The Capacity of Quantum Neural Networks

Logan G. Wright[1,2,*] and Peter L. McMahon[1,2,†]
1. School of Applied and Engineering Physics, Cornell University, Ithaca, NY 14853, USA and
2. E. L. Ginzton Laboratory, Stanford University, Stanford, CA 94305, USA

A key open question in quantum computation is what advantages quantum neural networks (QNNs) may have over classical neural networks (NNs), and in what situations these advantages may transpire. Here we address this question by studying the memory capacity C of QNNs, which is a metric of the expressive power of a QNN that we have adapted from classical NN theory. We present a capacity inequality showing that the capacity of a QNN is bounded by the information W that can be trained into its parameters: $C \leq W$. One consequence of this bound is that QNNs that are parameterized classically do not show an advantage in capacity over classical NNs having an equal number of parameters. However, QNNs that are parameterized with quantum states can have exponentially larger capacities (although may require exponentially longer execution times). We illustrate our theoretical results with numerical experiments by simulating a particular QNN based on a Gaussian Boson Sampler. We also study the influence of sampling due to wavefunction collapse during operation of the QNN, and provide an analytical expression connecting the capacity to the number of times the quantum system is measured.

While they have been of interest since the 1990s [1–3], research on quantum artificial neural networks (QNNs) is expanding, invigorated by possible implementations on near-term hardware [4]. What is a QNN? Here we take a broad definition that encompasses emergent themes across numerous proposals (and that, like classical NNs, has a mostly historical connection to biological NNs). QNNs being considered for near term implementations are variational quantum algorithms [5–8]: quantum circuits having gates whose parameters are adjusted to perform a desired transformation on either classical or quantum information that is fed through the circuit. QNN architectures include "quantized" classical NNs [4, 9–17] and Boltzmann machines [18–22], as well as (within our broad definition) schemes that exploit trainable input-output mapping of quantum circuits more generally or opportunistically [23–27].

What are QNNs good for? While there is not yet a consensus, some proposals have indicated potential quantum speed-ups or advantages in the size or training of the network (e.g., [11, 18, 20, 28–31]). Many are intended as general quantum learning machines capable of a range of tasks (e.g., [4, 10, 12–14, 32, 33]). This possibility of generality is stimulating: classical NNs have found widespread use in part because even inexpert users can experimentally discover algorithms for diverse tasks. Last, many QNNs appear suitable for implementation on noisy intermediate-scale quantum (NISQ) computing devices [34], for which there is an ongoing search for suitable algorithms.

Despite its promise, QNN development faces challenges. First, a systematic way of comparing QNNs would be helpful due to the many diverse proposals and possible hardware implementations, spanning numerous platforms and discrete and continuous-variable quantum systems. Training QNNs is also challenging [20, 31, 35–38]. Many QNN proposals for supervised learning use classical computers to train the (classical-valued) parameters of the quantum circuit. The probabilistic nature of the quantum variables used imposes trade-offs between time and accuracy, and the optimization problem landscape poses challenges for gradient descent [38]. Quantum Boltzmann machines [22], especially quantum training of Boltzmann machines [18, 20] are promising, but other proposals [14, 17, 23, 35–39] may also help. Finally, extrapolation of small-scale QNNs is challenging: typically researchers cannot confidently predict how or if scaled-up QNNs will be useful.

Our main result is inspired by information-theoretic models of classical NNs [40–47], and is a generalization of the memory capacity [41, 44–50] that is related to the Vapnik-Chervonenkis dimension [40, 51, 52]. These measures aim to quantify the complexity of representation a given classifier can achieve. The capacity is simplistic, but can be insightful. First, with a suitable learning method a classifier may learn a range of tasks whose maximum complexity is bounded by its capacity. Second, the capacity describes the information that must be provided in training to ensure generalization, i.e., to avoid overfitting (this can also be accomplished by automatically restricting capacity, as in parameter regularization).

Adapting capacity to QNNs allows us to quantify the range of tasks a QNN may be used for, and its training requirements. Quantum learning machines intuitively support more complex models than classical ones, so overfitting may ultimately be a challenge. Conversely, learning *arbitrary* quantum tasks requires enormous capacity: arbitrary quantum models may be exponentially more complex than those required for classical tasks.

Here, by considering the memory capacity in information units (bits or qubits), we show that a simple capacity measure can be universally defined to encompass all learning machines: classical, quantum and hybrids. To relate this capacity to machine's physical implementa-

tion, we introduce a capacity inequality that bounds capacity by the information that can be imparted by training to the machine's trainable parameters. We anticipate this approach will be useful for establishing rough guidelines for QNN design and application, similar to how capacity has been used to understand classical NNs [41, 44–50].

The idea of memory capacity as the limiting learnable complexity is intuitive. Consider a task such as a sequence of errands. The instructions for completing the task may be partially redundant, but there is a minimum amount that one must remember to execute the task flawlessly. In the limit of a random sequence of errands, the impossibility of universal lossless compression implies that a learner must remember all the information in the instructions, T, exactly. Let us call the size of the largest random task a learner can "learn" (memorize) its capacity, C. A necessary condition if there are T bits to memorize is that the learner possess the ability to change at least T bits of its degrees of freedom as it learns. This follows from the pigeonhole principle. When trained for an arbitrary task specified by data $T > C$, a learner cannot memorize all T bits. Instead, to minimize training error it must learn a compressed model by exploiting patterns within the data.

Definitions: More formally, we define the memory capacity, C, of a learning machine as the largest Nm bits of N, m-bit labels for which it can learn to label N general position inputs (which may be quantum and/or classical, depending on the machine) by any combination of N, m-bit labellings. General position implies uniformly distributed distinct random points, drawn from the across the range of acceptable inputs to the learning machine [41]. Essentially, C is the largest Nm-bit RAM the learning machine can be trained to emulate. We also introduce W, defined as the amount of information which can be stored through training in the trainable parameters of the learning machine. For example, consider a learning machine whose $N_{\rm w}$ classical trainable parameters w_i can each be trained with certainty to take on one of M_i distinct levels. Then $W = \sum_{i=1}^{N_{\rm w}} \log_2 M_i = \sum_{i=1}^{N_{\rm w}} b_i$ bits.

The rationale for these definitions is as follows. Consider the task of exactly learning a random map, i.e., mapping N uniformly distributed, distinct random inputs to N uniformly distributed, distinct random m-bit labels. Because this data is incompressible, a learning machine must be able to learn a model whose complexity, measured in terms of the information needed to describe it, is no less than the amount of information contained in the random labels. This statement assumes, however, that the learning machine's initial state (prior to learning) is completely uncorrelated with the random map; it may be possible to find a particular task or subset of T-bit tasks that the learning machine is (by chance or design) initialized to perform, or can learn with only small parameter adjustments. Such a machine would not necessarily be able to learn arbitrary T-bit tasks, nor function as a T-bit RAM. Therefore, for N inputs in general position, all possible labellings of N, m-bit numbers should be learnable exactly in order to ensure $C \geq Nm$ bits.

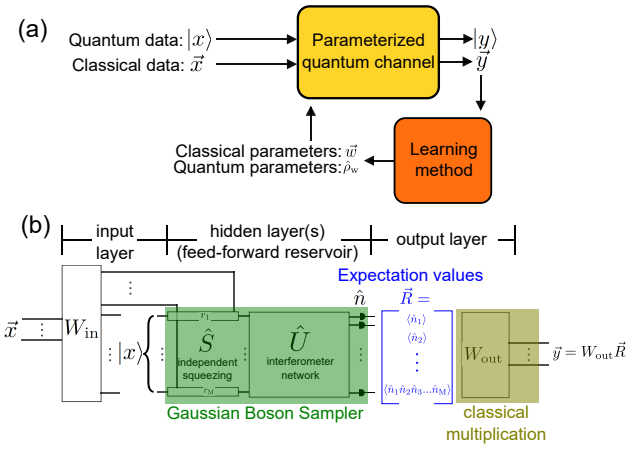

FIG. 1. (a) Schematic of a general QNN, a parameterized quantum channel (which could include unitary and/or dissipative quantum evolutions, classical processing, etc.) which is trained in a supervised fashion to optimize the classical and quantum parameters $\vec{w}$ and/or $\hat{\rho}_{\rm w}$ so that the QNN best approximates the transformation implied by the training data. (b) Schematic of a feed-forward quantum reservoir computer based on a Gaussian Boson Sampler. For classical tasks considered here, $|x\rangle = |0\rangle$ and data is then encoded through the squeezing parameters. Here, we take $W_{\rm in}$ to be the identify matrix.

