R_{||}(H'_1)^2}{(2^{N_k} - 1)^2} \left(\langle \vec{1} | D_{\bar{\alpha}} | \vec{1} \rangle \right)^2 \\ &\quad + \frac{R_{\vee}(H'_1)^2}{(2^{N_k} - 1)^2} \langle \vec{1} | D_{\bar{\alpha}}^T D_{\bar{\alpha}} | \vec{1} \rangle \\ &\quad + \frac{R_{\wedge} H'_2}{2^{N_k} - 1} \left[(\langle \vec{1} | D_{\bar{\alpha}} | \vec{1} \rangle)^2 - \frac{1}{N_k} \langle \vec{1} | D_{\bar{\alpha}} D_{\bar{\alpha}}^T | \vec{1} \rangle \right] + \frac{R_{\wedge} H_1}{N_k(2^{N_k} - 1)} \langle \vec{1} | D_{\bar{\alpha}} D_{\bar{\alpha}}^T | \vec{1} \rangle \\ &\quad + \frac{R_{|} H'_2}{2^{N_k} - 1} \left[\langle \vec{1} | D_{\bar{\alpha}}^T D_{\bar{\alpha}} | \vec{1} \rangle - \frac{1}{N_k} \text{Tr}(D_{\bar{\alpha}} D_{\bar{\alpha}}^T) \right] + \frac{R_{|} H_1}{N_k(2^{N_k} - 1)} \text{Tr}(D_{\bar{\alpha}} D_{\bar{\alpha}}^T), \end{aligned} \quad (80)$$

which after rearranging terms, becomes

$$\begin{aligned}
S_{\vec{\alpha}}^v &= \left(\langle \vec{1} | D_{\vec{\alpha}} | \vec{1} \rangle \right)^2 \left[\frac{R_{||}(H'_1)^2}{(2^{N_k} - 1)^2} + \frac{R_{\wedge} H'_2}{2^{N_k} - 1} \right] \\
&+ \langle \vec{1} | D_{\vec{\alpha}}^T D_{\vec{\alpha}} | \vec{1} \rangle \left[\frac{R_{\vee}(H'_1)^2}{(2^{N_k} - 1)^2} + \frac{R_{|} H'_2}{2^{N_k} - 1} \right] \\
&+ \langle \vec{1} | D_{\vec{\alpha}} D_{\vec{\alpha}}^T | \vec{1} \rangle \left[\frac{R_{\wedge}(H_1 - H'_2)}{N_k(2^{N_k} - 1)} \right] \\
&+ \text{Tr} (D_{\vec{\alpha}} D_{\vec{\alpha}}^T) \left[\frac{R_{|}(H_1 - H'_2)}{N_k(2^{N_k} - 1)} \right]. \tag{81}
\end{aligned}$$

Using this rearranged formula, we gain a clearer view of the contributions to $S_{\vec{\alpha}}$: There are four different terms whose prefactors depend solely on the ansatz (which can in turn depend on N_k), but usually not on N_i . The factors in square brackets are then different terms of the form $R_i \cdot f(N_k)$, where R_i depends strictly on N_i and the rest depends on N_k .

The calculation for $S_{\vec{\alpha}}^c$ defined in Eq. 68 is almost identical, except the second bracket in every term is flipped. Therefore, we can quickly write the final formula for this by performing the following changes:

$$\begin{aligned}
\left(\langle \vec{1} | D_{\vec{\alpha}} | \vec{1} \rangle \right)^2 &= \langle \vec{1} | D_{\vec{\alpha}} | \vec{1} \rangle \langle \vec{1} | D_{\vec{\alpha}} | \vec{1} \rangle \mapsto \langle \vec{1} | D_{\vec{\alpha}} | \vec{1} \rangle \langle \vec{1} | D_{\vec{\alpha}}^\dagger | \vec{1} \rangle = \left| \langle \vec{1} | D_{\vec{\alpha}} | \vec{1} \rangle \right|^2, \\
\langle \vec{1} | D_{\vec{\alpha}}^T D_{\vec{\alpha}} | \vec{1} \rangle &= \sum_a \langle e_a | D_{\vec{\alpha}} | \vec{1} \rangle \langle e_a | D_{\vec{\alpha}} | \vec{1} \rangle \mapsto \sum_a \langle e_a | D_{\vec{\alpha}} | \vec{1} \rangle \langle \vec{1} | D_{\vec{\alpha}}^\dagger | e_a \rangle = 1, \\
\langle \vec{1} | D_{\vec{\alpha}} D_{\vec{\alpha}}^T | \vec{1} \rangle &= \sum_a \langle \vec{1} | D_{\vec{\alpha}} | e_a \rangle \langle \vec{1} | D_{\vec{\alpha}} | e_a \rangle \mapsto \sum_a \langle \vec{1} | D_{\vec{\alpha}} | e_a \rangle \langle e_a | D_{\vec{\alpha}}^\dagger | \vec{1} \rangle = 1, \\
\text{Tr}(D_{\vec{\alpha}} D_{\vec{\alpha}}^T) &= \sum_{a,b} \langle e_a | D_{\vec{\alpha}} | e_b \rangle \langle e_a | D_{\vec{\alpha}}^\dagger | e_b \rangle \mapsto \sum_{a,b} \langle e_a | D_{\vec{\alpha}} | e_b \rangle \langle e_b | D_{\vec{\alpha}}^\dagger | e_a \rangle = N_k. \tag{82}
\end{aligned}$$

Using this map, the expression for $S_{\vec{\alpha}}^c$ reduces to:

$$\begin{aligned}
S_{\vec{\alpha}}^c &= \left| \langle \vec{1} | D_{\vec{\alpha}} | \vec{1} \rangle \right|^2 \left[\frac{R_{||}(H'_1)^2}{(2^{N_k} - 1)^2} + \frac{R_{\wedge} H'_2}{2^{N_k} - 1} \right] \\
&+ \left[\frac{R_{\vee}(H'_1)^2}{(2^{N_k} - 1)^2} + \frac{R_{\wedge}(H_1 - H'_2)}{N_k(2^{N_k} - 1)} + \frac{R_{|}(H_1)}{(2^{N_k} - 1)} \right]. \tag{83}
\end{aligned}$$

Here we note the following bounds:

$$\left| \left(\langle \vec{1} | D_{\vec{\alpha}} | \vec{1} \rangle \right)^2 \right| \leq \left| \langle \vec{1} | D_{\vec{\alpha}} | \vec{1} \rangle \right|^2 \leq 1, \tag{84}$$

$$\left| \langle \vec{1} | D_{\vec{\alpha}}^T D_{\vec{\alpha}} | \vec{1} \rangle \right| \leq 1, \tag{85}$$

$$\left| \langle \vec{1} | D_{\vec{\alpha}} D_{\vec{\alpha}}^T | \vec{1} \rangle \right| \leq 1, \tag{86}$$

$$\left| \text{Tr} (D_{\vec{\alpha}} D_{\vec{\alpha}}^T) \right| \leq N_k. \tag{87}$$

These bounds respectively match the lines in the map 82 we used above to derive $S_{\vec{\alpha}}^c$ from $S_{\vec{\alpha}}^v$. Therefore, by taking the modulus in Eq. 81, we can further obtain

$$\begin{aligned}
|S_{\vec{\alpha}}^v| &\leq \left| \left(\langle \vec{1} | D_{\vec{\alpha}} | \vec{1} \rangle \right)^2 \right| \left[\frac{R_{||}(H'_1)^2}{(2^{N_k} - 1)^2} + \frac{R_{\wedge} H'_2}{2^{N_k} - 1} \right] + \left| \langle \vec{1} | D_{\vec{\alpha}}^T D_{\vec{\alpha}} | \vec{1} \rangle \right| \left[\frac{R_{\vee}(H'_1)^2}{(2^{N_k} - 1)^2} + \frac{R_{|} H'_2}{2^{N_k} - 1} \right] \\
&+ \left| \langle \vec{1} | D_{\vec{\alpha}} D_{\vec{\alpha}}^T | \vec{1} \rangle \right| \left[\frac{R_{\wedge}(H_1 - H'_2)}{N_k(2^{N_k} - 1)} \right] + \left| \text{Tr} (D_{\vec{\alpha}} D_{\vec{\alpha}}^T) \right| \left[\frac{R_{|}(H_1 - H'_2)}{N_k(2^{N_k} - 1)} \right] \\
&\leq S_{\vec{\alpha}}^c. \tag{88}
\end{aligned}$$

This will become relevant for the final calculation of the asymptotic behavior of $\mathbb{E}[\text{Var}(\partial_\mu L_{QSGC})]$.

To this end, we can simplify the results in Eqs. 81 and 83 by approximating to leading order in N_k . For this, we use the results in Lemma 9:

$$\begin{aligned}
S_\alpha^v &= \left(\langle \vec{1} | D_\alpha | \vec{1} \rangle \right)^2 \left[\frac{R_{||}(H'_1)^2}{(2^{N_k} - 1)^2} + \frac{R_\wedge H'_2}{2^{N_k} - 1} \right] + \langle \vec{1} | D_\alpha^T D_\alpha | \vec{1} \rangle \left[\frac{R_\vee (H'_1)^2}{(2^{N_k} - 1)^2} + \frac{R_\downarrow H'_2}{2^{N_k} - 1} \right] \\
&\quad + \langle \vec{1} | D_\alpha D_\alpha^T | \vec{1} \rangle \left[\frac{R_\wedge (H_1 - H'_2)}{N_k (2^{N_k} - 1)} \right] + \text{Tr} (D_\alpha D_\alpha^T) \left[\frac{R_\downarrow (H_1 - H'_2)}{N_k (2^{N_k} - 1)} \right] \\
&= \left(\langle \vec{1} | D_\alpha | \vec{1} \rangle \right)^2 (R_{||} + R_\wedge) \left(\frac{1}{2^{N_k}} + O(N_k^{-2}) \right) + \langle \vec{1} | D_\alpha^T D_\alpha | \vec{1} \rangle (R_\vee + R_\downarrow) \left(\frac{1}{2^{N_k}} + O(N_k^{-2}) \right) \\
&\quad + \langle \vec{1} | D_\alpha D_\alpha^T | \vec{1} \rangle \frac{R_\wedge}{N_k} \left(\frac{1}{2^{N_k}} + O(N_k^{-2}) \right) + \text{Tr} (D_\alpha D_\alpha^T) R_\downarrow \left(\frac{1}{2^{N_k}} + O(N_k^{-3}) \right) \\
&= \left[\left(\langle \vec{1} | D_\alpha | \vec{1} \rangle \right)^2 (R_{||} + R_\wedge) + \langle \vec{1} | D_\alpha^T D_\alpha | \vec{1} \rangle (R_\vee + R_\downarrow) + \langle \vec{1} | D_\alpha D_\alpha^T | \vec{1} \rangle \frac{R_\wedge}{N_k} + \text{Tr} (D_\alpha D_\alpha^T) \frac{R_\downarrow}{N_k} \right] \\
&\quad \cdot \left(\frac{1}{2^{N_k}} + O(N_k^{-2}) \right). \tag{89}
\end{aligned}$$

Here, we used Lemma 9 to plug in the approximations for the coefficients of the R -terms, and then grouped them into a single one by adding a factor N_k^{-1} to the trace.

