Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Benchmarking analysis of various Quantum computing hardware units for VQC through QAOA

Veronika Liutarevich

veronika.liutarevich.22@ucl.ac.uk

University College London

Cyril Renaud

University College London

Article

Keywords:

Posted Date: November 16th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11478438/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Benchmarking analysis of various Quantum computing hardware units for VQC through QAOA

Veronika Liutarevich[1,*] **and Cyril Renaud**[1]

[1]Department of Electrical Engineering, University College London, London, United Kingdom.
[*]veronika.liutarevich.22@ucl.ac.uk

ABSTRACT

This study follows a holistic and highly abstracted analysis of the performance of a set of quantum computing devices for Quantum Machine Learning by proxy of running standardized Quantum Approximate Optimization Algorithm (QAOA) and Variational Quantum Circuits (VQC). The aim of the study was to determine the degree of performance discrepancies between devices with different architectures from hardware providers, including IBM, IQM and IonQ. The study has concluded that device rankings stay broadly stable across QAOA and VQC, making noise floor and calibration more indicative than modality. Same-family devices diverged as much as cross-vendor ones, indicating architecture alone as a weak predictor as well. The device performance was differentiated based on factors outstretching binary accuracy, such as Hamming weight distribution, variational distance and parity error for a more complete picture.

1 Introduction

In the Noisy Intermediate Scale Quantum (NISQ) era, quantum computing systems vary in modality and architecture, making it crucial to differentiate between system performance. Rapid growth in hardware from vendors such as IBM, Google, Xanadu, and IonQ has been accompanied by a proliferation of benchmarking metrics, complicating systematic comparison.[1,2] A growing body of work advocates protocol-based benchmarking, where structured quantum workloads expose device behaviour in a hardware-agnostic manner.[3–5] Such protocols reveal systematic error patterns that aggregate metrics can miss.[6,7] At the application level, work has been done to enable cross-platform comparisons of cross-modality systems.[8] Platforms significantly affect algorithmic performance, highlighting the importance of hardware-based rather than purely simulation-based benchmarking.[1,9].

Quantum Machine Learning (QML) is cited to provide a meaningful speedup over classical systems.[10,11] QML encompasses classification, regression, generative modeling, and optimisation algorithms implemented using quantum circuits. Parameterised quantum circuits (PQCs) are the dominant computational primitive underlying variational quantum circuits, quantum neural networks, kernel methods, and hybrid quantum-classical models.[12,13] While this common structure facilitates standardisation, performance remains highly sensitive to ansatz design, data encoding, optimisation strategy, and noise.[14] Among near-term quantum algorithms, Variational Quantum Classifiers (VQCs) and the Quantum Approximate Optimisation Algorithm (QAOA) have received the most benchmarking attention.[15,16] A major challenge for benchmarking variational algorithms is the barren plateau phenomenon.[17] Plateaus arise from factors including high-dimensional parameter spaces, global cost functions, and hardware noise, causing gradients to vanish and optimisation to fail.[18] Consequently, final model accuracy alone may be an insufficient benchmark; trainability and loss-landscape structure are also important performance characteristics.[16]

In this paper, we will benchmark different quantum hardware models' performance on the same QML algorithms: QAOA and VQC. This article will aim to look further than accuracy alone, focusing on how the the models perform rather than the binary task completion status. It will be following 13 quantum computing systems across superconducting and ion-trap modalities quantum hardware and architecturally different classical simulators.

2 Background

2.1 Quantum Approximate Optimization Algorithm

The Quantum Approximate Optimization Algorithm (QAOA) is a hybrid quantum-classical variational algorithm designed to solve combinatorial optimization problems. Originally proposed by Farhi, Goldstone, and Gutmann, QAOA encodes problems as Ising models or quadratic unconstrained binary optimization (QUBO) cost functions.[19] The algorithm prepares a parameterized quantum state through alternating time evolutions under a problem-specific cost Hamiltonian and a transverse-field mixing Hamiltonian. Layer-dependent angles γ and β control these evolutions and are optimized classically to minimize the expectation value of the cost function.[20] For a fixed circuit depth p, QAOA scales linearly in both p and the locality of the

problem Hamiltonian, making it particularly suitable for noisy intermediate-scale quantum (NISQ) processors. Theoretical analysis shows that constant-depth QAOA instances achieve nontrivial approximation ratios on regular graphs and can outperform classical local algorithms in certain regimes. Max-Cut is the NP-hard problem of finding a partition of the graph's vertices into two distinct sets that maximizes the number of edges between the two sets. The MaxCut problem on an undirected graph