Our main result is to relate C for an arbitrary learning machine to its physical implementation including its learning algorithm, via W, through the inequality:

$$C \leq W. \tag{1}$$

The proof is simple and has been essentially stated earlier in this paper and by others in more specific contexts, but for completeness we give it in full here. We will for now consider devices producing classical outputs, the quantum case will be considered subsequently.

Proof: If a learning machine is able to learn a model that requires T bits to specify, then the machine must possess degrees of freedom that can store at least T bits of information, those degrees of freedom must be trainable, and its training must be able to store at least T bits of information in them. The first requirement is just due to the pigeonhole principle, and the second and third requirements are necessary to describe parameterized learning machines. Thus C, the memory capacity or maximum model complexity of the learning machine, can be no larger than the information that can be stored in its trainable parameters through learning, W, i.e., $C \leq W$.

Until now, our considered learning machine has been general. Equation 1 *is* general: it applies to any learning machine with trainable parameters. We now consider

the parameterized quantum channel depicted in Fig. 1a, which describes a general feed-forward QNN. The device maps inputs – a tuple of quantum and classical data – to outputs that may also contain quantum and classical parts, i.e., $(|x\rangle, \vec{x}) \mapsto (|y\rangle, \vec{y})$. Supervised training of the QNN uses input-output pairs as training data (e.g., the x and $y = f(x)$ values from a function) or quantum channel (e.g., a unitary quantum circuit or dissipative evolution), and attempts to optimize the QNN's parameters to make the QNN's outputs for each input match the training set. In addition to depending on the QNN architecture (the layout of the QNN and its trainable parameters), C and W also depend on the execution and training protocols (which include, e.g., the input data encoding, the learning method, and how many repeats of the data or executions of the QNN occur per input [53–55]). Although C and W depend on these details, Eqn. 1 applies universally.

The interpretation of Eqn. 1 in the case of quantum parameterization and/or quantum tasks can be understood as follows. If the QNN's trainable parameters are quantum states that can be learned to some finite (m-bit) precision, they comprise a total density operator $\hat{\rho}_{\mathrm{w}}$ in a Hilbert space of dimension D, which can be described by a matrix containing $D^2 - 1$ real m-bit numbers. That is, $W = m(D^2 - 1)$ bits. Using this same technique of considering quantum objects as m-bit approximations of their classical matrix representations, we can similarly interpret C and quantum-output tasks. Quantum tasks, such as preparing states or learning a quantum circuit, are unitary approximation tasks. For example, if the optimal quantum model for a given quantum task is an M-qubit unitary, it may be described by a matrix with up to $2^{2M} - 1$ real degrees of freedom. An m-bit approximation of the unitary would be one in which each matrix element is correct to within m-bit precision. To learn an m-bit approximation of an *arbitrary* M-qubit unitary, one requires a QNN with $C \geq m(2^{2M} - 1)$ bits. C is not equivalent to computational hardness or "power", e.g. a circuit initialized to implement Shor's algorithm exactly can perform the hard task of factoring even with a small C. However, if we attempt to train it to perform other tasks, we can only expect it to learn tasks within some "radius" of C bits from the initial unitary (i.e., unitaries accessible by changing parameters by C bits). Finally, our definitions of C and W above apply to learning machines producing classical outputs. For machines producing quantum outputs (or producing samples from a probability distribution, as in generative tasks [56–58]), a generalization is to task the device with mapping each of N inputs to N label states [53]. To quantify C, tomography is performed to arbitrary precision (so that errors in the measured matrix elements are dominated by the imprecision of the QNN). The information content m of each label state is quantified as the summation of the QNN-limited precision across the degrees of freedom of each label state, i.e., $m = \sum_i \log_2 e_i^{-1}$, where e_i is the normalized absolute mean error in the reproduction of the states' i-th (real) degree of freedom.

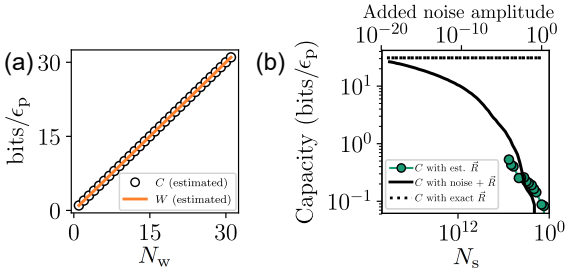

FIG. 2. Estimation of the capacity of the GBS-fQRC. (a) shows the estimated capacity and W for the GBS-fQRC in which expectation values in $\vec{R}$ are extracted directly from simulations. The units are relative to the numerical-noise-limited bit depth $\epsilon_{\mathrm{p}} \approx 45$ bits (for details, see Supplemental Material [53]). (b) shows the estimated capacity when the expectation values in $\vec{R}$ are instead estimated from N_{s} simulated measurements, versus the number of measurements used to make the estimate, N_{s} (bottom axis), and versus added noise amplitude (top axis).

As a concrete example, we consider a feed-forward reservoir computer (fQRC) based on a Gaussian Boson Sampler (GBS). The GBS-fQRC (Fig. 1b) consists of an input layer, where quantum or classical data is injected into a reservoir circuit (a GBS). The reservoir circuit is feed-forward and untrained; it serves to transform the input data into a feature space, similar to the scheme proposed in Ref. [26], producing a feature vector $\vec{R}$ of expectation values for each input. The final result $\vec{y}$ follows from a linear output layer matrix multiplication, which selects a linear combination of features to approximate the function, $y = W_{\mathrm{out}}\vec{R}$. Thus, this QNN produces a classical output regardless of whether its inputs are quantum or classical data. Training consists of a linear regression to optimize the matrix W_{out} over a training set (for details see [53]). This is a QNN analogue of the "extreme learning machine" (ELM) [59] in the classical literature. Recent works have explored quantum reservoir computers (QRC) [60–65] that adapt reservoir computing [66, 67] to quantum dynamical systems. The GBS-fQRC simplifies this approach by considering a feed-forward device, implemented with a GBS. Boson Samplers are based on the propagation of photons or squeezed states through a multimode interferometer, followed by photon detection [68–71]. Our primary motivation in proposing the GBS-fQRC is that it is a simple QNN, amenable to theoretical and experimental studies, but GBS also appear promising for applications [72–74].

To measure C for a QNN, we need to determine the largest RAM it can emulate. That is, given N distinct

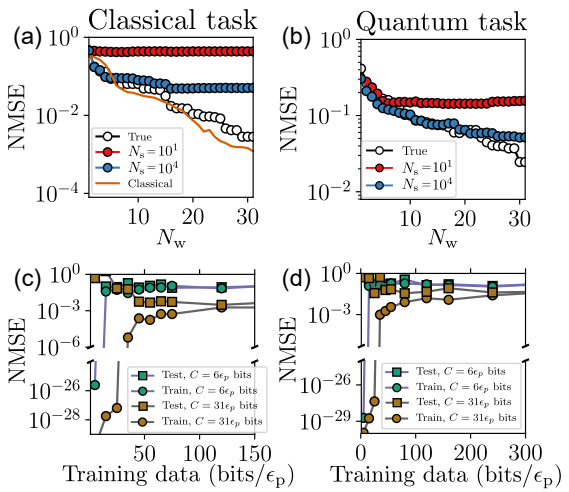

FIG. 3. Performance and generalization of a QNN. (a) Performance of a 5-mode GBS-fQRC in computing a nonlinear function (quantified as normalized-mean-squared-error on a validation data set) for varying number of trained parameters, N_w. N_s is the number of samples used to estimate the quantum variables in $\vec{R}$. The orange solid line shows the same for a classical ELM (for details see SM [53]). (b) Performance on a quantum task of predicting an operator's expectation value on injected quantum states, versus the number of trained parameters. (c) Generalization occurs when training data exceeds C, as can be seen by plotting training and test error versus the amount of data used to train for classical and (d) quantum task. Simulations were performed using the Strawberry Fields and QuTiP Python packages [75, 76].

random inputs distributed uniformly across the relevant input space, the network is trained to classify the N inputs into N distinct, m-bit outputs. If perfect (m-bit precision-limited) classification is achieved for all possible N m-bit labellings, then $C \geq Nm$. Thus, to find C, Nm is increased until training error increases. A systematic measurement of C is prohibitive even for small networks, but we can estimate C by trying a fraction of labellings. Fig. 2 shows the capacity estimate for a 5-mode GBS-fQRC. The estimated capacity linearly increases with the number of trainable parameters, increases with the number of samples used to estimate $\vec{R}$, and decreases when artificial noise is added to the expectation values. Although it can be proven that the fQRC can theoretically achieve $C = W$, we find that simulation precision prevents this in practice [53].