Similarly, for S_α^c , using the map given above in Eq. 82, we get

$$S_\alpha^c = \left[\left| \langle \vec{1} | D_\alpha | \vec{1} \rangle \right|^2 (R_{||} + R_\wedge) + 1 \cdot (R_\vee + \frac{R_\wedge}{N_k} + 2R_\downarrow) \right] \left(\frac{1}{2^{N_k}} + O(N_k^{-2}) \right). \tag{90}$$

Using the second inequality in 84, We know that in the asymptotic limit of N_k the first term in the square brackets above is $O(1)$, and that $R_\vee + 2R_\downarrow = \Theta(1)$, so we can infer

$$S_\alpha^c = \Theta(1) \left(\frac{1}{2^{N_k}} + O(N_k^{-2}) \right) = \Theta(N_k^{-1}). \tag{91}$$

For S_α^v , the bound cannot be tightened more without adding constraints to the ansatz.

Assembling the expected variance using Eq. 69:

$$\begin{aligned}
\mathbb{E}[\text{Var}(\partial_\mu L_{QSGC})] &= \frac{2}{2^D N_i^2 (k+1)^2} \sum_{\vec{\alpha} \in \{0,1\}^D} \left[(-1)^{w(\vec{\alpha})} \text{Re}(S_\alpha^v) + S_\alpha^c \right] \\
&= \frac{2}{2^D N_i^2 (k+1)^2} \left[\sum_{\vec{\alpha} \in \{0,1\}^D} (-1)^{w(\vec{\alpha})} \text{Re}(S_\alpha^v) + 2^D \Theta(N_k^{-1}) \right] \\
&= \frac{2}{2^D N_i^2 (k+1)^2} \sum_{\vec{\alpha} \in \{0,1\}^D} (-1)^{w(\vec{\alpha})} \text{Re}(S_\alpha^v) + \Theta(N_k^{-1}). \tag{92}
\end{aligned}$$

If the remaining summation in the RHS is negative, it could cancel out the leading term in $\Theta(N_k^{-1})$. However, one can employ a trick to constrain the ansatz slightly so that the sum is guaranteed to be positive. We prove this in Lemma 10. Therefore, the leading order of the variance is indeed $\Theta(N_k^{-1})$.

C.7 Bounds for the expected variance

Finally, we compute upper and lower bounds for the expected variance $\mathbb{E}[\text{Var}(\partial_\mu L_{QSGC})]$.

C.7.1 Upper bound

Given the bound 88, we can calculate an upper bound of the expected variance as such:

$$\begin{aligned}
\mathbb{E}[\text{Var}(\partial_\mu L_{QSGC})] &= \frac{2}{2^D N_i^2 (k+1)^2} \sum_{\vec{\alpha} \in \{0,1\}^D} \underbrace{\left[(-1)^{w(\vec{\alpha})} \text{Re}(S_{\vec{\alpha}}^v) + S_{\vec{\alpha}}^c \right]}_{\geq 0} \\
&= \frac{2}{2^D N_i^2 (k+1)^2} \sum_{\vec{\alpha} \in \{0,1\}^D} \left| (-1)^{w(\vec{\alpha})} \text{Re}(S_{\vec{\alpha}}^v) + \underbrace{S_{\vec{\alpha}}^c}_{\geq 0} \right| \\
&\leq \frac{2}{2^D N_i^2 (k+1)^2} \sum_{\vec{\alpha} \in \{0,1\}^D} \left[\underbrace{\left| (-1)^{w(\vec{\alpha})} \text{Re}(S_{\vec{\alpha}}^v) \right|}_{\leq S_{\vec{\alpha}}^c} + S_{\vec{\alpha}}^c \right] \\
&\leq \frac{4}{2^D N_i^2 (k+1)^2} \sum_{\vec{\alpha} \in \{0,1\}^D} S_{\vec{\alpha}}^c.
\end{aligned} \tag{93}$$

Next, we can apply bound 84 on $S_{\vec{\alpha}}^c$ to

$$S_{\vec{\alpha}}^c \leq \left[(R_{||} + R_{\wedge}) + (R_{\vee} + \frac{R_{\wedge}}{N_k} + 2R_{|}) \right] \left(\frac{1}{2N_k} + O(N_k^{-2}) \right) \equiv S. \tag{94}$$

After neglecting the $R_{\wedge}/N_k$ term (that contributes as $O(N_k^{-2})$ and plugging in the values from Table 1, we get

$$S = [N_i^2 (k+1)^2 + N_i (k+1)] \left(\frac{1}{2N_k} + O(N_k^{-2}) \right). \tag{95}$$

Therefore, the expected variance is upper bounded by

$$\begin{aligned}
\mathbb{E}[\text{Var}(\partial_\mu L_{QSGC})] &\leq \frac{4}{2^D N_i^2 (k+1)^2} 2^D S \\
&= \left[1 + \frac{1}{N_i (k+1)} \right] \left(\frac{2}{N_k} + O(N_k^{-2}) \right) = \Theta(N_k^{-1}) \equiv \Theta(C^{-1}).
\end{aligned} \tag{96}$$

Finding whether this bound is tight is a work in progress, as one needs to find the right ansatz that maximizes the expected variance.

C.7.2 Lower bound

Using Lemma 10, we can always force the sum over $\text{Re}(S_{\vec{\alpha}}^v)$ to be nonnegative. Therefore, given the additional constraint in the lemma, a lower bound for the expected variance is the following:

$$\begin{aligned}
\mathbb{E}[\text{Var}(\partial_\mu L_{QSGC})] &= \frac{2}{2^D N_i^2 (k+1)^2} \left[\underbrace{\sum_{\vec{\alpha} \in \{0,1\}^D} (-1)^{w(\vec{\alpha})} \text{Re}(S_{\vec{\alpha}}^v)}_{\geq 0} + \sum_{\vec{\alpha} \in \{0,1\}^D} S_{\vec{\alpha}}^c \right] \\
&\geq \frac{2}{2^D N_i^2 (k+1)^2} \sum_{\vec{\alpha} \in \{0,1\}^D} S_{\vec{\alpha}}^c.
\end{aligned} \tag{97}$$

To give a lower bound for $S_{\vec{\alpha}}^c$, the term $\left| \langle \vec{1} | D_{\vec{\alpha}} | \vec{1} \rangle \right|^2$ can be taken to 0:

$$\begin{aligned}
S_{\vec{\alpha}}^c &= \left[\underbrace{\left| \langle \vec{1} | D_{\vec{\alpha}} | \vec{1} \rangle \right|^2}_{\geq 0} (R_{||} + R_{\wedge}) + 1 \cdot (R_{\vee} + \underbrace{\frac{R_{\wedge}}{N_k}}_{=O(N_k^{-1})} + 2R_{|}) \right] \left(\frac{1}{2N_k} + O(N_k^{-2}) \right) \\
&\geq (R_{\vee} + 2R_{|}) \left(\frac{1}{2N_k} + O(N_k^{-2}) \right) \\
&= [N_i (k+1)^2 + N_i (k+1)] \left(\frac{1}{2N_k} + O(N_k^{-2}) \right),
\end{aligned} \tag{98}$$

which means the expected variance is ultimately lower-bounded by

$$\mathbb{E}[\text{Var}(\partial_\mu L_{QSGC})] \geq \left[\frac{1}{N_i} + \frac{1}{N_i(k+1)} \right] \left(\frac{1}{N_k} + O(N_k^{-2}) \right) = \Theta(N_i^{-1} N_k^{-1}) \equiv \Theta(N^{-1} C^{-1}). \quad (99)$$

We emphasize on the fact that this bound can only be attained if the term $\left| \langle \vec{1} | D_{\vec{\alpha}} | \vec{1} \rangle \right|^2 = 0$, which is a significant constraint to the ansatz.

C.8 Note on the curse of dimensionality

The variance bounds given above are computed by considering the strict bounds on the leading term, namely,

$$0 \leq \left| \langle \vec{1} | D_{\vec{\alpha}} | \vec{1} \rangle \right|^2 \leq 1.$$

We note here that in reality, for most VQCs¹⁴, attaining either of these ends is highly unlikely. However, because of the vanishing overlap between two arbitrary states (the **curse of dimensionality**)[45], the distribution of possible moduli $\left| \langle \vec{1} | D_{\vec{\alpha}} | \vec{1} \rangle \right|^2$ strongly peaks around N_k^{-1} . We prove this with Lemma 11, in Appendix G.

Therefore, most (arbitrarily chosen) ansätze lead to

$$\begin{aligned} S_{\vec{\alpha}}^c &= \underbrace{\left[\left| \langle \vec{1} | D_{\vec{\alpha}} | \vec{1} \rangle \right|^2 \right]}_{\sim N_k^{-1}} (R_{||} + R_{\wedge}) + R_{\vee} + \frac{R_{\wedge}}{N_k} + 2R_{|} \left(\frac{1}{2N_k} + O(N_k^{-2}) \right) \\ &\sim \left[R_{\vee} + 2R_{|} + \frac{R_{||} + 2R_{\wedge}}{N_k} \right] \left(\frac{1}{2N_k} + O(N_k^{-2}) \right) \\ &= \Theta(N_i k^2 N_k^{-1}) + O(N_i^2 k^2 N_k^{-2}). \end{aligned} \quad (100)$$

Plugging this in Eq. 69, we see that apart from the $S_{\vec{\alpha}}^v$ alternating sum (which as seen above, can be bounded between 0 and the $S_{\vec{\alpha}}^c$ sum), most ansätze will have an additional $O(N_k^{-2})$ term¹⁵:

$$\begin{aligned} \mathbb{E}[\text{Var}(\partial_\mu L_{QSGC})] &\sim \Theta(N_i^{-1} N_k^{-1}) + \Theta(N_k^{-2}) \\ &= \Theta(N^{-1} C^{-1}) + \Theta(C^{-2}). \end{aligned} \quad (101)$$

This implies on the one hand that most ansätze encounter the flattening with graph size, but at a much harsher tradeoff with the N_k scaling.

C.9 Example: IQP ansatz

In this section we show the application of the results above on a worked example: the IQP ansatz. This takes the form:

$$U(\theta) = H^{\otimes n_k} \exp \left(\sum_{j=1}^{n_k} i\theta_j Z_j + \sum_{j < k} i\theta_{jk} Z_j Z_k \right) H^{\otimes n_k}. \quad (102)$$

In terms of our general ansatz formula, each θ_j and θ_{jk} is a parameter, so the depth is

$$D = n_k + \frac{n_k(n_k - 1)}{2} = \frac{n_k(n_k + 1)}{2}. \quad (103)$$

The W_η gates are:

$$W_\eta = \begin{cases} \mathbb{1} & \text{for } \eta \in \{1, D+1\}, \\ H^{\otimes n_k} & \text{otherwise,} \end{cases} \quad (104)$$

¹⁴By this we mean that the set of generating gates V_η and the fixed gates W_η are arbitrary.

