KI - Künstliche Intelligenz (2024) 38:379–385
https://doi.org/10.1007/s13218-024-00864-7

SYSTEMS DESCRIPTION

Benchmarking Quantum Generative Learning: A Study on Scalability and Noise Resilience using QUARK

Florian J. Kiwit[1,2] · **Maximilian A. Wolf**[1,2] · **Marwa Marso**[1,2,3] · **Philipp Ross**[2] · **Jeanette M. Lorenz**[1,4] · **Carlos A. Riofrío**[2] · **Andre Luckow**[1,2]

Received: 1 February 2024 / Accepted: 10 July 2024 / Published online: 19 August 2024

Abstract
Quantum computing promises a disruptive impact on machine learning algorithms, taking advantage of the exponentially large Hilbert space available. However, it is not clear how to scale quantum machine learning (QML) to industrial-level applications. This paper investigates the scalability and noise resilience of quantum generative learning applications. We consider the training performance in the presence of statistical noise due to finite-shot noise statistics and quantum noise due to decoherence to analyze the scalability of QML methods. We employ rigorous benchmarking techniques to track progress and identify challenges in scaling QML algorithms, and show how characterization of QML systems can be accelerated, simplified, and made reproducible when the QUARK framework is used. We show that QGANs are not as affected by the curse of dimensionality as QCBMs and to which extent QCBMs are resilient to noise.

Keywords Quantum computing · Machine learning · Noise resilience · Generative modeling · Benchmark framework

1 Introduction

Systematic evaluation of quantum processors and algorithms through benchmarking offers valuable insights into the current capabilities and future potential of available quantum processing units [1–9]. However, benchmarking quantum computing is far from a straightforward task. The field is characterized by a diversity of technologies [10], each with unique requirements for precise and meaningful assessment. As a result, current benchmarks often focus on specific aspects of the technology, which can sometimes lead to an incomplete picture of the end-to-end performance of quantum computing.

The Quantum computing Application benchmark (QUARK) framework [8] was explicitly developed for challenges of application-oriented quantum computing. QUARK's benchmarking approach ensures a comprehensive evaluation, covering the entire benchmarking pipeline from hardware to algorithmic design for the problems under investigation. Its versatility and modular implementation are central to QUARK, allowing for component expansion and customization. Additionally, it hosts benchmarks from the domain of optimization [8] and machine learning [11].

In quantum machine learning (QML), scaling algorithms and maintaining performance amidst noise are crucial for practical applications, particularly in industries reliant on generative models. This work shows an extension that enables us to include noisy simulations for QML applications. It is important to understand the limitations and track the development of current quantum hardware and algorithms over time. Concretely, we present a comprehensive study of the scalability of QML models, evaluating their intrinsic robustness against statistical and quantum noise. We aim to bridge the gap between theoretical QML advancements and their practical implementation in real-world scenarios while using the QUARK framework to accelerate and standardize performance assessment.

✉ Florian J. Kiwit
f.kiwit@campus.lmu.de

[1] Ludwig Maximilian Universität, Munich, Germany

[2] BMW Group, Munich, Germany

[3] Technical University, Munich, Germany

[4] Fraunhofer Institute for Cognitive Systems IKS, Munich, Germany

2 Quantum Generative Learning

Generative modeling is a growing area of interest across all industries. Applications include anomaly detection, text and image generation, or speech and video synthesis. Ultimately, the objective of training a generative model is to express the underlying distribution of a dataset by a machine learning model. In QML, this model is typically represented by a parameterized quantum circuit (PQC) [12]. The structure of a PQC typically comprises sequences of quantum gates, each parameterized by real values. These parameters govern the transformations applied to qubits within the circuit, enabling the representation of complex functions and distributions. During the training of a quantum generative model, the probability amplitudes of the quantum state vector generated by a PQC are fitted to the probability distribution of the dataset; see, for example, reference [13]. We will refer to the probability mass function (PMF) of the dataset as p and to that of the state generated by the PQC as q. The absolute square values of the state vector give the PMF of the PQC.