The GBS-fQRC can be trained to perform classical tasks, and its performance depends on C. Fig. 3 shows the normalized mean-squared error (NMSE) on a test data set (i.e., a different set than was used to train) for a 5-mode GBS-fQRC trained to compute the function of classical numbers x_i, $f(x_i) \coloneqq (\sum_{i=1}^{5} x_i)^4 + \sum_{i=1}^{5} x_i^3$, which is an arbitrarily chosen nonlinear function. In Fig. 3a, the NMSE is plotted versus the number of expectation values considered in the output layer (equal to the number of trainable parameters, N_w, here, since the output y is a scalar). As N_w increases, the NMSE on the task drops. Similar behavior is observed for a variety of continuous functions [53]. If, instead of using the "true" expectation values directly from quantum simulations, we estimate them by performing N_s simulated measurements, we observe similar trends, except that the performance improvement saturates at a N_w that depends on N_s. This is a consequence of the reduced capacity due to reduced precision of training and execution. For reference, the continuous curve in Fig. 3a shows the NMSE for a (classical) ELM with a varying number of hidden nodes (which for this task is equal to N_w).

Similar findings are obtained for quantum tasks. As an example, we consider the prediction of the expectation value of an operator acting on injected quantum states. Injected states are randomly selected states $|x\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes |\psi_3\rangle \otimes |0\rangle \otimes |0\rangle$ where each $|\psi_i\rangle$ is randomly selected to be a Fock, cat, coherent or thermal state, with mean photon number between 0 and 1 (or either 1 or 2 for Fock states). After passing $|x\rangle$ through the GBS, $\vec{R}$ is measured, and multiplied by the trained output matrix to produce the prediction. Fig. 3b shows the NMSE for a network trained to predict the absolute value of the operator $\hat{O} = \hat{x}_1^4 \hat{x}_2^2 \hat{x}_3^4$, where $\hat{x}_1 \coloneqq \hat{x} \otimes \hat{1} \otimes \hat{1} \otimes \hat{1} \otimes \hat{1}$ etc., versus N_w. As above, this task is chosen for concreteness; the GBS-fQRC can be trained to predict other expectation values or to perform other quantum tasks [53, 64].

Figures 2 and 3 illustrate the impact of trainable parameter precision. Consider that $W = \sum_{i=1}^{N_w} \log_2(M_i) = \sum_{i=1}^{N_w} \log_2(\max(w_i)/\delta w)$, where δw and $\max(w_i)$ are the precision and maximum possible value of the ith parameter, w_i. For fQRC, $y = W_{\text{out}} \vec{R}$. Thus, if the noise in $\vec{R}$ is $\propto \frac{1}{\sqrt{N_s}}$, then W can be simplified to $W \propto N_w \log_2(N_s)$. Assuming the upper bound $C = W$ and neglecting other precision-limiting effects, this implies that C increases linearly with N_w, but only logarithmically with the number of samples N_s used to estimate $\vec{R}$.

We want to learn models that generalize. Generalization is expected when the training information exceeds C, such that memorization ceases to be the best strategy to minimize training error. Figs 3c-d show the training and test error versus the uncompressed training information, $T = N_T b_T$ bits (where N_T is the number of training points and b_T is the precision of the training y_i values). Once T exceeds C, the test and training errors converge, evidencing that the learned model generalizes over the trained domain [53].

Conclusion: We have introduced an information-theoretic measure constraining the learnable complexity of quantum learning machines, and have shown that it

is bounded by the number of trainable parameters and their trainable precision. Our results apply to QNNs regardless of whether the trained parameters, learning method, or data are quantum or classical. We demonstrated these findings on the example of a Gaussian Boson Sampler-based feed-forward quantum reservoir computer and showed how the need to estimate expectation values (for this particular QNN [53]) results in an exponential decrease in the capacity. While advantages in other metrics (e.g., training speed or performance on specific tasks [53]) are still possible, without quantum training QNNs will have limited advantages over classical NNs in terms of capacity. With quantum training, QNNs seem promising for a range of learning tasks.

We thank Tatsuhiro Onodera, Yihui Quek, Keisuke Fujii, Dmitri Pavlichin, Gerald Friedland, Mario Krell, Brian Coyle, and Ashley Montanaro for helpful comments and discussions. This work was partially supported by the Impulsing Paradigm Change through Disruptive Technologies (ImPACT) Program of the Council of Science, Technology and Innovation (Cabinet Office, Government of Japan). LGW acknowledges additional support from Cornell Neurotech. *Note:* During preparation of this manuscript, we noticed the posting of two related preprints [77, 78].

* lgw32@cornell.edu
† pmcmahon@cornell.edu

[1] T. Menneer and A. Narayanan, Department of Computer Science, University of Exeter, Exeter, United Kingdom, Technical Report **329** (1995).
[2] S. C. Kak, Advances in Imaging and Electron Physics (1995).
[3] E. Behrman, J. Niemel, J. Steck, and S. Skinner, in *Proceedings of the 4th Workshop on Physics of Computation* (1996).
[4] E. Farhi and H. Neven, arXiv:1802.06002.
[5] E. Farhi, J. Goldstone, and S. Gutmann, arXiv:1411.4028.
[6] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, Nature Communications **5**, 4213 (2014).
[7] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, New Journal of Physics **18**, 023023 (2016).
[8] M. Benedetti, E. Lloyd, and S. Sack, arXiv:1906.07682.
[9] M. Schuld, I. Sinayskiy, and F. Petruccione, Quantum Information Processing **13**, 2567 (2016).
[10] K. H. Wan, O. Dahlsten, H. Kristjánsson, R. Gardner, and M. S. Kim, npj Quantum Information **3**, 36 (2017).
[11] P. Rebentrost, T. R. Bromley, C. Weedbrook, and S. Lloyd, Physical Review A **98**, 042308 (2018).
[12] N. Killoran, T. R. Bromley, J. M. Arrazola, M. Schuld, N. Quesada, and S. Lloyd, arXiv:1806.06871.
[13] G. R. Steinbrecher, J. P. Olson, D. Englund, and J. Carolan, npj Quantum Information **5**, 60 (2019).
[14] K. Beer, D. Bondarenko, T. Farrelly, T. J. Osborne, R. Salzmann, and R. Wolf, arXiv:1902.10445.
[15] F. Tacchino, C. Macchiavello, D. Gerace, and D. Bajoni, npj Quantum Information **5**, 26 (2019).
[16] I. Cong, S. Choi, and M. D. Lukin, arXiv:1810.03787.
[17] J. Carolan, M. Mosheni, J. P. Olson, M. Prabhu, C. Chen, D. Bunandar, N. C. Harris, F. N. C. Wong, M. Hochberg, S. Lloyd, and D. Englund, arXiv:1904.10463.
[18] N. Wiebe, A. Kapoor, and K. M. Svore, arXiv:1412.3489.
[19] M. Benedetti, J. Realpe-Gómez, R. Biswas, and A. Perdomo-Ortiz, Physical Review A **94**, 022308 (2016).
[20] G. Verdon, M. Broughton, and J. Biamonte, arXiv:1712.05304.
[21] S. H. Adachi and M. P. Henderson, arXiv:1510.06356.
[22] M. H. Amin, E. Andriyash, J. Rolfe, B. Kulchytskyy, and R. Melko, Physical Review X **8**, 021050 (2018).
[23] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, Physical Review A **98**, 032309 (2018).
[24] C. M. Wilson, J. S. Otterbach, N. Tezak, R. S. Smith, G. E. Crooks, and M. P. da Silva, arXiv:1806.08321.
[25] M. Schuld, A. Bocharov, K. Svore, and N. Wiebe, arXiv:1804.00633.
[26] M. Schuld and N. Killoran, Physical Review Letters **122**, 040504 (2019).
[27] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, Nature **567**, 209 (2019).
[28] D. Ventura and T. Martinez, arXiv:9807053.
[29] N. Wiebe, A. Kapoor, and K. Svore, arXiv:1401.2142.
[30] S. Arunachalam and R. de Wolf, arXiv:1701.06806.
[31] G. Verdon, J. Pye, and M. Broughton, arXiv:1806.09729.
[32] J. D. Biamonte and P. J. Love, Physical Review A **78**, 012352 (2008).
[33] J. Biamonte, arXiv:1903.04500.
[34] J. Preskill, Quantum **2**, 79 (2018).
[35] K. M. Nakanishi, K. Fujii, and S. Todo, arXiv:1903.12166.
[36] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, C. Blank, K. McKiernan, and N. Killoran, arXiv:1811.04968.
[37] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, Physical Review A **99**, 032331 (2019).
[38] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, Nature Communications **9**, 4812 (2018).
[39] R. M. Parrish, E. G. Hohenstein, P. L. McMahon, and T. J. Martinez, arXiv:1906.08728.
[40] V. N. Vapnik and A. Y. Chervonenkis, in *Measures of Complexity* (Springer International Publishing, Cham, 2015) pp. 11–30.
[41] D. J. C. Mackay, *Information Theory, Inference, and Learning Algorithms* (Cambridge University Press, UK, 2005).
[42] N. Tishby and N. Zaslavsky, arXiv:1503.02406.
[43] R. Shwartz-Ziv and N. Tishby, arXiv:1703.00810.
[44] G. Friedland and M. Krell, arXiv:1708.06019.
[45] P. Baldi and R. Vershynin, Advances in Neural Information Processing Systems, 7729-7738 (2018).
[46] G. Friedland, A. Metere, and M. Krell, arXiv:1810.02328.
[47] P. Baldi and R. Vershynin, arXiv:1901.00434.
[48] T. M. Cover, IEEE Transactions on Electronic Computers **3**, 326 (1965).
[49] E. Gardner, Europhysics Letters **4**, 481 (1987).
[50] W. Kinzel, Philosophical Magazine B **77**, 1455 (1998).
[51] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms* (Cambridge