¹⁵By expanding the H terms in Lemma 9 to next order, one can check that the leading term is present, so we upgrade the Big- O notation to a stricter $\Theta(N_k^{-2})$.

and the V_η are:

$$V_\eta = \begin{cases} Z_j, & \text{for } 1 \leq \eta = j \leq n_k, \\ Z_j Z_k, & \text{for } n_k \leq \eta = j \cdot n_k + k \leq D. \end{cases} \quad (105)$$

Then, when computing the expected variance we need $D_{\vec{\alpha}}$, which takes the following form:

$$D_{\vec{\alpha}} = H^{\otimes n_k} \cdot \prod_{\eta} V_{\eta}^{\alpha_{\eta}} \cdot H^{\otimes n_k}, \quad (106)$$

where the product ends up being a tensor product of identities and Z gates. Note that all gates in $D_{\vec{\alpha}}$ are real, square to identity, and the Z gates commute, so $D_{\vec{\alpha}} = D_{\vec{\alpha}}^T = D_{\vec{\alpha}}^\dagger$. Therefore, the following simplifications occur:

$$\langle \vec{1} | D_{\vec{\alpha}} | \vec{1} \rangle = \langle \vec{1} | H^{\otimes n_k} \cdot \prod_{\eta} V_{\eta}^{\alpha_{\eta}} \cdot H^{\otimes n_k} | \vec{1} \rangle = \langle 0 | \prod_{\eta} V_{\eta}^{\alpha_{\eta}} | 0 \rangle = 1 \quad (107)$$

$$\langle \vec{1} | D_{\vec{\alpha}}^T D_{\vec{\alpha}} | \vec{1} \rangle = \langle \vec{1} | D_{\vec{\alpha}} D_{\vec{\alpha}}^T | \vec{1} \rangle = \langle \vec{1} | \vec{1} \rangle = 1 \quad (108)$$

$$\text{Tr}(D_{\vec{\alpha}} D_{\vec{\alpha}}^T) = \text{Tr}(\mathbb{1}) = N_k. \quad (109)$$

Hence one can observe that the bounds from Eqs. 84-87 are reached. To this end, the bound in Eq. 88 is also attained,

$$S_{\vec{\alpha}}^v = S_{\vec{\alpha}}^c = \left[(R_{||} + R_{\wedge}) + (R_{\vee} + \frac{R_{\wedge}}{N_k} + 2R_{|}) \right] \left(\frac{1}{2N_k} + O(N_k^{-2}) \right) = S. \quad (110)$$

Using Eq. 95, the full expected variance is

$$\begin{aligned} \mathbb{E}[\text{Var}(\partial_\mu)] &= \frac{2}{2^D N_i^2 (k+1)^2} \sum_{\vec{\alpha} \in \{0,1\}^D} \left[(-1)^{w(\vec{\alpha})} \text{Re}(S_{\vec{\alpha}}^v) + S_{\vec{\alpha}}^c \right] \\ &= \frac{2}{2^D N_i^2 (k+1)^2} S \underbrace{\sum_{\vec{\alpha} \in \{0,1\}^D} \left[(-1)^{w(\vec{\alpha})} + 1 \right]}_{2^D} \\ &= \left[1 + \frac{1}{N_i(k+1)} \right] \left(\frac{1}{N_k} + O(N_k^{-2}) \right). \end{aligned} \quad (111)$$

Therefore, to leading order, this expected variance does flatten in the limit of large N_i and k , and decays as N_k^{-1} .

We note that whilst $S_{\vec{\alpha}}^v$ and $S_{\vec{\alpha}}^c$ themselves have been maximised using this ansatz, the summation over $\vec{\alpha}$ has not. This is because the $S_{\vec{\alpha}}^v$ contribution cancels out. In fact, this expected variance is exactly half the maximum possible. This also means that this ansatz has an optimal asymptotic form, up to a factor of 2.

D Generalizing the trainability results across different QGCNs

In this section, we show that by maintaining the two assumptions from Sections C.4 and C.5, namely the k -regularity and averaging across different inputs and targets, we can easily generalize our one-layer QSGC result to any QSGC or even QLGC architecture. For that, we first need to make a stronger generalization of the results above to *graphs with weighted edges*.

D.1 Graphs with weighted edges

By generalizing the architecture to weighted graphs, we allow the matrix $\hat{A}_{ij}$ to contain custom edge weights by inheriting them from the unnormalized adjacency matrix $\tilde{A}_{ij}$; see Appendix A.2 for details.

Therefore the results on the bias and variance of the cost function gradient stay the same for arbitrary choices of the matrix $\hat{A}$. We recall the formulas:

$$\mathbb{E}_{\boldsymbol{\theta}}[\partial_{\mu} L_1] = \frac{1}{(2\pi)^D N_i} \sum_{i,j=1}^{N_i} \hat{A}_{ij} \int d\boldsymbol{\theta}' \langle \mathbf{y}_j | U(\boldsymbol{\theta}', \theta_{\mu} = \pi) - U(\boldsymbol{\theta}', \theta_{\mu} = -\pi) | \mathbf{x}_i \rangle = 0; \quad (112)$$

$$\text{Var}_{\boldsymbol{\theta}}[\partial_{\mu} L_1] = \frac{1}{2^D N_i^2} \sum_{i,j,k,l=1}^{N_i} \hat{A}_{ij} \hat{A}_{kl} \sum_{\vec{\alpha} \in \{0,1\}^D} (-1)^{w(\vec{\alpha})} \langle \mathbf{y}_j | D_{\vec{\alpha}} | \mathbf{x}_i \rangle \langle \mathbf{y}_l | D_{\vec{\alpha}} | \mathbf{x}_k \rangle, \quad (113)$$

$$\text{Cov}_{\boldsymbol{\theta}}[\partial_{\mu} L_1, \partial_{\mu} L_1^*] = \frac{1}{2^D N_i^2} \sum_{i,j,k,l=1}^{N_i} \hat{A}_{ij} \hat{A}_{kl} \sum_{\vec{\alpha} \in \{0,1\}^D} \langle \mathbf{y}_j | D_{\vec{\alpha}} | \mathbf{x}_i \rangle \langle \mathbf{x}_k | D_{\vec{\alpha}}^{\dagger} | \mathbf{y}_l \rangle. \quad (114)$$

We see that for any choice of $\hat{A}$, the bias is zero, thus maintaining the results for the variance and covariance terms. For these terms, we first take the expected value over the input and target *before* making the regular graph assumption:

$$\mathbb{E}_{i,t}[\text{Var}[\partial_{\mu} L_1]] = \frac{1}{2^D N_i^2} \sum_{i,j,k,l=1}^{N_i} \hat{A}_{ij} \hat{A}_{kl} \sum_{\vec{\alpha} \in \{0,1\}^D} (-1)^{w(\vec{\alpha})} \mathbb{E}_{i,t}[\langle \mathbf{y}_j | D_{\vec{\alpha}} | \mathbf{x}_i \rangle \langle \mathbf{y}_l | D_{\vec{\alpha}} | \mathbf{x}_k \rangle],$$

$$\mathbb{E}_{i,t}[\text{Cov}[\partial_{\mu} L_1, \partial_{\mu} L_1^*]] = \frac{1}{2^D N_i^2} \sum_{i,j,k,l=1}^{N_i} \hat{A}_{ij} \hat{A}_{kl} \sum_{\vec{\alpha} \in \{0,1\}^D} \mathbb{E}_{i,t}[\langle \mathbf{y}_j | D_{\vec{\alpha}} | \mathbf{x}_i \rangle \langle \mathbf{x}_k | D_{\vec{\alpha}}^{\dagger} | \mathbf{y}_l \rangle].$$

Here, we observe the same splitting of the sum as in Appendix C.5, depending on whether $i = k$ and/or $j = l$. The R values calculated above can thus be generalized as:

$$\hat{R}_{||} := \sum_{\substack{i \neq k \\ j \neq l}} \hat{A}_{ij} \hat{A}_{kl} = \sum_{i,j,k,l} \hat{A}_{ij} \hat{A}_{kl} - \hat{R}_{\wedge} - \hat{R}_{\vee} - \hat{R}_{|}, \quad (115)$$

$$\hat{R}_{\wedge} := \sum_{i,j \neq l} \hat{A}_{ij} \hat{A}_{il} = \sum_{i,j,l} \hat{A}_{ij} \hat{A}_{il} - \hat{R}_{|}, \quad (116)$$

$$\hat{R}_{\vee} := \sum_{i \neq k,j} \hat{A}_{ij} \hat{A}_{kj} = \sum_{i,k,l} \hat{A}_{ij} \hat{A}_{kj} - \hat{R}_{|}, \quad (117)$$

$$\hat{R}_{|} := \sum_{i,j} \hat{A}_{ij} \hat{A}_{ij} = \text{Tr}(\hat{A}^2). \quad (118)$$

In the regular graph case (Appendices C.4 and C.5) the $\hat{R}$ notation is simply the R notation divided by $(k+1)^2$.

Weighted regular graphs. For the case when the graph is weighted k -regular, meaning that $\tilde{D}_{ii} = (k+1)\mathbb{1}$, we use Lemma 12 to bound the $\hat{R}$ values above. In terms of orders, the $\hat{R}$ terms maintain the orders from the unweighted graph case, although they depend heavily on $\hat{R}_{|}$:

$$\hat{R}_{||} = N(N-2) + \hat{R}_{|} = \Theta(N^2), \quad (119)$$

$$\hat{R}_{\wedge} = \hat{R}_{\vee} = N - \hat{R}_{|} = O(N), \quad (120)$$

$$\hat{R}_{|} = O(N). \quad (121)$$

Therefore, we can guarantee that to leading order, the trainability results from Appendix C hold for the case of weighted graphs.

Upper bound. If $\left|\langle \vec{1} | D_{\vec{\alpha}} | \vec{1} \rangle\right|^2 = 1$, then repeating the calculation in Appendix C.7.1 leads to $S_{\vec{\alpha}}^c \leq S$ where

$$\begin{aligned} S &= \left[(R_{||} + R_{\wedge}) + (R_{\vee} + \frac{R_{\wedge}}{N_k} + 2R_{|}) \right] \left(\frac{1}{2N_k} + O(N_k^{-2}) \right) \\ &= (k+1)^2 \left[\left(\sum_{i,j} \hat{A}_{ij} \right)^2 + \hat{R}_{|} + \frac{1}{N_k} \left(\sum_{i,j,k} \hat{A}_{ij} \hat{A}_{kj} - \hat{R}_{|} \right) \right] \left(\frac{1}{2N_k} + O(N_k^{-2}) \right) \\ &= \Theta(N^2 k^2 N_k^{-1}) \quad \left(= \Theta(N_i^2 k^2 N_k^{-1}) \right). \end{aligned} \quad (122)$$

As derived for the unweighted case, this leads to

$$\mathbb{E}[\text{Var}(\partial_{\mu} L_{QSGC})]_{\text{upper}} = \Theta(C^{-1}). \quad (123)$$

Lower bound. If $\left|\langle \vec{1} | D_{\vec{\alpha}} | \vec{1} \rangle\right|^2 = 1$, then repeating the calculation in Appendix C.7.2 leads to

$$\begin{aligned} S_{\vec{\alpha}}^c &\geq (R_{\vee} + 2R_{|}) \left(\frac{1}{2N_k} + O(N_k^{-2}) \right) \\ &= (k+1)^2 \left(\sum_{i,j,k} \hat{A}_{ij} \hat{A}_{kj} + \hat{R}_{|} \right) \left(\frac{1}{2N_k} + O(N_k^{-2}) \right) \\ &= \Theta(N k^2 N_k^{-1}), \end{aligned} \quad (124)$$

which leads to the same lower bound

$$\mathbb{E}[\text{Var}(\partial_{\mu} L_{QSGC})]_{\text{lower}} = \Theta(N^{-1} C^{-1}). \quad (125)$$

Expected dependence. To this end, by adding the factor given by the curse of dimensionality, $\left|\langle \vec{1} | D_{\vec{\alpha}} | \vec{1} \rangle\right|^2 \sim N_k^{-1}$, we obtain the same result as in Appendix C.8,

$$\begin{aligned} \mathbb{E}[\text{Var}(\partial_{\mu} L_{QSGC})]_{\text{expected}} &\sim \mathbb{E}[\text{Var}(\partial_{\mu} L_{QSGC})]_{\text{upper}} / N_k + \mathbb{E}[\text{Var}(\partial_{\mu} L_{QSGC})]_{\text{lower}} \\ &= \Theta(C^{-2} + N^{-1} C^{-1}). \end{aligned} \quad (126)$$

D.2 Results for QSGCs with K layers

The results above provide a significant extension to the one-layer QSGC. Taking two layers for the QSGC, for example, is equivalent to updating all of our results with $\hat{A} \mapsto \hat{A}^2$. However, as proven in Proposition 3, multiplying two commuting normalized adjacency matrices with self-loops gives a new normalized adjacency matrix with self-loops. Therefore, because $\hat{A}$ commutes with itself, then $\hat{A}^2$ is a valid normalized adjacency matrix with self-loops of a new graph.¹⁶ Thus, our results should hold by reassigning $\hat{A}^2 \equiv \hat{B}$ for some new graph $\mathcal{G}_B$. Moreover, thanks to Corrolary 4, if the initial graph is k -regular, then graph $\mathcal{G}_B$ is also k -regular.