Two popular training routines are the *quantum circuit Born machine* (QCBM) [14] and the *quantum generative adversarial network* (QGAN) [15], see [16] for a detailed review of both methods. The QCBM tries to minimize the Kullback–Leibler (KL) divergence, a well-known statistical distance, between p and q by adapting the model parameters via the covariance matrix adaption evolutionary strategy (CMA-ES) [17], a gradient-free optimizer.

Conversely, the QGAN follows the architecture of a classical GAN [18]. A GAN operates on the principle of adversarial training in a minimax 2-player game, employing two neural networks, the generator and the discriminator. The generator creates synthetic data instances, while the discriminator evaluates these generated samples alongside real ones. The two networks engage in a continual feedback loop, with the generator striving to improve its output and the discriminator refining its ability to differentiate between real and fake samples. Different architectures are proposed for QGANs, but a typical approach is to replace the classical generator with a PQC. The model parameters of the PQC are updated by gradient descent, and the parameter-shift rule determines the gradients, for example, reference [19]. The classical discriminator of the QGAN is optimized with the ADAM [20] optimizer, and the gradients are determined via backpropagation.

3 Benchmarking Quantum Computing

In its most general form, benchmarking is the process of comparing the performance of systems by a set of measurements. The workloads used are referred to as benchmarks [21], and the metrics are the criteria to compare the performance. In quantum computing, these metrics should characterize scale, quality, and speed [22]. Depending on the scope of attributes a benchmark assesses, they can be attributed to three categories [23]. (1) Physical Benchmarks focus on the basic physical properties of quantum hardware, such as the number of qubits and quantum gates. Examples are T1 and T2 relaxation times, gate fidelity, and readout fidelity; for example, see reference [24] for an extensive review. (2) Aggregated benchmarks assess the performance over a large set of device attributes. Prominent examples are quantum volume [25] and circuit layer operations per second [26]. (3) Application-oriented benchmarks test the performance of quantum computers in real-world scenarios. They simulate specific computational tasks that quantum computers are expected to perform, e. g., optimization, machine learning and chemistry algorithms. These benchmarks are particularly important for demonstrating the practical utility and efficiency of quantum systems in solving complex, real-world problems and are a key indicator of progress toward quantum advantage. Examples include QPack [27], QED-C [6, 7, 28] and Q-Score [29].

3.1 The QUARK Framework

The QUARK framework [30] orchestrates application-oriented quantum benchmarks in a well-defined, standardized, reproducible and verifiable way. It remains vendor-neutral to ensure unbiased application across different quantum computing platforms. At the heart of QUARK are three distinct modules: (1) the benchmark configuration is defined with the `Config Manager`. Here, attributes such as the number of qubits or the number of training iterations can be selected. In the second step, the (2) `benchmark manager` executes the configuration. Furthermore, the data is collected by the `Benchmark Record`, which contains information about the benchmark run (e. g., the git revisions of the framework), as well as the performance metrics and the configuration of a benchmark. In the following, we will describe the implementation of the generative modeling application in QUARK.

A benchmarking instance is defined by configuring six distinct modules (see Fig. 1): In the (1) generative modeling class, global properties, such as the number of qubits n, are defined. Subsequently, a (2) continuous or discrete