University Press, 2013).
[52] V. N. Vapnik, *The nature of statistical learning theory* (Springer, 2000).
[53] See the attached Supplemental Material.
[54] A. V. Nayak, Ph.D. thesis, University of California, Berkeley (1999).
[55] A. S. Holevo, Problems of Information Transmission **9**, 177-183 (1973).
[56] S. Lloyd and C. Weedbrook, Physical Review Letters **121**, 040502 (2018).
[57] X. Gao, Z.-Y. Zhang, and L.-M Duan, Science Advances **4**, eeat9004 (2018).
[58] B. Coyle, D. Mills, V. Danos, and E. Kashefi, arXiv:1904.02214.
[59] G. B. Huang, Q. Y. Zhu, and C. K. Siew, in *IEEE International Conference on Neural Networks - Conference Proceedings* (2004).
[60] K. Fujii and K. Nakajima, Physical Review Applied **8**, 024030 (2017).
[61] M. Negoro, K. Mitarai, K. Fujii, K. Nakajima, and M. Kitagawa, arXiv:1806.10910.
[62] K. Nakajima, K. Fujii, M. Negoro, K. Mitarai, and M. Kitagawa, arXiv:1803.04574.
[63] A. Kutvonen, T. Sagawa, and K. Fujii, arXiv:1807.03947.
[64] S. Ghosh, A. Opala, M. Matuszewski, T. Paterek, and T. C. H. Liew, npj Quantum Information **5**, 35 (2019).
[65] J. Chen and H. I. Nurdin, arXiv:1901.01653.
[66] H. Jaeger, in *GMD-Forschungszentrum Informationstechnik Report 148* (2001).
[67] W. Maass, T. Natschläger, and H. Markram, Neural Computation **14**, 2531 (2002).
[68] S. Aaronson and A. Arkhipov, in *Proceedings of the 43rd annual ACM symposium on Theory of computing - STOC '11* (ACM Press, New York, New York, USA, 2011) p. 333.
[69] A. Lund, A. Laing, S. Rahimi-Keshari, T. Rudolph, J. O'Brien, and T. Ralph, Physical Review Letters **113**, 100502 (2014).
[70] C. S. Hamilton, R. Kruse, L. Sansoni, S. Barkhofen, C. Silberhorn, and I. Jex, Physical Review Letters **119**, 170501 (2017).
[71] J. Huh, G. G. Guerreschi, B. Peropadre, J. R. McClean, and A. Aspuru-Guzik, Nature Photonics **9**, 615 (2015).
[72] K. Brádler, P.-L. Dallaire-Demers, P. Rebentrost, D. Su, and C. Weedbrook, Physical Review A **98**, 032310 (2018).
[73] J. M. Arrazola, T. R. Bromley, and P. Rebentrost, Physical Review A **98**, 012322 (2018).
[74] M. Schuld, K. Brádler, R. Israel, D. Su, and B. Gupt, arXiv:1905.12646.
[75] N. Killoran, J. Izaac, N. Quesada, V. Bergholm, M. Amy, and C. Weedbrook, arXiv:1804.03159.
[76] J. Johansson, P. Nation, and F. Nori, Computer Physics Communications **184**, 1234 (2013).
[77] L. Banchi and S. Pirandola, in *Quantum Information and Measurement (QIM) V: Quantum Technologies* (OSA, Washington, D.C., 2019) p. S4B.5.
[78] E. Dolzhkov, B. Ghandchi, and D. O. Theis, arXiv:1903.12611.
[79] P. E. Latham and Y. Roudi, Scholarpedia **4**, 1658 (2009).
[80] N. J. Beaudry and R. Renner, Quantum Inf. Comput. **12**, 0432-0441 (2012).

The Capacity of Quantum Neural Networks: Supplemental Material

Logan G. Wright[1,2] and Peter L. McMahon[1,2]
1. School of Applied and Engineering Physics, Cornell University, Ithaca, NY 14853, USA and
2. E. L. Ginzton Laboratory, Stanford University, Stanford, CA 94305, USA *

I. CAPACITY OF QNNS FOR SPECIFIC SPECIAL CASES

A. Single-use capacity and relationship to Holevo's and Nayak's bounds

Consider an execution protocol for a QNN in which, given an input, the QNN circuit may be run only once. In this case, C will be limited by Nayak's bound [1] (itself a consequence of Holevo's bound [2]). For example, in this scenario, an N-qubit QNN learned parameter can only provide at most N additional *bits* of capacity. However, if exponentially many runs of the QNN are allowed, then an N-qubit QNN learned parameter may provide up to exponentially many (in N) additional bits of capacity.

B. Capacity for QNNs producing quantum outputs

As discussed in the main article, for a QNN producing a quantum output (Supplementary Figure 1b), $\hat{\rho}$, the capacity can be determined similarly to the case where the QNN produces a classical output (Supplementary Figure 1a). The QNN is trained to act as a RAM, where information is stored and retrieved from the elements of the density operators produced by the QNN. To measure C, the QNN is trained to map N distinct general-position inputs to N "label states". To determine the information content of the label states, one must measure each label density matrix ρ to an arbitrary precision (such that the precision in the estimated values of each independent degree of freedom of the density matrix is dominated by the precision of the QNN). The capacity C for the QNN producing quantum outputs is then (in units of bits here),

$$C_{\mathrm{Q}} \coloneqq \sum_{i=1}^{N} \sum_{j=1}^{D} \log_2 e_{ij}^{-1}, \tag{1}$$

where e_{ij} is the absolute normalized mean error in the QNN's reproduction of the i-th label state's j-th degree of freedom (recall e.g., a density matrix in a D-dimensional Hilbert space has $D^2 - 1$ real degrees of freedom, due to the constraints of Hermiticity and normalization). C_{Q} in Eqn. 1 above is the equivalent of C for classical tasks for QNN's producing quantum outputs, and is expressed here in the same units, but for clarity in subsequent sections of the document, we denote it specifically by C_{Q}.

C. Generative quantum neural networks

Although our interest in the main article is on QNNs performing classification-like tasks, a very similar approach, albeit with important distinctions, can be used for generative tasks. For generative tasks, one is sampling from a probability distribution for the results of measurements on a state (or distribution of states) the QNN is trained to produce (Supplementary Figure 1c). Note that any measurement protocol may be used, including one in which multiple sequential measurements are made of different observables, and/or in which the QNN is run multiple times and a single sample comprises a vector of measurement results across several runs. Importantly, although the goal of the generative learning machine is in some sense to model a distribution, a valid output of the machine contains only an exponentially small fraction of the information needed to describe that distribution.