This can be continued. As $\hat{A}$ and $\hat{B} = \hat{A}^2$ commute, multiplying them gives a new graph with normalized adjacency matrix with self-loops $\hat{C} = \hat{A}^3$, which is also k -regular. Continuing this process, we confirm that our results hold for any QSGC model with an arbitrary number of layers, K . Explicitly, for a general, K -layer QSGC, the upper bound in Eq. 123, lower bound in Eq. 125, and expected dependence for arbitrary ansätze in Eq. 126 hold.

¹⁶This works regardless of whether $\hat{A}$ has weighted edges or not. In the unweighted case, we just define trivial weights $w_{ij} = 1$ on all edges of the graphs.

D.3 Results for QLGCs

As mentioned in Section B.1, here we redefine *Quantum Linear Graph Convolutions* with a polynomial of $\hat{A}$ rather than the graph laplacian, as it is easier to implement. In our variance calculations, unsurprisingly, we only have to replace

$$\hat{A} \mapsto P(\hat{A}) = \sum_{p=1}^K a_p \hat{A}^p. \quad (127)$$

Here, in order to implement the polynomial on the address register $Reg(i)$, its spectral norm must be subunitary, and so we choose the range of coefficients $a_p \in [0, 1]$, with at least one being non-zero and their sum must be one, $\sum_p a_p = 1$. This is essentially a linear superposition of p -layer QSGC models.

Employing Corollary 6, the convex sum $P(\hat{A})$ of commuting matrices $\hat{A}^p$ is a valid normalized adjacency matrix with self-loops of some new graph. This will reduce most of the QLGC trainability calculation to the one-layer QSGC, as seen above. However, here we note that the coefficients a_p are also trainable parameters.

D.3.1 New gradient variance calculation

Following the result in Eq. 34, the new cost function is given by

$$\begin{aligned} L_{QLGC} &= -\frac{1}{N_i} \sum_{i,j} P(\hat{A})_{ij} \text{Re}(\langle \mathbf{y}_j | U(\boldsymbol{\theta}) | \mathbf{x}_i \rangle) \\ &= \sum_p a_p \left(-\frac{1}{N_i} \sum_{i,j} (\hat{A}^p)_{ij} \text{Re}(\langle \mathbf{y}_j | U(\boldsymbol{\theta}) | \mathbf{x}_i \rangle) \right) \\ &\equiv \sum_p a_p L_p. \end{aligned} \quad (128)$$

Here we note L_p the cost function of a p -layer QSGC.

The new set of parameters is $\theta_\eta \in [-\pi, \pi)$, along with $a_p \in [0, 1]$, with the additional condition $\sum_p a_p = 1$. We already know that the volume for the θ_η parameters is $(2\pi)^D$, where D is the number of weights in the VQC. Let the volume $\int d\vec{a} \equiv \mathcal{N}_a$.¹⁷

Therefore, there are two gradients that need to be analyzed: $\partial_\mu L_{QLGC}$ and $\partial_l L_{QLGC} \equiv \partial L_{QLGC} / \partial a_l$.

Gradient $\partial_\mu L_{QLGC}$. The cost gradient bias in this case is

$$\mathbb{E}_{\vec{a}} \mathbb{E}_{\boldsymbol{\theta}} [\partial_\mu L_{QLGC}] = \mathbb{E}_{\vec{a}} \sum_p a_p \mathbb{E}_{\boldsymbol{\theta}} [\partial_\mu L_p]. \quad (129)$$

The terms in the sum are simply the biases of the QSGC models, which are zero. Therefore, the overall bias is also zero.

The variance is

$$\text{Var}_{\vec{a}, \boldsymbol{\theta}} [\partial_\mu L_{QLGC}] = \mathbb{E}_{\vec{a}} \mathbb{E}_{\boldsymbol{\theta}} [(\partial_\mu L_{QLGC})^2] = \mathbb{E}_{\vec{a}} \sum_{p,p'} a_p a_{p'} \text{Cov}_{\boldsymbol{\theta}} (\partial_\mu L_p, \partial_\mu L_{p'}). \quad (130)$$

We see that the terms

$$\text{Cov}_{\boldsymbol{\theta}} (\partial_\mu L_p, \partial_\mu L_{p'}) = -\frac{1}{N_i^2} \sum_{i,j,k,l} (\hat{A}^p)_{ij} (\hat{A}^{p'})_{kl} \text{Re}(\langle \mathbf{y}_j | U(\boldsymbol{\theta}) | \mathbf{x}_i \rangle) \text{Re}(\langle \mathbf{y}_l | U(\boldsymbol{\theta}) | \mathbf{x}_k \rangle) \quad (131)$$

are a slightly more generalized version of the variance in Eq. 113, where instead of counting terms $\hat{A}_{ij} \hat{A}_{kl}$, we use terms $\hat{B}_{ij} \hat{C}_{kl}$, where the commuting normalized adjacency matrices with self-loops $\hat{B} = \hat{A}^p$ and $\hat{C} = \hat{A}^{p'}$. By adapting Eqs. 115, 116, 117, and 118 with this change, we see that, in fact, the k -regular approximations

¹⁷This is the hypersurface section of constant sum $\sum_p a_p = 1$ through the unit hypercube, which is $\mathcal{N}_a = \sqrt{K}/(K-1)!$, but for our purposes the exact value will not be relevant.

are exactly the same. By rerunning the same set of arguments, unsurprisingly we can check that the terms $\text{Cov}_{\boldsymbol{\theta}}(\partial_{\mu}L_p, \partial_{\mu}L_{p'})$ have the same dependence as any QSGC, which is solely on the ansatz and the system size. We call this dependence $\mathcal{D}(N, C)$ for simplicity.

Therefore, the overall variance simplifies to

$$\text{Var}_{\vec{a};\boldsymbol{\theta}}[\partial_{\mu}L_{QLGC}] = \mathbb{E}_{\vec{a}} \sum_{p,p'} a_p a_{p'} \mathcal{D}(N, C) = \mathbb{E}_{\vec{a}} \underbrace{\left(\sum_p a_p \right)^2}_1 \mathcal{D}(N, C) = \mathcal{D}(N, C). \quad (132)$$

Gradient $\partial_l L_{QLGC}$. The gradient in this case takes the form

$$\partial_l L_{QLGC} = \partial_l \left(\sum_p a_p L_p \right) = L_l. \quad (133)$$

This does not depend on any a_p and so the bias and variance simply reduce to those of the p -layer QSGC cost.

$$\mathbb{E}_{\vec{a}} \mathbb{E}_{\boldsymbol{\theta}}[\partial_l L_{QLGC}] = \mathbb{E}_{\boldsymbol{\theta}}[L_p], \quad (134)$$

$$\text{Var}_{\vec{a};\boldsymbol{\theta}}[\partial_l L_{QLGC}] = \text{Var}_{\boldsymbol{\theta}}[L_p]. \quad (135)$$

How different is calculating the bias and variance of the QSGC cost function instead of its gradient? Because of our choice of sequential ansatz $U(\boldsymbol{\theta}) = \prod_{\eta} (\exp(-i\theta_{\eta} V_{\eta}) \cdot W_{\eta})$, it turns out the results are exactly the same. This is thanks to the *parameter-shift rule*: because $V_{\eta}^2 = \mathbb{1}$, taking the derivative

$$\partial_{\mu} \exp(-i\theta_{\mu} V_{\mu}) = -iV_{\mu} \exp(-i\theta_{\mu} V_{\mu}) = \exp(-i(\theta_{\mu} + \pi/2)V_{\mu}). \quad (136)$$

Therefore, by relabeling $\boldsymbol{\theta} \mapsto \boldsymbol{\theta}'$ where all components are the same except $\theta_{\mu} \mapsto \theta_{\mu} + \pi/2$, the overall gradient $\partial_{\mu} U(\boldsymbol{\theta}) \equiv U(\boldsymbol{\theta}')$. In all of the integrals from the variance derivation, the exact endpoints of the angle domains were irrelevant; it only matters that they integrate over a full sine period. Hence, shifting $\theta_{\mu} \mapsto \theta_{\mu} + \pi/2$ does not affect the results.

To this end, the bias of $\partial_l L_{QLGC}$ is also zero, and the variance is $\mathcal{D}(N, C)$. We conclude that all QSGCs and QLGCs are equally trainable, leading to the exact same gradient cost variance for a given system size and ansatz.

E Experiments

The results on the cost gradient variance provided are part of a more extensive set of experiments ran to test the implementation and benchmarking of the QGNN model, covering architectures beyond the QSGC one described here.

E.1 Experimental setup

Two types of quantum models—*Quantum Simplified Graph Convolutional (QSGC) network* and *Quantum Linear Graph Convolutional (QLGC) network*—are considered, together with their respective classical counterparts, *CSGC* and *CLGC*. Quantum models are implemented using *PennyLane* [46] and run in a noiseless simulator, employing *JAX* [47] to accelerate the underlying linear algebra.

Table 2: Summary of the datasets employed in the numerical experiments. *Mask* represents the number of labeled nodes considered for the training of models.

Name	Nodes	Features	Classes	Mask	Reference
Karate Club	34	34	4	4	[18]
Wisconsin	251	1,703	5	120	[19]

All models are trained for the datasets listed in Tab. 2. In the case of the graph *Karate Club*, no features are provided originally for nodes. For this reason, nodes are one-hot-encoded and the data matrix $X = I_N$.