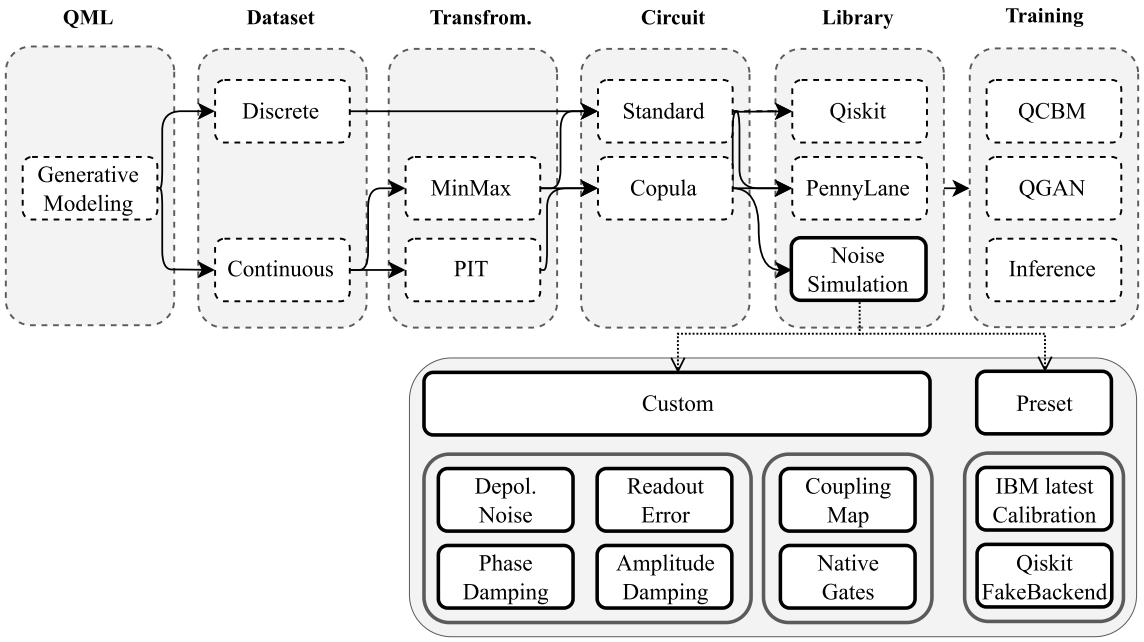

Fig. 1 Illustration of the components of the QUARK framework for quantum generative modeling with a detailed depiction of the noise module. We start at the left by defining the application, in this case, the training of a QML model. Then, the user defines the dataset for training, followed by necessary data transformations, before choosing the PQC ansatz to be used as a model. As explained in the main text, QUARK offers great flexibility regarding quantum libraries for implementing the circuit ansatz and training via QCBM or QGAN methods. At the bottom, we show the structure of the new Noise Simulation module. This module is designed to offer both predefined noise configurations and the flexibility to create entirely custom noise profiles and backend specifications. This allows for more accurate assessments of the robustness and performance of quantum algorithms under various noise conditions. The new modules are indicated by bold solid lines in the figure. More details are given in the main text

dataset is selected. The discrete datasets are characterized by a constraint on the bit string of length n, while the continuous datasets include both low-dimensional synthetic and real data. The continuous datasets are passed to a (3) transformation. This ensures the data is in a standard and normalized form. The MinMax transformation maps the marginal distributions to the interval [0, 1]. An alternative is the probability integral transformation (PIT), which makes the marginal distributions uniformly distributed. After applying the transformation, the data is mapped to a discrete probability distribution with 2^n bins. Next, the architecture of the PQC is selected in the (4) circuit module. The available quantum circuits include the *copula circuit* [31], which naturally respects the properties of data transformed via the PIT, and can only learn a probability distribution whose cumulative marginals are uniformly distributed. In the subsequent step, the architecture is mapped to the (5) Qiskit [32] or PennyLane [33] SDK. The library-agnostic definition of the architecture of the PQC enables comparative studies of quantum simulators from different vendors. In the last step, a pre-trained PQC is loaded for (6) inference or a training routine is configured. Training routines include the QCBM and QGAN, discussed in the previous section, but can be extended to other QML models. After defining the benchmarking instance, QUARK orchestrates the execution, data collection and visualization of the benchmark. For a detailed report on the QUARK framework, see References [8, 11].

Furthermore, both noisy and noise-free simulators are available. Depending on the individual requirements, one can configure the backend of the circuit to include multiple sources of errors. This can be done in two ways: (1) By specifying different error sources, like readout and depolarizing errors or amplitude and phase damping and chip-agnostic parameters, e. g., the coupling map, which represents the connectivity of the qubits; the basis gates, i.e., the gates that can be used on the backend. (2) We also provide an implementation of Qiskit FakeBackends, which are a predefined snapshot of the error rates, coupling map and basis gates [34]. Additionally, the latest calibration data of IBM's quantum processors can be accessed using qiskit_ibm_runtime. These backends include models of error sources present in current quantum hardware. This feature enables users to understand how quantum noise might influence their selected applications and compare them to an ideal environment or a real QPU.

4 Methodology and Results

In this Section, we use the QUARK framework to study the effects of statistical and quantum noise in QML applications. First, we compare the training routines of the QCBM and the QGAN. Next, we study the effects of quantum noise in the training of QCBMs.