A capacity-like measure, the generative capacity C_{G} for generative tasks, can then be defined similarly to C in the main article, except we consider the task of mapping a single trivial input ($N = 1$, a vacuum input in Supplementary

* lgw32@cornell.edu; pmcmahon@cornell.edu

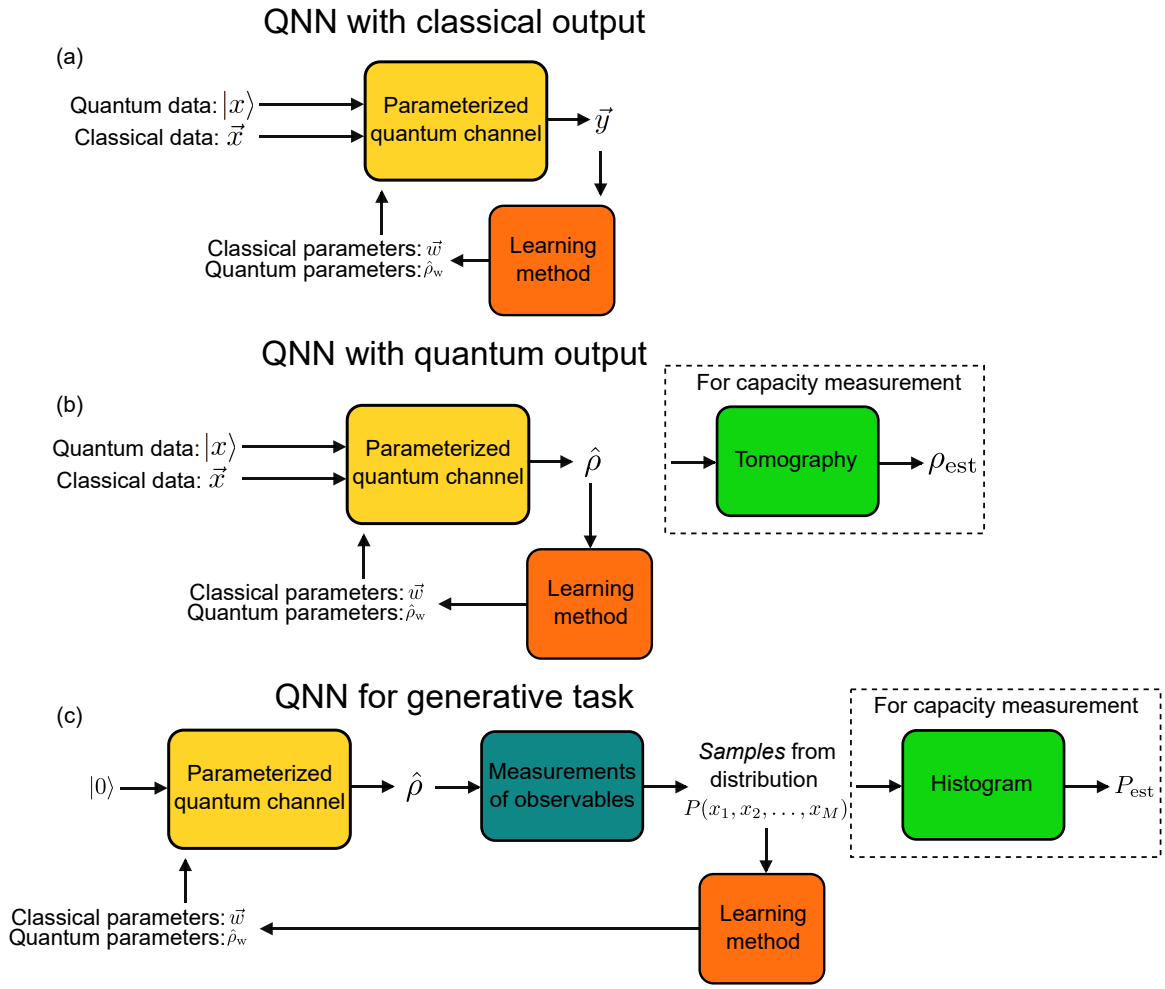

FIG. 1. Outline of QNNs performing different tasks. (a) QNN producing a classical output, $\vec{y}$. For this case, C from the main article directly applies. (b) QNN producing a quantum output, $\hat{\rho}$. As discussed in the main article and elaborated here, for a QNN producing a quantum output the capacity can be determined through a procedure similar to the case of a classical output (see main text). (c) For generative tasks, a QNN produces a quantum state, on which a specific set/sequence of measurements is performed, producing a sample from a distribution. A learning method is used to optimize the parameters of the QNN so that the distribution matches a desired one implied by training data. We can quantify the range of complexity of distributions the generative QNN can produce using a similar approach as applied to the QNNs in (a) and (b).

Figure 1c) to all possible output distributions of the machine. That is, C_{G} is the information that can be, through training, imparted into the *distribution* produced by the machine. This is defined as:

$$C_{\text{G}} \coloneqq \sum_i \log_2 f_i^{-1}, \tag{2}$$

where f_i is the normalized absolute mean error in the reproduction of the i-th degree of freedom of the output distribution, averaged over all possible output distributions. For example, a distribution in which M observables are measured, $P(x_1, \ldots, x_M)$, and in which each observable has L possible outcomes, has $L^M - 1$ degrees of freedom. That is, each possible outcome has an associated degree of freedom that is the probability of that outcome (e.g., $P(1,1,1)$ and $P(1,2,1)$ each correspond to continuous degrees of freedom of the distribution $P(x_1, x_2, x_3)$.) Of course, this definition applies equally well to any nature of generative learning machine.

The capacity of a generative QNN has an intuitive relationship with the capacity for quantum outputs defined above. As Supplementary Figure 1c shows, the generative QNN produces, before producing its ultimate classical output, a sample from the intended distribution, a quantum state. We could, instead of performing measurements to produce the generative samples, instead quantify the capacity for quantum outputs as described above, $C_{\text{Q}}(1) = \sum_{j=1}^{D} \log_2 e_j^{-1}$ (modified from Eqn. 1 since $N = 1$ for the generative QNN here). The generative capacity of a given learning machine

is bounded by the machine's capacity for producing quantum outputs and W of the machine in the following ways:

$$C_{\mathrm{G}} \leq C_{\mathrm{Q}}(1) \leq W, \tag{3}$$

where $C_{\mathrm{Q}}(1)$ is the capacity of the generative QNN for producing quantum outputs, prior to the measurements that produce its final classical output. W here is defined as in the main article: it is the amount of information that can be stored through training in the trainable parameters of the learning machine. This order of bounds is easily appreciated. The capacity for quantum outputs, C_{Q}, is bounded by W through the pigeonhole principle. Any distribution produced by the machine can be computed from the full information of the state it produces, $\hat{\rho}$ and so from the data-processing inequality [3, 4], the information required to specify the produced probability distribution cannot exceed that of the state that was used to generate it, thus $C_{\mathrm{G}} \leq C_{\mathrm{Q}}(1)$.

It is important to see that this complexity measure, C_{G} is not directly comparable to C. Since the purpose of the generative QNN is not to produce deterministically an input-output relationship, but to provide samples from a possibly complex distribution, the above complexity quantification C_{G} is appropriate, since it is a measure of the range of complexity of tasks the generative QNN can perform. It is instructive to consider C for the generative QNN nonetheless. To directly apply C as introduced in the main article would require that we know the number and nature of measurements used in the generative QNN's execution. However, in virtually all reasonable execution protocols for a generative QNN, one is only obtaining an exponentially-small fraction of the information needed to describe the output state (or distribution of states) produced by the QNN. Consequently, C for the generative QNN is exponentially smaller than C_{G}.

A first conclusion from the above analysis is that generative QNNs do not possess more capacity, C, or generative capacity, C_{G}, than classical generative NNs for a given fixed number of equivalent classical parameters. Similar to the case of classifier QNNs, this does not imply that generative QNNs cannot provide quantum advantages for specific sampling tasks – many examples, such as Boson Sampling, are known to be classically hard. A low-C_{G} generative QNN could – given suitable initialization – perform Boson Sampling; it just would not be effective in learning a large range of other distributions. Like classifier QNNs, the exponential classical-equivalent information of quantum parameters implies that quantum-parameterized, quantum-trained generative QNNs may provide a C_{G}-advantage over classical generative QNNs.

The distinction between their intended use and the meaning of C_{G} and C presents an important apparent distinction between generative QNNs and classifier QNNs of the format considered in the main article, in which estimated expectation values are used to provide a nominally-deterministic classification result. In the case of a classifier QNN outputting expectation values, we found a quantum-classical information bottleneck due to the probabilistic nature of the QNN's outputs and the intended deterministic quality of the QNN output. In contrast, generative QNNs avoid this exponential cost because outputs are probabilistic by nature and intent alike, and so a given random sample is a viable and accurate output (assuming ideal measurements). The generative QNN's C is, of course, exponentially small, but the task it is used for benefits from the complexity it can provide without requiring large C.

D. Implications of exponential capacity

The generative case is an instructive example on the practical meaning of the exponential complexity and capacity of QNNs relative to classical NNs. Quantum parameters provide exponentially more complexity for a model than classical parameters; intuitively to represent fully a quantum-state-parameter of N qubits one needs exponentially more classical information than to describe a classical parameter of N bits. As a result, quantum-parameterized QNNs may realize exponentially larger W and, if suitably designed, exponentially larger C compared to classical NNs. As there is some potential for quantum training algorithms to allow the training of these (in effect) exponentially many parameters faster than would be possible for a classical NN with equal W, there are numerous exciting possible advantages of QNNs for arbitrary machine-learning tasks.