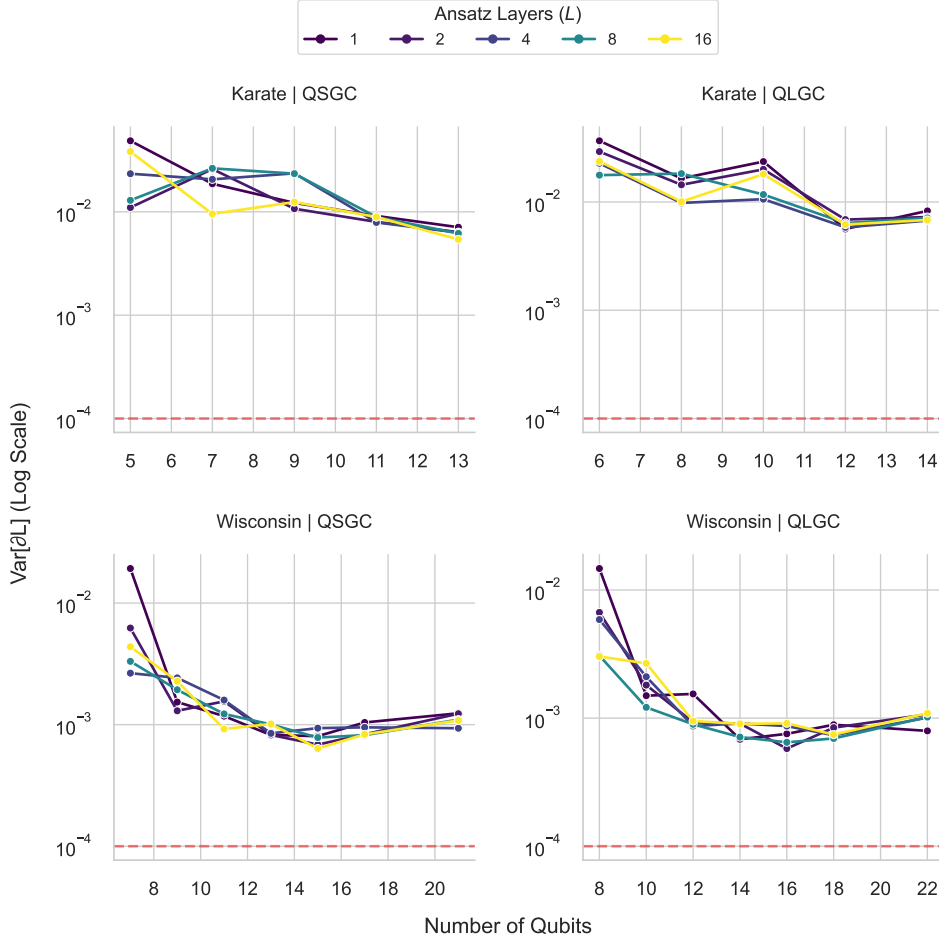


Figure 3: Variance of the cost function gradient $\text{Var}(\partial L / \partial \theta_\mu)$ (log scale) as a function of the number of qubits for the Karate and Wisconsin datasets. For the VQC we use the Strongly Entangled Ansatz [48]. The dashed red line represents the precision limit for 10,000 shots, where the signal-to-noise ratio becomes insufficient for gradient-based optimization. The flattening variance of different layers depths for larger qubit counts suggest that circuit expressivity does not incur the scaling limits typically associated with random-like quantum landscapes, maintaining trainability. Larger version of Fig. 1, including the QLGC results.

In those cases, the amplitude-encoded quantum state can be efficiently prepared in $\mathcal{O}(\log_2 N)$ depth and space [43].

As variational ansatz in the quantum models, the widely spread *Strongly Entangling Layers* [48] architecture is employed, adding 3 optimizable parameters per qubit and per layer. Those variational parameters are optimized using the classical optimizer *Adam* [49] using the same learning rate for all models, but ensuring their convergence into a minimum. Additionally, this choice of ansatz is very expressive, which usually is linked to low trainability, thus we consider this a good test of the model’s gradient analysis.

E.2 Numerical Results

The results have been obtained for a number of layers of the ansatz applied and for varying graph size and connectivity. For this, various subgraphs of *Karate* and *Wisconsin* datasets are randomly sampled, in order to encode them in different qubit-sized circuits. For each number of qubits and layers in the quantum circuit, 1000 random initializations of parameters are made, for which the gradient of the loss function is computed. Fig. 3 shows the scaling of the variance of those gradients depending on the number of qubits in the quantum circuit. These results, particularly for the *Wisconsin* dataset, back the possibility of flattening of the cost gradient variance with graph size, as shown in the theoretical analysis in Appendix C. Moreover, the addition of a varying number of layers increases the expressivity, whilst maintaining the trainability, as

seen in the figure above.

F Complexity and dequantization expanded

In this section, we show how one can benefit from the space-time complexity tradeoff in the case of the QSGC, and how it can be dequantized for graphs giving rise to low rank adjacency matrices $\hat{A}$.

F.1 Complexity benefits and caveats

We begin this by importing the complexity results from the QSGC's original paper [13]. For the classical SGC (CSGC) model, the general complexity bounds are

$$S_{CSGC} = O(|E| + NC + C^2), \quad (137)$$

$$T_{CSGC} = O((|E|C + NC^2) \log(1/\varepsilon')), \quad (138)$$

where $|E| = Nk/2$ is the number of edges, N that of nodes, k the average degree, and C the number of features per node. ε' is a precision parameter and is considered fixed, so we will absorb it in the Big- O notation as an overhead. We mention that the logarithmic dependence makes this overhead have a small impact to the practical complexity.

The QSGC model attains the following complexities in the number of qubits and circuit depth:

$$n_{QSGC} = O(\log(NC) + n_{anc} + n'_{anc}), \quad (139)$$

$$D_{QSGC} = \tilde{O}\left(NC \log(1/\varepsilon_1) \frac{\log(n_{anc})}{n_{anc}} + N \log(N) \cdot s \log(s) \log(1/\varepsilon_2) \frac{\log(n'_{anc})}{n'_{anc}}\right). \quad (140)$$

Here, n_{anc} and ε_1 are respectively the required number of ancillary qubits and precision of the amplitude encoding of the input and target states. Similarly, n'_{anc} and ε_2 are those required for the block encoding of the adjacency matrix $\hat{A}$. The algorithm implementing the sparse block encoding assumes that $\hat{A}$ has sparsity s . This is to say, every column in $\hat{A}$ has at most s non-zero values. Because this includes the self-loops, that is equivalent to saying $s - 1$ corresponds to the maximum degree in the graph. The Big- $\tilde{O}$ notation excludes the doubly logarithmic factors as they scale very slowly. We note that the numbers of ancillary qubits n_{anc} and n'_{anc} can be adjusted, and lie in the following ranges:

$$\Omega(\log(NC)) \leq n_{anc} \leq O(NC), \quad (141)$$

$$\Omega(\log(N)) \leq n'_{anc} \leq O(N \log(N) \cdot s \log(s)). \quad (142)$$

The lower bound of n_{anc} indicates that the logarithmic term in the total qubit number n_{QSGC} can be omitted. We use this in Eq. 144.

Gradient calculation. Firstly, the *parameter-shift rule* [50, 51] states that in order to compute the cost function gradient for a gradient descent method, one needs to shift in turn each of the weights θ_η of the ansatz $U(\boldsymbol{\theta})$. This adds a contribution of $O(2D)$ queries, where D is the number of weights.

Shot noise. As mentioned in [13], is of order $\Omega(\log(1/\delta)/\varepsilon^2)$. Here, ε is the additive error to the gradient, and the probability of failure δ is fixed. In order to resolve points on the loss landscape so that the gradient descent can be performed, this error needs to lie below the standard deviation $\sigma_{\partial_\mu L} \sim \text{Var}_{\boldsymbol{\theta}}[\partial_\mu L_{QSGC}]$. If we choose an arbitrary ansatz, the expected variance is of order $\Theta(N^{-1}C^{-1} + C^{-2})$. Then, the number of shots must be at least

$$\#\text{shots} = \Omega\left(\frac{1}{\varepsilon^2}\right) \geq \Omega\left(\frac{1}{\sigma^2}\right) = \Omega\left(\frac{1}{N^{-1}C^{-1} + C^{-2}}\right) = \Omega\left(\frac{C^2}{1 + C/N}\right) = \Omega(C^2), \quad (143)$$

where $C/N \ll 1$ by choice.

Input problem. As mentioned in Section 4, in order to get the full space and time complexities, one needs to add to the general circuit complexities the full classical compilation overhead. For the QSGC, this requires that all classical data of the input, target, and adjacency matrix $\hat{A}$ to be stored and processed classically in order to generate the transpiled circuit. This adds a term $O(|E|C + NC)$ to the space complexity and $\Omega(|E|C + NC)$ to the time complexity:

$$S_{QSGC} = O(|E|C + NC) + O(n_{QSGC}), \quad (144)$$

$$T_{QSGC} = \Omega(|E|C + NC) + O(2D)O(C^2)O(D_{QSGC}). \quad (145)$$

We see that the input problem compromises any significant quantum advantage in this case as the first terms in both of Eqs. 144 and 145 are comparable to the classical SGC space and time complexities. There could be significant differences in the overheads, but in general the problem as it stands is not scalable. Hence, we must assume an efficient oracle or QRAM to bypass the input problem for now. Hereafter, the classical compilation complexities are ignored.

F.2 Complexity analysis for fixed C

For our purposes, it is enough to consider C as well as all precisions and failure probabilities fixed as we want to vary the graph size N and connectivity k . The depth D is also strictly dependent on the data register and should vary only with C . This simplifies the complexities to

$$S_{CSGC} = O(N(k + C)) = O(Nk), \quad (146)$$

$$T_{CSGC} = O(N(k + C)) = O(Nk), \quad (147)$$

$$S_{QSGC} = O(n_{anc} + n'_{anc}), \quad (148)$$

$$T_{QSGC} = \tilde{O} \left(N \frac{\log(n_{anc})}{n_{anc}} + N \log(N) \cdot s \log(s) \frac{\log(n'_{anc})}{n'_{anc}} \right). \quad (149)$$

In Eqs. 146 and 147, $C = O(1)$ in N and $k = \Omega(1)$ so term Nk dominates. Maintaining the assumption of (almost) regular graphs, we can write $k = \Theta(s)$ given that $s - 1$ is the maximum degree of any node, and k is the average degree. We now show that exponential advantage can be achieved for either space or time complexities, but not both, by classifying them over the choices of n_{anc} and n'_{anc} . We split the analysis in three cases: a balanced polynomial advantage in space and time complexities, an exponential speed-up with maximized space reduction, and an exponential reduction with maximized speed-up. As we will see below, the latter case is a bit trickier and the choice of ancillas depends on the sparsity of the adjacency matrix.

F.2.1 Polynomial speed-up and space reduction

For this task we choose a sub-exponential number of ancillas, $n_{anc} = n'_{anc} = O(N^\alpha)$, with $\alpha \in (0, 1)$. This obeys the bounds in Eq. 141 and Eq. 142. Then, the updated space and time complexities in Eq. 148 and Eq. 149 are

$$S_{QSGC} = O(N^\alpha), \quad (150)$$

$$T_{QSGC} = \tilde{O} \left(N \frac{\alpha \log(N)}{N^\alpha} + N \log(N) \cdot s \log(s) \frac{\alpha \log(N)}{N^\alpha} \right) = \tilde{O}(N^{1-\alpha} s \log^2(N) \log(s)). \quad (151)$$

The sparsity $s \in [1, N]$ is an integer, so $\Omega(1) \leq s \leq O(N)$. Then, for each limit, we have the following cases for the space reduction:

$$\begin{cases} s = \Theta(1) & \Rightarrow & S_{CSGC} = O(N) & \Rightarrow & S_{QSGC} = O(S_{CSGC}^\alpha), \\ s = \Theta(N) & \Rightarrow & S_{CSGC} = O(N^2) & \Rightarrow & S_{QSGC} = O(S_{CSGC}^{\alpha/2}), \end{cases} \quad (152)$$

both leading to a polynomial advantage. Strictly speaking, the space reduction is sublinear.