4.1 Scalability of Quantum Generative Models Under the Influence of Statistical Noise

The training of the QCBM relies on estimating the full PMF q generated by the PQC. As we scale the number of qubits, n, the number of bins of the PMF scales exponentially. In the following, we will describe how the number of circuit executions, n_{shots}, scales, to keep the statistical error of the PMF, $|\alpha_i|^2$, bounded, where α_i are the probability amplitudes of the quantum state vector generated by a PQC. The probability of frequency n_i in bin i is given by the binomial distribution $B_{n_{\text{shots}},p_i}(n_i)$, with $\sum n_i = n_{\text{shots}}$ and $p_i = n_i/n_{\text{shots}}$. Furthermore, p_i is the estimator of $|\alpha_i|^2$. On average, if $|\alpha_i|^2$ decays exponentially with the number of qubits n, i.e. $|\alpha_i|^2 \propto 1/n^2$, then the variance is proportional to $\sigma^2(p_i) \propto 2^n/n_{\text{shots}}$. Therefore, the number of circuit executions needs to scale exponentially with the number of qubits to keep the statistical error on each estimator of $|\alpha_i^2|$ bounded. For the QCBM, the number of circuit executions of one epoch is given by $n_{\text{shots}} \cdot \lambda$. The population size λ is a hyperparameter of the CMA-ES optimizer and refers to the number of model parameters evaluated at any given iteration.

Unlike the QCBM, the number of circuit executions of the QGAN needed to update the model parameters does not increase exponentially with the number of qubits. The total number of circuit executions per epoch is given by $(2 \cdot n_{\text{parameters}} + 1) \cdot n_{\text{samples}}$. One circuit execution with the model parameters is needed to generate synthetic samples to update the discriminator. In the backward pass, two additional circuit executions per model parameter are necessary to determine the gradients with the parameter-shift rule.

Experimental Design: To showcase the different scaling behaviors of QCBMs and QGANs, we track the KL divergence as a function of the cumulative number of circuit executions, as depicted in Fig. 2. We fitted the copula circuit with 12 qubits and a depth of one to a dataset resembling the shape of the letter X, using the quantum noise-free Qiskit AerSimulator. For the QCBM, we use a population size of $\lambda = 5$ and train the models with $4 \cdot 10^5$ and $1 \cdot 10^6$ circuit executions to determine the PMF generated by the PQC. For the QGAN, we use a batch size of 20 and alternately update the generator and discriminator on each mini-batch. While training the generative models with a shot-based simulator, we report the KL divergence between the PMF[1] of the PQC and the target distribution. To compare the performance of the trained models, we fit a stretched exponential function, $f(x) = \alpha \cdot \exp(-\beta x^\gamma) + C_{\text{KL}}^{\text{conv}}$, to the loss curves and report the limit of the KL divergence $C_{\text{KL}}^{\text{conv}}$.

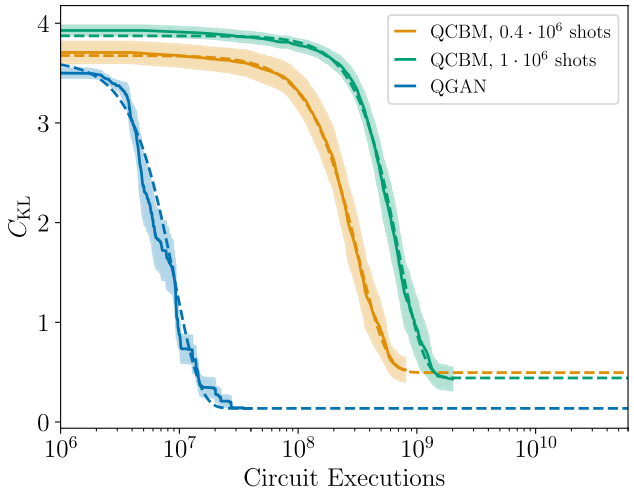

Fig. 2 KL divergence, C_{KL}, as a function of the cumulative number of circuit executions for the QCBM and the QGAN. The dashed lines indicate the stretched exponential function $f(x) = \alpha \cdot \exp(-\beta x^\gamma) + C_{\text{KL}}^{\text{conv}}$ fitted to the loss curves. For the training of the QCBM we show results of experiments, where we used 0.4×10^6 and 1×10^6 circuit executions to determine the PMF generated by the PQC. The QGAN converges with more than an order of magnitude fewer circuit executions than the QCBM to a lower limit. Each model was trained ten times and mean values and standard error on the mean $\sigma/\sqrt{10}$ are depicted as the solid lines and shaded areas, respectively