That said, making *direct* use of this exponential capacity advantage for deterministic (non-generative) classical tasks will come at an exponential cost in runtime. A practical interpretation of the benefit one may get from the exponentially larger capacity then is similar to the interpretation of how the HHL algorithm can be used beneficially despite also being subject to a readout bottleneck: a quantum-trained and parameterized QNN may learn and execute exponentially complex models, but to read out deterministic classical answers quickly one must only extract from the device a summary or compressed version of the full answer. QNNs providing an advantage for classical tasks would thus be ones in which an information compressing output layer (or layers) is used prior to classical read-out. This architectural measure erases the capacity advantage of the QNN (we are forcing the QNN to output a compressed quantity, so clearly its RAM-equivalent functionality and the input-output complexity of its models are reduced). However, as with the HHL algorithm it preserves the potential speed advantage of the QNN in performing a hard

computation. In a sense, this is how generative QNNs avoid the exponential cost of exponential capacity: they are outputting an exponentially-compressed version of the what they actually produce (a state or distribution of states).

For QNNs producing quantum outputs, the QNN's exponential capacity advantage over classical is necessary if one intends for the QNN to perform arbitrary quantum operations. As quantum data provides exponentially more equivalent information than classical data, training of QNNs for quantum tasks should be manageable despite the enormous capacity. Moreover, even capacity-limited QNNs offer a natural advantage over classical NNs for operations on quantum data, since using a QNN avoids needing to first perform tomography to determine the classical representation of the input quantum state. Likewise, QNNs producing quantum outputs naturally avoid the quantum-classical bottleneck that applies to QNNs that need to produce a classical deterministic result.

II. DETAILED DESCRIPTION OF THE GBS-FQRC

Our GBS-fQRC reservoir circuit consists of M input squeezers, followed by a random M-mode interferometer and finally, measurement in the Fock basis. Quantum data is input as states directly into the circuit. We input classical data, x_i, to the device by encoding it in the squeezing parameter of the squeezed-vacuum states (i.e., the ith squeezer is realized with an operator $\hat{S}_i \propto \exp[x_i(\hat{a}_i^2 - (\hat{a}_i^\dagger)^2)]$). The choice of a nonlinear embedding, rather than state-amplitude encoding, is an important part of transforming the input classical data into the reservoir circuit's high-dimensional Hilbert space. We consider a quantum-classical hybrid approach in which training and the final layer of the QNN is implemented with a classical computer. The output of the GBS-fQRC reservoir circuit is chosen to be a vector of photon-detection expectation values, including coincidences. There are $2^M - 1$ such expectation values. For the 5-mode device we consider throughout, the reservoir output vector is $\vec{R} = [\langle\hat{n}_1\rangle,\langle\hat{n}_2\rangle,...,\langle\hat{n}_1\hat{n}_2\rangle,...,\langle\hat{n}_1\hat{n}_2\hat{n}_3\hat{n}_4\hat{n}_5\rangle]$. To vary N_w, we truncate $\vec{R}$ to the first $N_\mathrm{w} \leq 2^5 - 1$ elements. For execution of the trained QNN, this vector is multiplied by a trained output-layer matrix W_out, which contains all the trainable parameters of the fQRC, to produce the computation result. Training the fQRC requires collecting the output vectors $\vec{R}$ across a training set, consisting of input-output pairs $\vec{x}_\mathrm{T}$ and $\vec{y}_\mathrm{T}$. The reservoir output vectors are assembled into a matrix R_out, and the optimum W_out is learned simply by $W_\mathrm{out} = R_\mathrm{out}^+ \vec{y}_\mathrm{T}$, where R_out^+ is the Moore-Penrose pseudo-inverse of R_out.

In our simulations of GBS-fQRC presented here, we considered Fock states up to $|3\rangle$. To ensure that states remained consistently within this cutoff, we did not include any displacement of the input states, and allowed for a maximum squeezing of 2.6 dB for any input. The interferometers were all set to implement Haar-random unitaries, where for each task including training and testing and varying of parameters, we considered a fixed interferometer. We did not observe significant changes in results when different random unitaries were chosen (from the set of Haar-random unitaries).

III. ACHIEVING THE CAPACITY UPPER BOUND WITH FQRC

One reason we have chosen fQRC as our example system for numerically demonstrating our results is that the simple training permits us to easily show that C may reach the maximum value. We are interested in the case where the exact mapping of N distinct inputs each to one of N distinct output m-bit numbers can be learned. In this case, we require that $W_\mathrm{out} = R_\mathrm{out}^{-1}\vec{y}_\mathrm{T}$ have an exact solution. If N_w is the number of m-bit outputs of the reservoir, then R_out is an N_T-by-N_w matrix, where N_T is the number of training input-output pairs. $\vec{y}_\mathrm{T}$ is a N_T-element vector. If the maximum number of *distinct* input-output pairs, N, is less than N_T, then all but N of the N_T outputs can be assigned zero weight (ignored), such that finding R_out^{-1} reduces to finding the inverse of a square N-by-N matrix. If, for $N \leq N_\mathrm{w}$, the rank of R_out is at least N, then an exact solution for W_out exists. Provided this can be true for $N = N_\mathrm{w}$, and that W_out consists of N_w m-bit numbers (i.e., $W = N_\mathrm{w}m$) then the fQRC's capacity can maximal, that is: $C = N_\mathrm{w}m = W$.

In other words, to achieve maximum capacity with respect to W with fQRC, the requirements are that the parameters can be trained to at least m-bit precision, and that the independent features that make up the reservoir-output vector $\vec{R}$ (here different expectation values) correspond to linearly independent transformations of the input data over the training data set. For the choice of $\vec{R}$ considered in the main article and the specific GBS device details we chose, reaching high precision requires impractically many samples but is possible in principle. Provided all the chosen expectation values have finite values, they correspond to linearly independent transformations of the input (since any of the chosen operators cannot be written as a linear combination of the others).

IV. ESTIMATION OF CAPACITY

To estimate the capacity of neural networks we employed two approaches. In both approaches a training data set is constructed as follows. N distinct random points distributed uniformly over the input space of the neural network were used as inputs, $\vec{x}_{\mathrm{T}}$. The intended output for each input was then assigned to be one of N random, uniformly distributed m-bit numbers. In practice, here we exclusively use for our m-bit numbers Python double-precision numbers, since these are the values consistent with squeezing parameters and expectation values in our quantum simulations.

To produce the results here, we computed the mean training error across many sets of outputs. To determine with certainty a QNN's capacity C, one needs to ensure that all possible N, m-bit labellings can be learned. This is impractical for even small numbers: for our simple 5-mode GBS-fQRC the number of labellings is $\approx 2^{(2^5-1)\times 45}$, where 2^5-1 is the number of distinct expectation values (the number of trainable parameters if the GBS-fQRC is used to produce scalar outputs as it is here), and 45 bits is the effective precision of these trainable parameters in our simulations (see later in this section for more details). Thus, we instead choose to merely estimate C by considering a small number, typically 10-100, of randomly chosen labellings, as little difference was observed with larger sample sizes. We also confirmed that similar results could be obtained for unusual labellings, such as if the intended output N m-bit number labels were all chosen to be identical, or when all but one were identical.

As a baseline test of this capacity-estimation procedure, we first applied it to classical extreme learning machines (ELMs), which are single-hidden-layer neural networks with randomly initialized parameters, trained with linear regression on the linear output layer. The classical ELMs considered here (and in the main article) have the following design. The networks had 5 input nodes, N_{w} hidden nodes, and 1 output node. Input data was normalized to ± 1 and hyperbolic tangent activation was used with zero bias. An ELM has a linear input layer, in which inputs are distributed to the hidden layer by multiplying the input vector by a random matrix, W_{in}. The matrix elements of the input layer to the ELMs were chosen from a uniform random distribution, and normalized to 1 by default. For evaluating performance of the ELM on various functions, we additionally varied the normalization by multiplying W_{in} by adjusting a scalar multiplier ρ (see later section of this document).

To assess the effect of noise in the read-out of the hidden layer (similar to the estimation-associated noise in the GBS-fQRC or other QNN), we added mean-zero noise from a uniform distribution with varying peak amplitude each time the activation of the hidden layer was calculated (i.e., the noise added to the output of the hidden layer every time it was executed, both in training and testing, was different). We also used this test to determine the numerical-noise-related precision of our calculations.

To estimate C we first calculated the absolute normalized error after training, $\epsilon(N) = \mathrm{mean}[\frac{|y_i - y_{ti}|}{|y_{ti}|}]$, where y_i is the NN output after training for N inputs/labels, y_{ti} is the intended training output (the N labels), and the mean is taken over both the individual training set and all the random labellings considered. From this, we then determined C as

$$C = \max[N \log_2(1/\epsilon(N))], \tag{4}$$

where the maximum is taken over different values of N. When this is done, we obtain the results shown in Supplementary Figure 2.