For the time speed-up:

$$\left\{ \begin{array}{l} s = \Theta(1) \Rightarrow \begin{cases} T_{CSGC} = O(N), \\ T_{QSGC} = \tilde{O}(N^{1-\alpha} \log^2(N)) \Rightarrow \sigma_{q/cl} = \tilde{\Omega}(N^\alpha / \text{polylog}(N)), \end{cases} \\ s = \Theta(N) \Rightarrow \begin{cases} T_{CSGC} = O(N^2), \\ T_{QSGC} = \tilde{O}(N^{2-\alpha} \log^3(N)) \Rightarrow \sigma_{q/cl} = \tilde{\Omega}(N^{\alpha/2} / \text{polylog}(N)). \end{cases} \end{array} \right. \quad (153)$$

Therefore, in both cases (and thus for any intermediate sparsity), the advantage is *polynomial*. In terms of the system size N , the speed-up is also sublinear. This is the optimal balance as trying to obtain better scaling for one type of complexity will negatively affect the other one. The choice of α depends on the type of hardware onto which this model is implemented.

F.2.2 Exponential space reduction

Here we focus on obtaining an exponential space reduction with optimal time speed-up. For that reason, n_{anc} and n'_{anc} must be of order $O(\text{polylog}(N))$. By explicitly defining the leading orders as $n_{anc} = O(\log^l(N))$ and $n'_{anc} = O(\log^{l'}(N))$, Eq. 149 becomes

$$T_{QSGC} = \tilde{O} \left(N / \log^l(N) + N \cdot s \log(s) / \log^{l'-1}(N) \right). \quad (154)$$

For $s = \Theta(1)$, the terms are comparable up to a polylogarithm, and for $\min(l, l' - 1) \geq 1$, the speed-up is *polylogarithmic*. In fact, thanks to the lower bound of n_{anc} , the condition is simply $l' \geq 2$. For $s = \Theta(N)$, the second term dominates, and given that $T_{CSGC} = Ns$, the speed-up is polylogarithmic for $l' \geq 3$.

F.2.3 Exponential speed-up

In this case, in order for the runtime to gain an exponential speed-up, each of the two terms in Eq. 149 must be polylogarithmic. For that to happen, we must choose $n_{anc} = O(N / \text{polylog}(N))$ and $n'_{anc} = O(Ns / \text{polylog}(N))$. By setting the dominant powers as m and m' respectively, Eq. 149 becomes

$$T_{QSGC} = \tilde{O} \left(N \frac{\log(N / \log^m(N))}{N / \log^m(N)} + N \log(N) \cdot s \log(s) \frac{\log(Ns / \log^{m'}(N))}{Ns / \log^{m'}(N)} \right) \quad (155)$$

$$= \tilde{O}(\log^{m+1}(N) + \log^{m'+2}(N) \log(s)) \quad (156)$$

$$= \tilde{O}(\text{polylog}(N)). \quad (157)$$

The space reduction is then

$$S_{QSGC} = O \left(N / \log^m(N) + Ns / \log^{m'}(N) \right). \quad (158)$$

For $s = \Theta(1)$, the terms are comparable and for $\min(m, m') \geq 1$ the space reduction is *polylogarithmic*. For $s = \Theta(N)$, the second term dominates and regardless of $m \in \mathbb{Z}$, the condition reduces to $m' \geq 1$.

F.3 A more accurate C -dependence

In practice, for an ever-increasing graph size, having a limited number of features leads to many nodes becoming indistinguishable, having the same features. If one wants to be able to distinguish between all nodes if necessary, C should increase at least logarithmically with the number of nodes. If one also desires additional expressivity from the embedding process, an even better choice that is still trainable is $C = O(\text{polylog}(N)) \equiv O(\log^c(N))$, where c is the leading order. How does this change our results?

In terms of trainability, the barren plateau is still avoided for most ansätze, as

$$\mathbb{E}_{i,t}[\text{Var}_{\theta}(\partial_{\mu} L_{QSGC})] \sim \Theta(N^{-1} \log^{-c}(N) + \log^{-2c}(N)) = \Theta(\log^{-2c}(N)). \quad (159)$$

Furthermore, the number D of weights θ_η now can depend on N . In order not to overfit, $D \ll O(C)$, so in the Big- $\tilde{O}$ notation, it can be dismissed.

With this form of C , the new complexities for the classical SGC are the following:

$$S_{CSGC} = O(Ns + N \log^c(N)), \quad (160)$$

$$T_{CSGC} = O(Ns \log^c(N) + N \log^{2c}(N)). \quad (161)$$

Moving to the QSGC, the space and time complexities from Eq. 144 and Eq. 145 become

$$S_{QSGC} = \tilde{O}(n_{anc} + n'_{anc}), \quad (162)$$

$$\begin{aligned} T_{QSGC} &= O(\log^{2c}(N)) \tilde{O} \left(N \log^c(N) \frac{\log(n_{anc})}{n_{anc}} + N \log(N) \cdot s \log(s) \frac{\log(n'_{anc})}{n'_{anc}} \right) \\ &= \tilde{O} \left(N \log^{3c}(N) \frac{\log(n_{anc})}{n_{anc}} + N \log^{c+1}(N) \cdot s \log(s) \frac{\log(n'_{anc})}{n'_{anc}} \right). \end{aligned} \quad (163)$$

Similarly, the bounds on the numbers of ancillary qubits change as

$$\Omega(\log(N)) \leq n_{anc} \leq O(N \log^c(N)), \quad (164)$$

$$\Omega(\log(N)) \leq n'_{anc} \leq O(N \log(N) \cdot s \log(s)). \quad (165)$$

We observe that, in fact, the bounds are broadened by this dependence.

F.3.1 Advantage regimes

The regimes described in the previous section are preserved as we let C depend on N ; the only difference is that the minimum dominant powers l , l' , m , and m' generally need to make up for the extra powers c in the QSGC model, but benefit from the extra polylogarithmic burden that the CSGC acquires.

Polynomial speed-up and space reduction. Because polynomial powers do not interact with logarithmic ones, this regime is completely unchanged. That is to say, $\alpha \in (0, 1)$ is still valid.

Exponential space reduction. In this case, n_{anc} needs to make up for the factor of $\log^{3c}(N)$ and $n_{anc'}$ for $\log^{c+1}(N)$. Therefore, the following updates are required:

$$s = \Theta(1) \Rightarrow \min(l - c, l' + c - 1) \geq 1; \quad (166)$$

$$s = \Theta(N) \Rightarrow l' \geq 3, \text{ unchanged}. \quad (167)$$

Exponential speed-up. Similar to the case above, the only update is on the conditions of the dominant logarithmic powers:

$$s = \Theta(1) \Rightarrow \min(m, m') \geq c + 1; \quad (168)$$

$$s = \Theta(N) \Rightarrow m' \geq 1, \text{ unchanged}. \quad (169)$$

F.4 Low-rank simulability

An efficient oracle is a severe assumption in general; to even out the comparison, we can strictly ignore the data loading process for both the classical and the quantum models, and strictly compare the architectures themselves. In this case, Tang's algorithm [32] that approximates close-to-low-rank matrices with an l^2 -norm sampling algorithm becomes the new reference for the classical SGC.

Namely, the algorithm can approximate a matrix A of rank r and dimension $(M \times N)$ in $O(\text{polylog}(M, N) \text{poly}(r, 1/\varepsilon))$. The approximation precision ε is defined such that $\|A_{\text{approx}} - A\|_F \leq \varepsilon \|A\|_F$ bounds the Frobenius norm of the error between the approximation and the exact matrix. Note that $r \leq \min(M, N)$. More explicitly, we check that the runtime goes as

$$T_{\text{sampling}} = \tilde{O} \left(\log^2(MN) \cdot \max \left\{ \frac{r^{16.5}}{\varepsilon^{51}}, \frac{r^{12}}{\varepsilon^{48}} \right\} \right). \quad (170)$$

The first, $\log^2(MN)$ factor accounts for the accessing of the data and the second factor is the number of queries required. The rank and error are assumed to vary much more slowly than the matrix dimensions, thus the Big- $\tilde{O}$ also excludes single logarithms of r and ε in addition to all doubly-logarithmic functions.

The space complexity required by the algorithm is

$$S_{\text{sampling}} = O \left(w \log^2(MN) + \max \left\{ \frac{r^{11}}{\varepsilon^{34}}, \frac{r^8}{\varepsilon^{32}} \right\} \right), \quad (171)$$

where the first term is the data loading (for w non-zero values of the sampled matrix) and the second is the auxiliary space required to perform the sampling. In order to make a fair analysis, we exclude the data structure from the calculation (considered to be as efficient as the quantum oracle of the QSGC).

To find the classically simulable regimes of the QSGC algorithm, we employ Tang's algorithm as follows: we find an approximation for the cost

$$L_{QSGC}^* = \text{Tr}(Y^T \hat{A} X U(\theta)), \quad (172)$$

where X and Y are the loaded input and target feature vectors, presented as $(N \times C)$ matrices. The star is used to show that this cost is the dequantized version of the quantum cost. Considering the slow dependence $C = O(\log^c(N))$ for generality, only the matrix

$$M = \frac{1}{p_{jk}} Y_j^T \cdot \hat{A}_{jk} \cdot X_k \quad (173)$$

needs to be sampled. Here, p_{jk} is the probability of choosing nodes j and k when performing the sampling, which we control.

To find the estimator of M , we assume the data structure in [32] for matrix $\hat{A}$. By performing the l^2 -norm sampling to find $\hat{A}_{jk}$ with probability p_{jk} , we can infer the relevant nodes j and k and thus extract the feature columns Y_j and X_k from similar data structures. Then, we sample the estimator for M , which gives the result

$$\mathbb{E}_{\text{est.}}[M] = \sum_{j,k=1}^N p_{jk} M = Y^T \hat{A} X. \quad (174)$$

Then, taking the trace of this result with the weight matrix $U(\theta)$ leads to an estimate of the cost.

Here, we consider ε to be a fixed error that only adds an overhead. Note that this overhead may be very large due to the powers involved. Therefore, in our case, a dequantized version of the QSGC using Tang's algorithm would have the following complexities:

$$S_{QSGC}^* = O(r^{11}), \quad (175)$$

$$T_{QSGC}^* = \tilde{O} \left([\log^2(N) + O(C) + O(C^2)] \cdot r^{16.5} \right) + O(C^2), \quad (176)$$

The difference between Eq. 176 and Eq. 170 is that in addition to the search through the data structure, for each query we also compute the matrix M which gives as $O(C)$ time retrieving the feature vectors and $O(C^2)$ time constructing the matrix. The final $O(C^2)$ comes from taking the trace $\text{Tr}(MU(\theta))$. Given that the rank is at least $O(1)$, the trace compute can be absorbed into the first term. Therefore, the time complexity simplifies to

$$T_{QSGC}^* = \tilde{O} \left(\log^{2\max(1,c)}(N) \cdot r^{16.5} \right). \quad (177)$$

Relation to sparsity. In Lemma 13, we show that for a normalized adjacency matrix with self-loops $\hat{A}$ with sparsity s , the rank is lower-bounded by $r \geq N/s$. Therefore, we can impose a lower-bound on the sparsity, $s = \Omega(N/r)$.