Discussion: For the QCBM, increasing the number of circuit executions to estimate the PMF of the state generated by the PQC from $4 \cdot 10^5$ to $1 \cdot 10^6$ leads to a slight decrease of the KL divergence at convergence from $C_{\text{KL}}^{\text{conv}} = 0.49 \pm 0.13$ to $C_{\text{KL}}^{\text{conv}} = 0.44 \pm 0.12$. The QGAN achieves faster convergence than the QCBM, requiring more than one order of magnitude fewer circuit executions. In addition to fewer circuit executions, the KL divergence of the QGAN converges to a lower limit with a value $C_{\text{KL}}^{\text{conv}} = 0.14 \pm 0.01$. To match the limit of the KL divergence of the QGAN with the QCBM, we would need to increase the circuit executions further, and the separation with respect to the circuit executions would become even more dominant.

Limitations: The QCBM was trained with the gradient-free optimizer CMA-ES. However, a gradient-based training of the QCBM [37] was not investigated and might lead to faster convergence despite needing more circuit evaluations per iteration to estimate the gradients. Furthermore, we used

[1] Here we use the precise PMF, not the estimated PMF, to circumvent the influence of shot-noise when comparing the model performance.

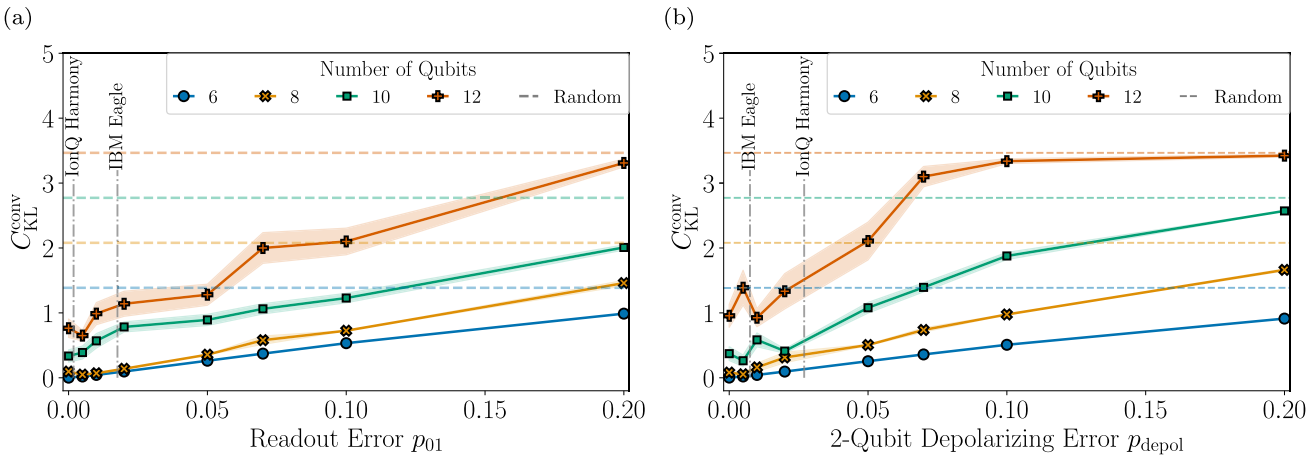

Fig. 3 Mean KL divergence after 4×10^4 circuit evaluations as a function of **a** the readout error and **b** the two-qubit depolarizing error for different circuit widths of a QCBM with the standard error ($\sigma/\sqrt{8}$). The vertical grey lines denote the error value of the IBM Eagle processor (Median ECR error: 7.477×10^{-3} and readout error: 1×10^{-2} of IBM Sherbrooke [35]) and IonQ Harmony (SPAM error: 1.8×10^{-3} and two-qubit gate errors: 2.7×10^{-2} [36]). The colored horizontal lines denote the KL divergence of the training data and a uniform distribution

only one dataset; exploring how the findings generalize to different datasets is an interesting path for future research.