Supplementary Figure 2a shows that, while there is typically a linear or sub-linear growth of C with more trainable parameters, even with zero artificial additive noise none of the tested neural networks reach the maximum possible capacity. Supplementary Figure 2b clarifies this finding as resulting from the limited numerical precision of our simulations. We see that, for artificial additive noise up to about 10^{-14}, no change in the estimated capacity occurs. This shows that the calculations (including training, and thus W) are limited by a numerical-precision noise to some finite precision we denote ϵ_{p} (measured in units of bits); the source of this noise is in general a combination of the finite-precision representation of numbers presumed real by some of the calculations, and the propagated error from those calculations. Based on the curves in Supplementary Figures 2b and 3b, we estimate ϵ_{p} to be $\log_2(1/10^{-14}) \approx 46.5$ bits. Once the additive noise is larger than this intrinsic noise, we see the expected logarithmic decay of the capacity (approximately linear on the log-x axis of Supplementary Figure 2b), until at around 10^{-8}, zero capacity is reached. This point corresponds to the point of zero signal-to-noise ratio in training of the parameters.

The previous results show that our estimate of C, via the direct Eqn. 4, has the feature of being sensitive to the precision of the calculations. This is correct: C itself depends on the precision of the calculations that define the learning machine's training and execution. However, here we are interested in evaluating C due to the physics of a simulated machine, not due the finite precision of the simulation. In the interest of concentrating on the physical aspects of the network that affect C, we chose to also measure C relative to the numerical precision ϵ_{p}. To do this, we simply take $C = \max[N] \times \epsilon_{\mathrm{p}}$ for networks where the mean error is smaller than 10^{-10} (a normalized mean squared error of 10^{-20}). Supplementary Figure 3 shows the result of applying this policy for increasing added noise amplitude.

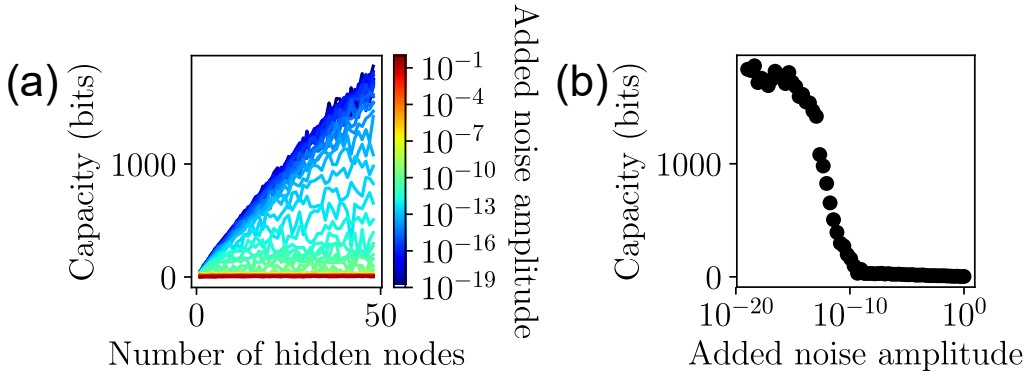

FIG. 2. Direct estimation of the capacity. (a) Estimated capacity for a classical single hidden layer neural network (extreme learning machine). Different curves correspond to different additive noise levels. (b) The estimated capacity (for 48 hidden nodes/trainable parameters) versus the additive noise amplitude.

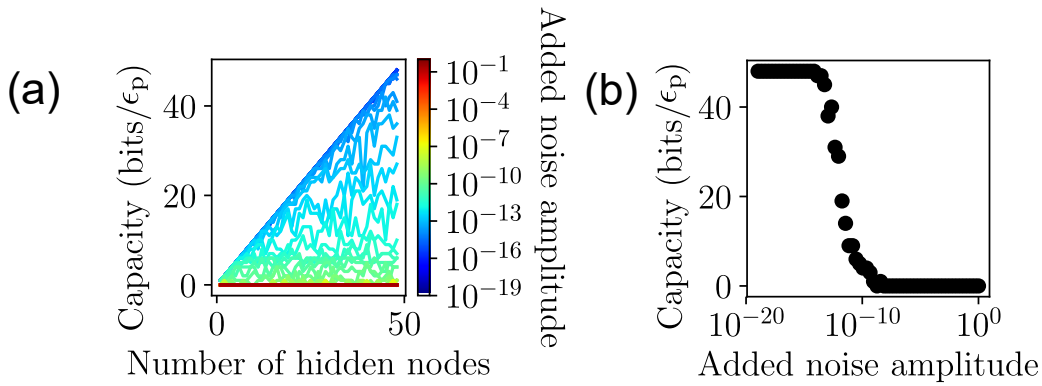

FIG. 3. Numerical-noise-normalized capacity estimate. (a) Estimated capacity for a classical single hidden layer neural network (extreme learning machine) in units of simulation precision ϵ_p. Different curves correspond to different additive noise levels. (b) The estimated capacity (for 48 hidden nodes/trainable parameters) versus the additive noise amplitude.

In the main article the latter policy is used to estimate the capacity of the GBS-fQRC when the exact expectation values are used in $\vec{R}$, in order to present the physics-limited capacity (within the assumption of arbitrary-precision in measuring the expectation values) rather than one limited by the precision of our simulations. However, Eqn. 4 and the associated procedure are the most general approach for estimating C, and are used otherwise.

V. OTHER EXAMPLES OF CLASSICAL AND QUANTUM TASKS

As mentioned in the main article, we see similar trends for a variety of quantum and classical continuous-function-approximation tasks. Supplementary Figure 4 shows a collection of these tasks. In each case, we consider a 5-mode GBS with 5 inputs. For training data, the intended function (operator-expectation-value, in the quantum case) was calculated for 500 random sets of x_i (classical) or 500 random states (quantum). In the quantum case, input states were of the form $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |0\rangle$ where each $|\psi_i\rangle$ is randomly selected to be a Fock, cat, coherent, or thermal

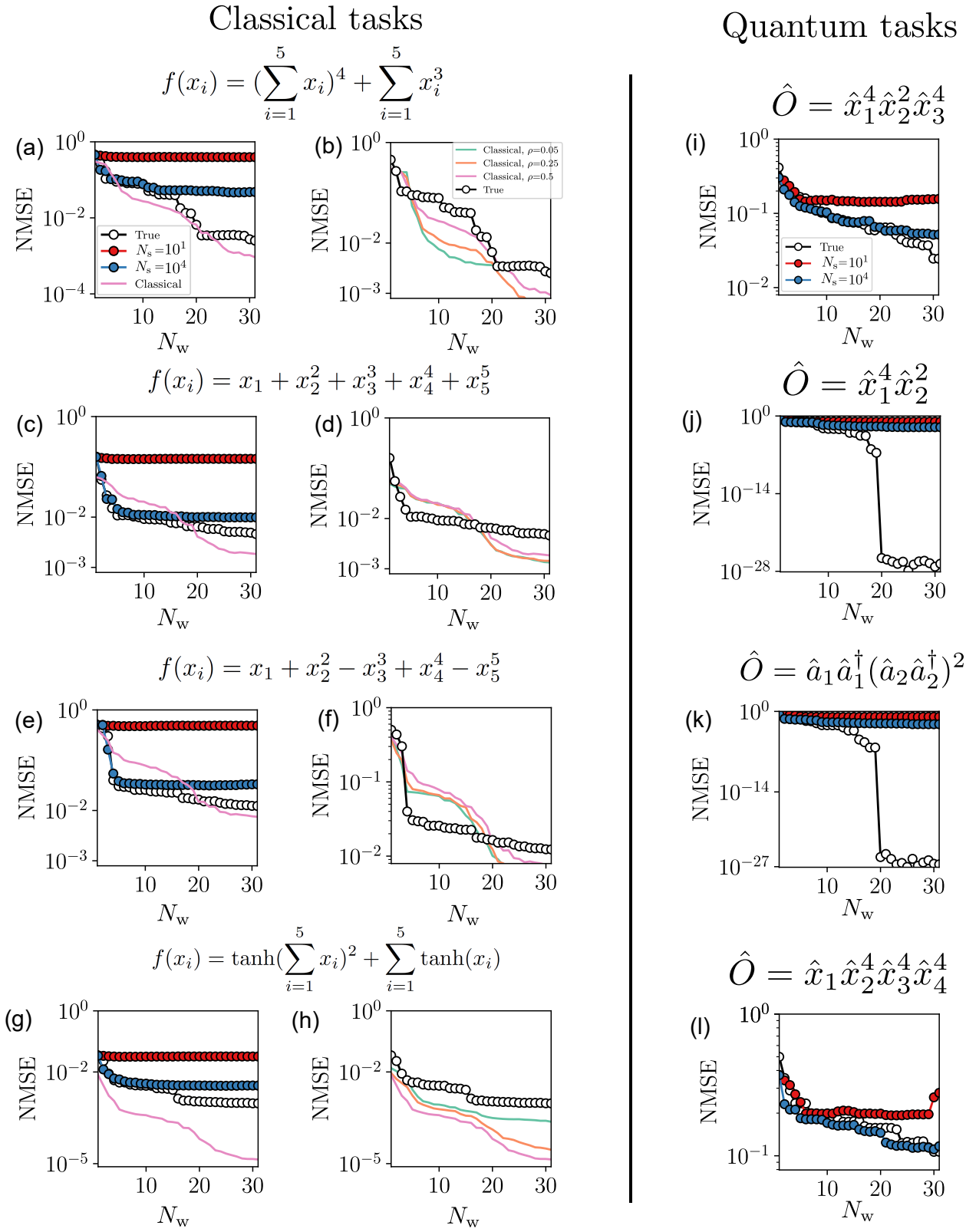

FIG. 4. Additional examples of classical and quantum function approximation. For details see text.