Sublinear rank. In order for this algorithm to gain an advantage over the standard CSGC model, $r \ll O(N)$ is required. Thus, we choose a form $r = O(N^\beta)$, where $\beta \in [0, 1)$. Employing the polynomial speed-up and space reduction case described in Appendix F.3.1, the dominant powers of N in the space and time complexities are respectively α and $2 - \alpha$. Then, equating these with the ones resulting from Eqs. 176 and 175, the conditions for dequantization are

$$11\beta < \alpha \quad \text{and} \quad 16.5\beta < 2 - \alpha - \beta, \quad (178)$$

which means that

$$\beta < \min \left\{ \frac{\alpha}{11}, \frac{2 - \alpha}{17.5} \right\}. \quad (179)$$

As α is a controllable parameter, the maximum possible choice of β is given when the two functions of α are equal:¹⁸

$$\frac{\alpha_m}{11} = \frac{2 - \alpha_m}{17.5} \Rightarrow \alpha_m \approx 0.77 \Rightarrow \beta_{max} \approx 0.07. \quad (180)$$

In our previous calculation we assumed that the numbers of ancillas n_{anc} and n'_{anc} vary with the same exponent α . Even by loosening this assumption and giving different subunitary exponents to both (which retains the polynomial advantage over the standard CSGC), the maximum possible β is 0.09, which is still very small. This confirms that by using Tang's algorithm, the rank of the adjacency matrix must be very low, $r = O(N^{0.09})$, simply so that $11\alpha < 1$. Otherwise, there is no space reduction.

Polylogarithmic rank. In the case where $r = O(\log^\rho(N))$ for some integer ρ , the new classical algorithm gains both exponential space reduction and speed-up compared to the standard CSGC. Compared to the QSGC which has a necessary trade-off, this is net superior, gaining an overall exponential advantage.

To this end, we conclude that the QSGC architecture is classically simulable for any $r = O(N^{1/11})$ and loses the advantage entirely for $r = O(\text{polylog}(N))$.

G Helper lemmas

In this section, we define and prove a few lemmas that will be relevant for the detailed gradient analysis given in Appendix C.

G.1 Sums involving Hamming weights

First, we find the explicit forms for the summations labelled as H_1 , H_2 , and H'_1 in Section C.6, by defining the following lemma:

Lemma 8 *The following summations over all nonzero bit-strings of length N_k hold:*

$$H_1 := \sum_{\vec{\gamma} \neq 0} \frac{\gamma_b}{w(\vec{\gamma})} = \frac{2^{N_k} - 1}{N_k}, \quad (181)$$

$$H_2 := \sum_{\vec{\gamma} \neq 0} \frac{\gamma_b \gamma_c}{w(\vec{\gamma})} = \begin{cases} \frac{(N_k - 2)2^{N_k - 1} + 1}{N_k(N_k - 1)} \equiv H'_2 & \text{if } b \neq c, \\ H_1 & \text{otherwise,} \end{cases} \quad (182)$$

$$H'_1 := \sum_{\vec{\gamma} \neq 0} \frac{\gamma_b}{\sqrt{w(\vec{\gamma})}} = 2^{N_k - 1} \left[\sqrt{\frac{2}{N_k}} \left(1 - \frac{1}{8N_k} \right) + O(N_k^{-5/2}) \right]. \quad (183)$$

where $w(\vec{\gamma})$ denotes the Hamming weight of a nonzero bit string $\vec{\gamma}$ of length $N_k \geq 3$ and b and c label the bits inside $\vec{\gamma}$.

¹⁸This maximisation condition works because the two functions have opposite monotony.

Proof. For H_1 , all strings where $\gamma_b = 0$ give no contribution, so the sum reduces to

$$H_1 = \sum_{\vec{\gamma}'} \frac{1}{w(\vec{\gamma}') + 1}, \quad (184)$$

where $\vec{\gamma}'$ is an $(N_k - 1)$ -bit string equal to $\vec{\gamma}$ with γ_b removed. For this sum, we notice that there are $\binom{N_k-1}{k}$ strings that have a Hamming weight $w(\vec{\gamma}) \equiv k$. Therefore, the sum can be further simplified to

$$H_1 = \sum_{k=0}^{N_k-1} \frac{1}{k+1} \binom{N_k-1}{k} = \frac{1}{N_k} \sum_{k=0}^{N_k-1} \binom{N_k}{k-1} = \frac{2^{N_k} - 1}{N_k}. \quad (185)$$

The second simplification comes from rearranging terms in the binomial formula $\binom{N_k-1}{k} = \frac{k+1}{N_k} \binom{N_k}{k-1}$, and the final result is derived using $\sum_{k=0}^{N_k} \binom{N_k}{k} = 2^{N_k}$.

For H_2 , there are two options. If $b = c$, then in Eq. 182, $\gamma_b \gamma_c = \gamma_b^2 = \gamma_b$, so $H_2 = H_1$. If $b \neq c$, we perform a similar calculation as done in Eq. 185, after eliminating both bits γ_b and γ_c from the string $\vec{\gamma}$, resulting into a new string $\vec{\gamma}''$ of length $N_k - 2$:

$$H_2 = \sum_{\vec{\gamma}''} \frac{1}{w(\vec{\gamma}'') + 2} = \sum_{k=0}^{N_k-2} \frac{1}{k+2} \binom{N_k-2}{k}. \quad (186)$$

Next, we write the terms as

$$\frac{1}{k+2} \binom{N_k-2}{k} = \frac{1}{N_k(N_k-1)} \left[N_k \binom{N_k-1}{k+1} - \binom{N_k}{k+2} \right], \quad (187)$$

which can be verified by writing out the binomial coefficients. Finally, by adding and subtracting terms to complete the summations $\sum_{k=0}^n \binom{n}{k} = 2^n$ and for $n = N_k - 1$ and N_k respectively, we obtain

$$H_2 = \frac{1}{N_k(N_k-1)} [N_k(2^{N_k-1} - 1) - (2^{N_k} - N_k - 1)] = \frac{(N_k-2)2^{N_k-1} + 1}{N_k(N_k-1)}. \quad (188)$$

The final expression, Eq. 183, can be approximated using a Taylor expansion. We start with the same simplification as done in Eq. 185:

$$H_1' = \sum_{k=0}^{N_k-1} \frac{1}{\sqrt{k+1}} \binom{N_k-1}{k} \equiv 2^{N_k-1} \mathbb{E}_k^{\mathbf{Bi}} \left[\frac{1}{k+1} \right], \quad (189)$$

where we have normalized by the exponential in order to reach a binomial distribution $\mathbf{Bi}(N_k - 1, 1/2)$ — hence the summation becomes an expected value.

For this binomial distribution, the mean is $\mu = \frac{N_k-1}{2}$, and the variance is $\sigma^2 = \frac{N_k-1}{4}$. Thus, an expansion around the mean of any function $f(k)$,

$$f(k) = f(\mu) + f'(\mu)(k - \mu) + \frac{1}{2} f''(\mu)(k - \mu)^2 + \xi \quad (190)$$

has an expected value

$$\mathbb{E}_k^{\mathbf{Bi}}[f(k)] = f(\mu) + f'(\mu) \mathbb{E}_k^{\mathbf{Bi}}[(k - \mu)] + \frac{1}{2} f''(\mu) \mathbb{E}_k^{\mathbf{Bi}}[(k - \mu)^2] + \mathbb{E}_k^{\mathbf{Bi}}[\xi]. \quad (191)$$

Here, ξ is the error of the Taylor expansion. The first moment vanishes (as well as all odd moments in the error, $\mathbb{E}_k^{\mathbf{Bi}}[(k - \mu)^{2l+1}]$). Furthermore, the second moment is simply the variance; therefore

$$\mathbb{E}_k^{\mathbf{Bi}}[f(k)] = f(\mu) + \frac{1}{2} f''(\mu) \sigma^2 + \mathbb{E}_k^{\mathbf{Bi}}[\xi]. \quad (192)$$

In our case, the function analyzed is $f(k) = (k+1)^{-1/2}$, so calculating the derivatives leads to

$$\mathbb{E}_k^{\text{Bi}}[(k+1)^{-1/2}] = \sqrt{\frac{2}{N_k+1}} + \frac{3\sqrt{2}}{8} \frac{N_k-1}{(N_k+1)^{\frac{5}{2}}} + O\left(N_k^{-\frac{5}{2}}\right), \quad (193)$$

where the $O(N_k^{-5/2})$ term comes from the error. By also involving a large- N_k approximation, we reach the desired result for H'_1 . $\square$

For the use of binomial formulas and that of the expected value of the Taylor expansion for H'_1 , we recommend reading [52] and [53], respectively. We also show some approximations necessary for the final formula in the following lemma:

Lemma 9 *The following relations hold:*

$$\frac{(H'_1)^2}{(2^{N_k}-1)^2} = \frac{1}{2N_k} + O(N_k^{-2}), \quad (194)$$

$$\frac{H'_2}{2^{N_k}-1} = \frac{1}{2N_k} + O(N_k^{-2}), \quad (195)$$

$$\frac{H_1 - H'_2}{N_k(2^{N_k}-1)} = \frac{1}{2N_k^2} + O(N_k^{-3}). \quad (196)$$

Proof. Following the previous lemma, we use Eq. 183 to calculate

$$\begin{aligned} \frac{(H'_1)^2}{(2^{N_k}-1)^2} &= \frac{2^{2N_k-2} \left[\frac{2}{N_k} \left(1 - \frac{1}{8N_k}\right)^2 + O(N_k^{-3}) \right]}{(2^{N_k}-1)^2} \\ &= \frac{1}{4} \left[\frac{2}{N_k} \left(1 - \frac{1}{8N_k}\right)^2 + O(N_k^{-3}) \right] + O(\exp(-N_k)) \\ &= \frac{1}{2N_k} + O(N_k^{-2}), \end{aligned} \quad (197)$$

where in the first line we neglected the “1” term adding an exponentially small error, and then reduced to $O(N_k^{-2})$. The following relation is computed as

$$\begin{aligned} \frac{H'_2}{2^{N_k}-1} &= \frac{(N_k-2)2^{N_k-1} + 1}{N_k(N_k-1)(2^{N_k}-1)} \\ &= \frac{N_k-2}{2N_k(N_k-1)} + O(\exp(-N_k)) \\ &= \frac{1}{2N_k} \cdot \frac{1-2N_k^{-1}}{1-N_k^{-1}} \\ &= \frac{1}{2N_k} \left(1 - \frac{2}{N_k} + O(N_k^{-2})\right) \left(1 + \frac{1}{N_k} + O(N_k^{-2})\right) \\ &= \frac{1}{2N_k} + O(N_k^{-2}). \end{aligned} \quad (198)$$

Finally, Eq. 196 is calculated as follows:

$$\begin{aligned} \frac{H_1 - H'_2}{N_k(2^{N_k}-1)} &= \frac{1}{N_k} \left[\frac{1}{N_k} - \left(\frac{1}{2N_k} + O(N_k^{-2}) \right) \right] \\ &= \frac{1}{2N_k^2} + O(N_k^{-3}). \end{aligned} \quad (199)$$

Thus the three asymptotes are derived. $\square$

G.2 Constraining the alternating sum in the expected variance

Here we prove that by adding a very loose constraint, we can force the summation in Eq. 92 to always be positive. We do this in the following lemma:

Lemma 10 *For $\{S_{\vec{\alpha}}^v\}_{\vec{\alpha}}$ given in Eq. 67 and for some ansatz $U(\theta)$ defined in Eq. 35, the map $W_\lambda \mapsto V_\lambda W_\lambda$ flips the sign of*

$$\sum_{\vec{\alpha} \in \{0,1\}^D} (-1)^{w(\vec{\alpha})} \text{Re}(S_{\vec{\alpha}}^v) \mapsto - \sum_{\vec{\alpha} \in \{0,1\}^D} (-1)^{w(\vec{\alpha})} \text{Re}(S_{\vec{\alpha}}^v). \quad (200)$$