4.2 Noisy Training of QCBMs

In the noisy intermediate-scale quantum (NISQ) era, the performance of QML algorithms is impacted by the presence of quantum noise [38, 39]. This noise stems from various sources, such as decoherence, imperfect gate operations, and environmental interference, and leads to a decreased fidelity of the quantum state generated by a PQC; for a detailed discussion of how noise can influence the training of PQCs, see [40].

Experimental design: We investigate the robustness of QCBMs against quantum noise to characterize the limits of current quantum hardware. To this aim, we train the QCBM and investigate the KL divergence at convergence under varying noise conditions. We fit the copula circuit with 6, 8, 10, and 12 qubits to a dataset that resembles the shape of the letter X. We execute the circuit $1 \cdot 10^4$ times to estimate the PMF. The gate set used for the copula circuit is native to IonQ Harmony. The number of gates of the circuit is reported in Table 1. We vary the probability of (a) readout and (b) two-qubit depolarizing errors, as depicted in Fig. 3. Readout errors are represented by a bit-flip channel, which means that with a probability p_{10} (p_{01}), the prepared input state $|1\rangle$ ($|0\rangle$) yields the measurement outcome 0 (1). In our experiments, we set $p_{10} = p_{01}$ [39]. Two-qubit depolarizing errors are characterized by the error rate, p_{depol}, that a two-qubit gate creates the fully mixed state instead of the desired output state of the operation.

Discussion: Figure 3a illustrates the influence of readout error on the KL divergence at convergence $C_{\mathrm{KL}}^{\mathrm{conv}}$. With increasing error rate, $C_{\mathrm{KL}}^{\mathrm{conv}}$ increases linearly. Even at an error rate of 0.1, the QCBM maintains robustness against readout errors, as the $C_{\mathrm{KL}}^{\mathrm{conv}}$ is still below the random baseline.[2] Unsurprisingly, the two-qubit depolarising error shows a stronger effect on $C_{\mathrm{KL}}^{\mathrm{conv}}$, as depicted in Fig. 3b for circuits with more gates, as multiple operations are performed per qubit, instead of measuring the state only once. The number of two-qubit gates increases with the circuit width (see Table 1), so does the influence of p_{depol} on the performance of the QCBM. The performance is still below the random baseline for 6 qubits and $p_{\mathrm{depol}} = 0.2$. For 12 qubits, however, the model performance corresponds to random guessing already for $p_{\mathrm{depol}} = 0.1$.

Limitations: Our focus on readout and two-qubit depolarizing errors provides a foundation for understanding specific noise sources, but other factors, such as phase errors

Table 1 The total number of the one-qubit gates (*RZ*, *SX*, *RX*, *H*) and two-qubit gates (*RXX*, *CX*) of the compiled circuit for 6, 8, 10 and 12 qubits

n_{qubits}	1-Qubit gates	2-Qubit gates
6	21	9
8	28	16
10	35	25
12	42	36

[2] The random baseline denotes the KL divergence between the training set PMF and the uniform distribution.

and crosstalk, are not considered. Our evaluation primarily considers the noise levels of state-of-the-art trapped ion quantum computers. However, the landscape of quantum hardware is rapidly evolving, and the generalizability of our findings to other quantum platforms with potentially different error characteristics needs further exploration. Incorporating mitigation and correction strategies for quantum errors could offer additional insights into enhancing the robustness of QCBMs in noisy quantum environments.

5 Conclusion and Future Work

In this work, we use and extend the QUARK benchmarking framework and illustrate its functioning with two applications that consider different aspects of noise in QML: statistical and quantum noise. The modular structure of QUARK makes it a versatile tool for a broad spectrum of research applications in QML and quantum computing in general.

Our experiments focused on the performance characterization of quantum generative models. A comparative analysis of QCBMs and QGANs revealed differences in their efficiencies. Remarkably, QGANs achieved faster convergence with reduced computational demands. Despite the recent advancements in QPU architectures leading to reduced error rates, noise is still a limiting factor in the current NISQ era. To effectively mitigate the impact of noise in QML, strategies such as minimizing gate counts, employing QPUs with lower noise profiles, and designing circuits with inherently more resistant architectures must be pursued.