state, with a randomly selected mean photon number between 0 and 1 (or equal to 1 or 2 for the Fock state). The number of non-vacuum states for each task was determined by the number of different modes in the operator (e.g., in Supplementary Fig 4i, 3 non-vacuum states were combined with 2 vacuum states to form the input state $|x\rangle$). For both types of tasks, the performance was measured on an independent verification (i.e., test) set of 500 independently drawn random inputs, where the normalized mean squared error (NMSE) was calculated with respect to the correct function (operator expectation) value. As is typical for machine learning, the test data is drawn from the same range of values as the training data (i.e., the functions are learned over a finite domain, and generalization ability is assessed only over the same domain). For the classical tasks, we compare the fQRC with a classical extreme learning machine. This classical ELM has a hyperparameter in its tanh activation function, ρ, that scales the inputs, i.e., $\vec{R}_{\text{classical}} = \tanh(\rho W_{\text{in}}\vec{x})$, where W_{in} is the random linear input-layer matrix, and $\vec{x}$ is the vector of input values. For completeness, the second column of Supplementary Figure 4 shows the classical NMSE for different values of this hyperparameter. Although the performance in the classical case can vary strongly with this parameter, we generally find that the performance of the classical and quantum NNs with true expectation values are similar when they have a similar number of parameters. Quantum NNs with estimated expectation values show worse performance than classical ELMs with similarly many parameters.

In general, for most tasks we consider, the same trends reported in the main article are observed: better performance is obtained with more trainable parameters and with more samples used to estimate expectation values in $\vec{R}$. For completeness however, we note that these are merely overall trends. We sometimes notice that particular features (expectation values in $\vec{R}$) lead to dramatically improved performance (e.g., Supplementary Figure 4j). This is to be expected when certain features are strongly correlated with the intended function. In addition, when the complexity of tasks is too high, the learned model is sufficiently far from the intended function that we often see unexpected trends, such as monotonic increases in performance with larger numbers of parameters N_{w} or samples N_{s} (e.g., Supplementary Figure 4l).

A. Definition of NMSE

The definition of the normalized mean squared error we use is:

$$\text{NMSE} \coloneqq \frac{1}{N}\sum_{i=1}^{N}\frac{(y_i - y_{\text{v},i})^2}{(y_{\text{v},i})^2}, \tag{5}$$

where N is the number of samples in the validation test set (or the training set, if the NMSE refers to training error), and $y_{\text{v},i}$ are the correct output values in the validation test (training) set.

VI. MORE ON GENERALIZATION AND CAPACITY

In the main article, we consider the generalization performance as a function of training data within a fixed range of input values. To better visualize the trade-offs between capacity and generalization, here we consider two 1-dimensional functions, and examine the effect of capacity on generalization *outside* the range of inputs used for training (Supplementary Figure 5).

VII. SUPPLEMENTARY REFERENCES

[1] A. V. Nayak, Ph.D. thesis, University of California, Berkeley (1999)
[2] A. S. Holevo, Problems of Information Transmission **9**, 177-183 (1973).
[3] P. E. Latham and Y. Roudi, Scholarpedia **4**, 1658 (2009).
[4] N. J. Beaudry and R. Renner, Quantum Inf. Comput. **12**, 0432-0441 (2012).

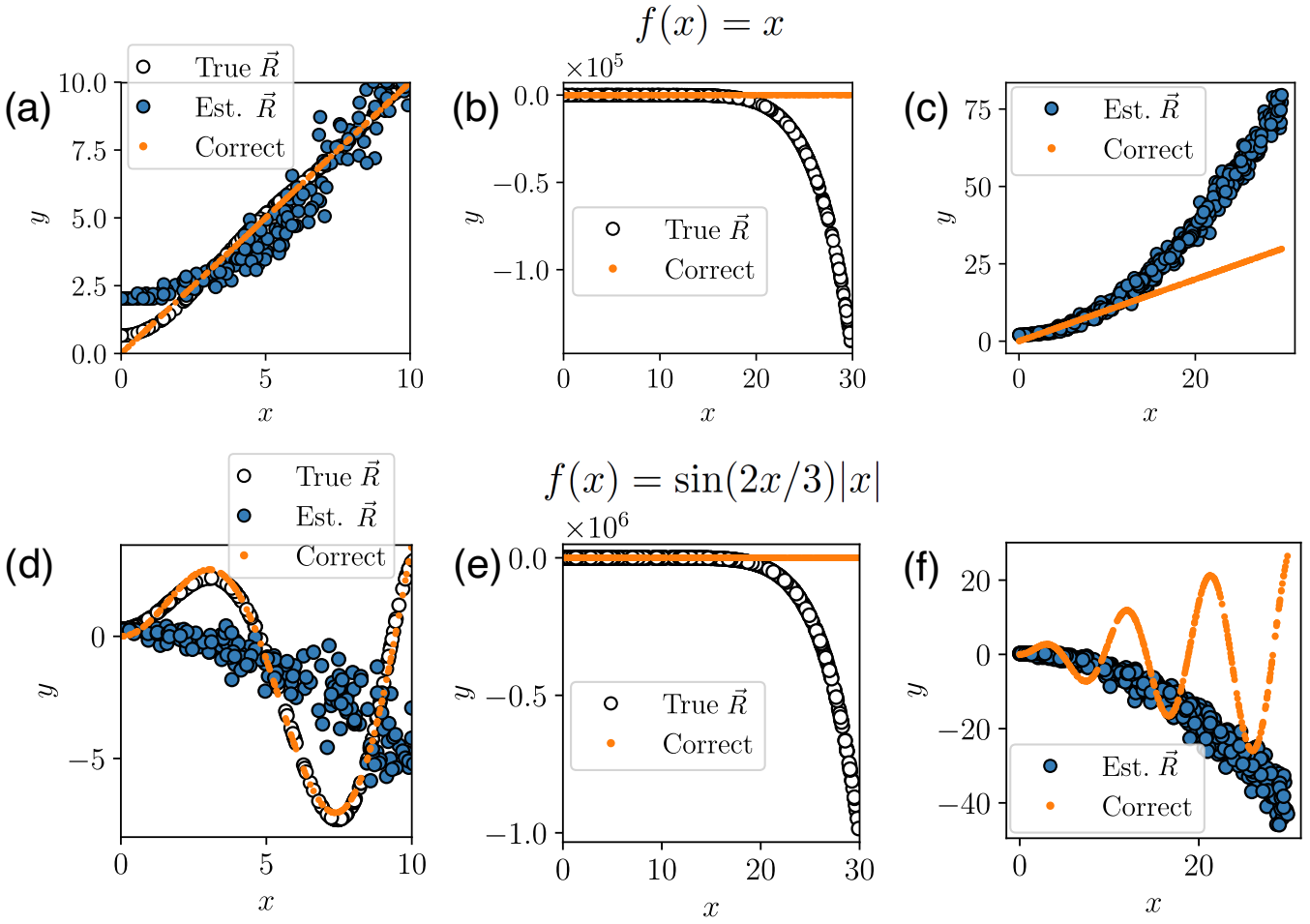

FIG. 5. Test performance of a high-capacity QNN (GBS-fQRC using exact simulated expectation values) and a low-capacity QNN (GBS-fQRC using estimated expectation values) on 1-dimensional function approximation. For both the simple linear function and the more complex sinusoidal function, significantly better performance is achieved by the high-capacity model within the training data range, $x \in [0, 10]$. For the complex function, the difference in performance is more noticeable. However, for testing with x outside the range $[0, 10]$, the high-capacity model's predictions diverge much more quickly from the correct answers than do the low-capacity model's ones. The experiment was set up such that, in both cases, the training data significantly exceeds the capacity. The panels from left (a,d) to right (c,f) show the test performance over the range of the training set (a,d), the test error over the extended test range for the case where exact expectation values are used (b,e), and when the estimated expectation values are used (c,f).