Proof. When mapping $W_\lambda \mapsto V_\lambda W_\lambda \equiv \tilde{W}_\lambda$, the unitary matrix $D_{\vec{\alpha}}$ transforms as

$$\begin{aligned} D_{\vec{\alpha}} &= \prod_{\eta=D}^1 V_\eta^{\alpha_\eta} W_\eta \\ &= V_D^{\alpha_D} W_D \dots V_{\lambda+1}^{\alpha_{\lambda+1}} W_{\lambda+1} \cdot V_\lambda^{\alpha_\lambda} W_\lambda \cdot V_{\lambda-1}^{\alpha_{\lambda-1}} W_{\lambda-1} \dots V_1^{\alpha_1} W_1 \\ &\mapsto V_D^{\alpha_D} W_D \dots V_{\lambda+1}^{\alpha_{\lambda+1}} W_{\lambda+1} \cdot V_\lambda^{\alpha_\lambda} \tilde{W}_\lambda \cdot V_{\lambda-1}^{\alpha_{\lambda-1}} W_{\lambda-1} \dots V_1^{\alpha_1} W_1 \\ &= V_D^{\alpha_D} W_D \dots V_{\lambda+1}^{\alpha_{\lambda+1}} W_{\lambda+1} \cdot V_\lambda^{\alpha_\lambda+1} W_\lambda \cdot V_{\lambda-1}^{\alpha_{\lambda-1}} W_{\lambda-1} \dots V_1^{\alpha_1} W_1 \\ &\equiv D_{\vec{\alpha}'}, \end{aligned} \quad (201)$$

where $\vec{\alpha}'$ has all bits equal to those of $\vec{\alpha}$, except for $\alpha'_\lambda := \alpha_\lambda \oplus 1$. Therefore,

$$\text{Re}(S_{\vec{\alpha}}^v) \mapsto \text{Re}(S_{\vec{\alpha}'}^v), \quad (202)$$

and the sum

$$\begin{aligned} \sum_{\vec{\alpha} \in \{0,1\}^D} (-1)^{w(\vec{\alpha})} \text{Re}(S_{\vec{\alpha}}^v) &\mapsto \sum_{\vec{\alpha} \in \{0,1\}^D} (-1)^{w(\vec{\alpha})} \text{Re}(S_{\vec{\alpha}'}^v) \\ &\equiv \sum_{\vec{\alpha}' \in \{0,1\}^D} (-1)^{w(\vec{\alpha}') \oplus 1} \text{Re}(S_{\vec{\alpha}'}^v) \\ &= - \sum_{\vec{\alpha}' \in \{0,1\}^D} (-1)^{w(\vec{\alpha}')} \text{Re}(S_{\vec{\alpha}'}^v). \end{aligned} \quad (203)$$

Here we first applied the map, thus obtaining $\text{Re}(S_{\vec{\alpha}'}^v)$, then explicitly rewrote $\vec{\alpha}$ in terms of $\vec{\alpha}'$. Flipping one bit is a bijection, so the sum maintains its range. Finally, after extracting the additional -1 factor, we can relabel the summation index with $\vec{\alpha}$, hence proving the lemma. $\square$

Therefore, if our summation is negative, we can pick any of the W_η gates (or any odd number of them, for that matter) and multiply them by its associated V_η , in order to make the sum positive. Note that, by applying the map $W_\lambda \mapsto \tilde{W}_\lambda$, the $S_{\vec{\alpha}}^c$ summation in the variance is unchanged.

G.3 Curse of dimensionality

The curse of dimensionality, first mentioned by Richard Bellman in 1957 [54], is a phenomenon that occurs in many areas of computational and data science [55–57]. In our case, we formally define it as a vanishingly small overlap between two random states as their Hilbert space dimension increases:

Lemma 11 *For two random n -qubit states $|\psi\rangle$ and $|\phi\rangle$, their overlap (modulus squared) averages to*

$$\mathbb{E}_{\mathcal{H}}[|\langle \phi | \psi \rangle|^2] = \frac{1}{2^n}. \quad (204)$$

Proof. By mapping state $|\psi\rangle \equiv U_\psi |0\rangle$ to $|0\rangle$ with some gate $U_\psi^\dagger$, the state $|\phi\rangle$ becomes $|\phi'\rangle = U_\psi^\dagger |\phi\rangle$. As such, the inner product becomes

$$\langle \phi | \psi \rangle = \langle \phi | U_\psi U_\psi^\dagger | \psi \rangle = \langle \phi' | 0 \rangle, \quad (205)$$

where $|\phi'\rangle$ is random. By writing the amplitudes of $|\phi'\rangle \equiv \sum_i c_i |i\rangle$ in the standard basis, we have $|\langle\phi'|0\rangle|^2 = |c_0|^2$. Finally, the state considered is randomly selected from an isotropic distribution, so

$$\begin{aligned}\mathbb{E}_{\mathcal{H}}[|c_0|^2] &= \mathbb{E}_{\mathcal{H}}[|c_i|^2] \\ &= \frac{1}{2^n} \sum_{i=1}^{2^n} \mathbb{E}_{\mathcal{H}}[|c_i|^2] \\ &= \frac{1}{2^n} \mathbb{E}_{\mathcal{H}} \left[\sum_{i=1}^{2^n} |c_i|^2 \right] \\ &= \frac{1}{2^n},\end{aligned}\tag{206}$$

which completes the proof. $\square$

G.4 Calculating the $\hat{R}$ values

Here we provide a derivation for the $\hat{R}$ values used in Appendix D.1:

Lemma 12 *For the $\hat{R}_{||}$, $\hat{R}_{\wedge}$, $\hat{R}_{\vee}$, and $\hat{R}_{|}$ defined in Eqs. 115, 116, 117, and 118 respectively, the bounds for weighted k -regular graphs are the following:*

$$(N-1)^2 \leq \hat{R}_{||} \leq N(N-1)\tag{207}$$

$$0 \leq \hat{R}_{\wedge} \leq N-1\tag{208}$$

$$0 \leq \hat{R}_{\vee} \leq N-1\tag{209}$$

$$1 \leq \hat{R}_{|} \leq N.\tag{210}$$

We recall that N is the number of nodes.

Proof. First, we recall that for a k -regular graph, $k+1 = \tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. Therefore, the normalized version reads $\sum_j \hat{A}_{ij} = 1$.

In this case, we begin with $\hat{R}_{|}$. For the upper bound we take the square of a sum to be larger than the sum of squared terms:

$$\hat{R}_{|} = \sum_{i,j} \hat{A}_{ij}^2 \leq \sum_i \left(\sum_j \hat{A}_{ij} \right)^2 = \sum_i 1 = N.\tag{211}$$

The lower bound is done via the Cauchy-Schwarz inequality:

$$\hat{R}_{|} = \sum_{i,j} \hat{A}_{ij}^2 = \sum_i \frac{1}{s_i} \left(\sum_{j'=1}^{s_i} \hat{A}_{ij}^2 \right) \left(\sum_{j'=1}^{s_i} 1 \right) \geq \sum_i \frac{1}{s_i} \sum_{j'=1}^{s_i} \hat{A}_{ij} \geq \frac{1}{s} \sum_i \sum_j \hat{A}_{ij} = \frac{N}{s}.\tag{212}$$

Here, s_i is the sparsity of row i and the s is the row sparsity of matrix $\hat{A}$. The worst case is when $s = N$, so $\hat{R}_{|} \geq 1$.

The other $\hat{R}$ values only inherit the bounds from $\hat{R}_{|}$, the other terms being calculated precisely. In $\hat{R}_{\wedge}$, the first term is

$$\sum_{i,j,k} \hat{A}_{ij} \hat{A}_{kj} = \sum_j \left(\sum_i \hat{A}_{ij} \right) \left(\sum_k \hat{A}_{kj} \right) = \sum_j 1 \cdot 1 = N.\tag{213}$$

Therefore, by subtracting the bounds of $\hat{R}_{|}$, we obtain the desired bounds for $\hat{R}_{\wedge}$. Thanks to the symmetry of $\hat{A}$, the same bounds hold for $\hat{R}_{\vee}$.

Finally, by rewriting Eq. 115 in terms of Eqs. 116 and 117, we have

$$\hat{R}_{||} = \underbrace{\left(\sum_{i,j} \hat{A}_{ij}\right)}_{=N} \underbrace{\left(\sum_{k,l} \hat{A}_{kl}\right)}_{=N} - 2 \underbrace{\sum_{i,j,k} \hat{A}_{ij} \hat{A}_{kj}}_N + \hat{R}_{|}. \quad (214)$$

Plugging in the bounds for $\hat{R}_{|}$ gives the desired bounds. $\square$

G.5 Rank-sparsity relation

Here we prove that the normalized adjacency matrix with self-loops $\hat{A}$ of a graph has a high rank only if it is sparse.

Lemma 13 *For a graph, the normalized adjacency matrix with self-loops $\hat{A}$ is defined as given in Appendix A. Then, its rank r depends on its dimension N and sparsity s by the bound*

$$r \geq \frac{N}{s}. \quad (215)$$

Proof. First, $\hat{A} = \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}$, where $\tilde{D}$ are invertible operators. Its rank is the dimension of the image of $\hat{A}$, which we denote by $\text{rank}(\hat{A}) = \dim(\text{im}(\hat{A}))$. Therefore, by mapping $\text{im}(\hat{A}) \rightarrow \text{im}(\tilde{D}^{1/2} \hat{A})$ with the bijection $\tilde{D}^{1/2}$ to the left, we preserve the rank. Because transposition also preserves both rank and invertibility, applying invertible operators to the right also preserves the rank. Therefore,

$$\begin{aligned} \text{rank}(\tilde{A}) &= \text{rank}(\tilde{D}^{1/2}(\hat{A} \tilde{D}^{1/2})) = \text{rank}(\hat{A} \tilde{D}^{1/2}) = \text{rank}((\hat{A} \tilde{D}^{1/2})^T) \\ &= \text{rank}((\tilde{D}^{1/2})^T \cdot \hat{A}^T) = \text{rank}(\hat{A}^T) = \text{rank}(\hat{A}) = r. \end{aligned} \quad (216)$$

Here, we expanded $\tilde{A}$, then reduced the bijection to the left, transposed, reduced the new bijection to the left, and transposed again, in this order, to reach r .

We now start from the Cauchy-Schwarz inequality applied on the sum of the eigenvalues λ_i of $\tilde{A}$:

$$\left(\sum_{i=1}^r \lambda_i \cdot 1\right)^2 \leq \left(\sum_{i=1}^r 1\right) \cdot \left(\sum_{i=1}^r \lambda_i^2\right), \quad (217)$$

however, the LHS is simply the squared trace $\text{Tr}(\tilde{A}) = N$, and the sum over squared eigenvalues can be further bounded by

$$\sum_{i=1}^r \lambda_i^2 = \text{Tr}(\tilde{A}^2) = \sum_{i,j} \tilde{A}_{ij} \tilde{A}_{ji} = \sum_{i,j} \tilde{A}_{ij}^2 \leq N \cdot s, \quad (218)$$

which counts the maximum number of ones present in the matrix. Therefore, the Eq. 217 leads to

$$N^2 \leq r \cdot N \cdot s \quad \Rightarrow \quad r \geq N/s. \quad (219)$$

So a sparse graph has high rank r . $\square$