Our studies on the influence of noise on the training of QCBMs are limited to readout and two-qubit depolarizing errors. Future studies may benefit from incorporating a more complete noise model, including noise sources such as amplitude damping or crosstalk between qubit pairs. Furthermore, extending our studies to different quantum generative models, such as QGANs, would be interesting. After simulating the influence of noise on the performance of quantum generative models, conducting the experiments on quantum hardware would be a natural step. One compelling aspect to explore is investigating if a model trained with a noisy simulator has learned to resist the noise. Based on our experience, the QUARK framework should be the tool of choice for future research.

Acknowledgements PR and CAR were partly funded by the German Ministry for Education and Research (BMB+F) in the project QAI2-Q-KIS under Grant 13N15583. AL was partly funded by the Bavarian State Ministry of Economic Affairs in the project BenchQC under Grant DIK-0425/03.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data availability The source code and configurations needed to replicate the experiments are publicly accessible on GitHub [30].

References

1. Proctor T, Rudinger K, Young K, Nielsen E, Blume-Kohout R (2022) Measuring the capabilities of quantum computers. Nat Phys 18(1):75–79
2. Erhard A, Wallman JJ, Postler L, Meth M, Stricker R, Martinez EA, Schindler P, Monz T, Emerson J, Blatt R (2019) Characterizing large-scale quantum computers via cycle benchmarking. Nat Commun 10(1):5347
3. Blume-Kohout Robin, Young Kevin C (2020) A volumetric framework for quantum computer benchmarks. Quantum 4:362
4. Mills Daniel, Sivarajah Seyon, Scholten Travis L, Duncan Ross (2021) Application-motivated, holistic benchmarking of a full quantum computing stack. Quantum 5:415
5. Resch Salonik, Karpuzcu Ulya R (2021) Benchmarking quantum computers and the impact of quantum noise. ACM Comput Surv 54(7):07
6. Lubinski Thomas, Johri Sonika, Varosy Paul, Coleman Jeremiah, Zhao Luning, Necaise Jason, Baldwin Charles H, Mayer Karl, Proctor Timothy (2023) Application-oriented performance benchmarks for quantum computing. IEEE Trans Quantum Eng 4:1–32
7. Lubinski T, Goings JJ, Mayer K, Johri S, Reddy N, Mehta A, Bhatia N, Rappaport S, Mills D, Baldwin CH, Zhao L, Barbosa A, Maity S, Mundada PS (2024). Quantum algorithm exploration using application-oriented performance benchmarks,
8. Finžgar JR, Ross P, Holscher L, Klepsch J, Luckow A (2022). QUARK: A framework for quantum computing application benchmarking. In 2022 IEEE International Conference on Quantum Computing and Engineering (QCE). IEEE, 9
9. Bowles J, Ahmed S, Schuld M (2024). Better than classical? the subtle art of benchmarking quantum machine learning models,
10. Ladd TD, Jelezko F, Laflamme R, Nakamura Y, Monroe C, O'Brien JL (2010) Quantum computers. Nature 464(7285):45–53
11. Kiwit FJ, Marso M, Ross P, Riofrio CA, Klepsch J, Luckow A (2023). Application-oriented benchmarking of quantum generative learning using quark. In 2023 IEEE International Conference on Quantum Computing and Engineering (QCE), volume 01, pages 475–484,
12. Benedetti M, Lloyd E, Sack S, Fiorentini M (2019) Parameterized quantum circuits as machine learning models. Quantum Sci Technol 4(4):043001
13. Schuld M, Petruccione F (2021) Machine Learning with Quantum Computers. Springer International Publishing, Quantum Science and Technology
14. Marcello Benedetti, Delfina Garcia-Pintos, Oscar Perdomo, Vicente Leyton-Ortega, Yunseong Nam, Alejandro Perdomo-Ortiz (2019) A generative modeling approach for benchmarking

and training shallow quantum circuits. NPJ Quantum Inform 5(1):1–9
15. Lloyd S, Weedbrook C (2018) Quantum generative adversarial learning. Phys Rev Lett 121:040502
16. Riofrío CA, Mitevski O, Jones C, Krellner F, Vučković A, Doetsch J, Klepsch J, Ehmer T, Luckow A (2024). A characterization of quantum generative models. ACM Transactions on Quantum Computing, 04 . Just Accepted
17. Hansen N, Akimoto Y, Baudis P (2019). CMA-ES/pycma on Github Zenodo. https://doi.org/10.5281/zenodo.2559634
18. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014). Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, Advances in Neural Information Processing Systems, volume 27. Curran Associates, Inc.,
19. Schuld M, Bergholm V, Gogolin C, Izaac J, Killoran N (2019) Evaluating analytic gradients on quantum hardware. Phys Rev A 99:032331
20. Kingma D, Ba J (2015). Adam: A method for stochastic optimization. In International Conference on Learning Representations (ICLR), San Diega, CA, USA,
21. Jain R (1991) The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling. Wiley, New York
22. Amico M, Zhang H, Jurcevic P, Bishop LS, Nation P, Wack A, McKay DC (2023). Defining best practices for quantum benchmarks. In 2023 IEEE International Conference on Quantum Computing and Engineering (QCE), volume 01, pages 692–702,
23. Wang Junchao, Guo Guoping, Shan Zheng (2022) Sok: Benchmarking the performance of a quantum computer. Entropy 24(10):1467
24. Eisert J, Hangleiter D, Walk N, Roth I, Markham D, Parekh R, Chabaud U, Kashefi E (2020) Quantum certification and benchmarking. Nat Rev Phys 2(7):382–390
25. Cross AW, Bishop LS, Sheldon S, Nation PD, Gambetta JM (2019) Validating quantum computers using randomized model circuits. Phys Rev A 100:032328
26. Wack A, Paik H, Javadi-Abhari A, Jurcevic P, Faro I, Gambetta JM, Johnson BR (2021). Quality, speed, and scale: three key attributes to measure the performance of near-term quantum computers,
27. Mesman K, Al-Ars Z (2022) and Matthias Moller. Quantum approximate optimization algorithms as universal benchmark for quantum computers, Qpack
28. Lubinski T, Coffrin C, McGeoch C, Sathe P, Apanavicius J, Bernal N, David E (2023). Optimization Applications as Quantum Performance Benchmarks
29. Martiel Simon, Ayral Thomas, Allouche Cyril (2021) Benchmarking quantum coprocessors in an application-centric, hardware-agnostic, and scalable way. IEEE Trans Quantum Eng 2:1–11
30. Quark (2023) A framework for quantum computing application benchmarking. quarkGithub: https://github.com/QUARK-framework/QUARK
31. Zhu EY, Johri S, Bacon D, Esencan M, Kim J, Muir M, Murgai N, Nguyen J, Pisenti N, Schouela A, Sosnova K, Wright K (2022) Generative quantum learning of joint probability distribution functions. Phys Rev Res 4:043092
32. Qiskit contributors. Qiskit: An open-source framework for quantum computing, 2023
33. Bergholm V, Izaac J, Schuld M, Gogolin C, Alam MS, Ahmed S, Arrazola JM, Blank C, Delgado A, Jahangiri S, et al. (2018) Pennylane: Automatic differentiation of hybrid quantum-classical computations. arXiv preprint arXiv:1811.04968,
34. Winston E, Moreda D (2018). Qiskit backends: What they are and how to work with them. Medium,
35. Steffen M, Chow J, Sheldon S, McClure D (2024). IBM Research A new eagle in the poughkeepsie quantum datacenter: Ibm quantum's most performant system yet,
36. IonQ Collaborators. IonQ Harmony, 2024
37. Liu Jin-Guo, Wang Lei (2018) Differentiable learning of quantum circuit born machines. Phys Rev A 98:12
38. Preskill John (2018) Quantum computing in the nisq era and beyond. Quantum 2:79
39. Nielsen MA, Chuang IL (2010) Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge University Press,
40. Oliv M, Matic A, Messerer T, Lorenz JM (2022). Evaluating the impact of noise on the performance of the Variational Quantum Eigensolver, 9