$$G = (V, E) \tag{1}$$

seeks a bipartition of vertices that maximizes the number of edges crossing between sets, which is NP-hard. The standard Ising formulation assigns binary variables $z_i \in \pm 1$ to vertices and defines the cost function:

$$C(z) = \frac{1}{2} \sum_{(i,j)\in E} (1 - z_i z_j) \tag{2}$$

, whose maximum equals the MaxCut size. In the quantum setting, this promotes to the diagonal cost Hamiltonian:

$$H_C = \frac{1}{2} \sum_{(i,j)\in E} (I - Z_i Z_j) \tag{3}$$

, where Z_i denotes the Pauli-Z operator on qubit i[21,22]. The canonical QAOA ansatz begins from the uniform superposition $|+\rangle^{\otimes n}$ and applies p layers of alternating unitaries:

$$|\gamma, \beta\rangle = \Pi_{l=1}^{p} e^{(-i\beta_l H_M)} e^{(-i\gamma_l H_C)} |+\rangle^{\otimes n} \tag{4}$$

with mixer Hamiltonian. The classical objective minimizes $\langle H_C \rangle_{\gamma,\beta}$, or the expectation value, concentrating measurement probability mass on high-quality cuts. For ring graphs, including the ten-node cycle considered here, the QAOA dynamics admit analytic treatment, serving as a canonical testbed for parameter landscapes and performance bounds. Because a 10-vertex ring is bipartite, its Max-Cut optimum corresponds to exactly two bitstrings out of 1024 and is trivially found classically via an alternating two-colouring; the instance is used here for its analytic tractability rather than its computational hardness, so any shortfall in hardware success probability below the noiseless ceiling reflects device noise rather than problem difficulty. The MaxCut problem is taken to maximize the cut size, or, equivalently, minimizing the negative costs.[23,24] QAOA exemplifies variational quantum circuits (VQCs), sharing the hybrid optimization structure central to quantum machine learning (QML). Both paradigms train parameterized quantum states via classical optimization of task-specific loss functions—energy expectation for QAOA, classification error or likelihood for QML models.[21] Insights from QAOA expressivity, trainability, and noise resilience thus directly inform QML architecture design. QAOA benchmarking probes noise-induced limitations, barren plateaus, and hardware-ansatz co-design principles critical for NISQ-era QML deployment.[25]

2.2 Variational Quantum Circuits

Variational quantum circuits (VQCs) are parametrised quantum circuits optimised in a hybrid quantum–classical loop, and currently constitute the central computational primitive for variational quantum algorithms on NISQ hardware.[26] The approach traces back to early variational methods such as the variational eigensolver, which demonstrated that ground-state properties of quantum systems can be estimated by preparing a trial state with a parameterized circuit and updating its parameters via classical feedback. A typical VQC based quantum machine learning model is defined by three components: a data-encoding unitary U_x that embeds classical inputs into quantum states, an ansatz $U(\theta)$ composed of layers of parameterised single qubit rotations interleaved with fixed entangling gates matching the device connectivity, and a measurement stage where expectation values of observables are mapped to task specific loss functions.[26] Within this framework, the learning task reduces to minimizing an empirical risk functional $L(\theta)$ over circuit parameters using classical optimisers, while the quantum device is used solely as a non classical state preparation and measurement engine.[14] This architecture underpins a broad family of quantum algorithms and has been deployed in applications ranging from quantum chemistry to data analysis to time series modelling, thereby positioning VQCs as a unifying abstraction for quantum machine learning on current hardware. Their flexibility, however, comes with important challenges: circuit expressivity must be balanced against trainability and noise resilience, and optimization can be hindered by barren plateaus and other landscape effects. VQC are often interpreted as quantum neural networks because of their layered arrangement of parametrized gates mirrors and training end to end by minimizing task specific loss functions.

3 QAOA benchmarking

3.1 Methods

This work implements a 10 qubit QAOA circuit solving MaxCut on a ten-node ring graph. Given a graph of nodes connected by edges, the goal is to partition the nodes into two sets such that the number of edges crossing the partition is maximized.[27] Specified in OpenQASM 2.0 (Listing 1) with native cost/mixer layers respecting cycle connectivity, the circuit avoids extraneous SWAP overhead beyond hardware topology constraints. The fixed-depth instance executed on 13 cloud-accessible quantum processors spanning simulators, superconducting and trapped-ion modalities, using 1024 shots per run for consistent statistical comparison. As each device was sampled in a single job, the confidence intervals reported in Figures 1 and 2 quantify shot-to-shot sampling uncertainty only; they do not capture day-to-day calibration drift, which Section 4.3 identifies as a likely larger source of cross-device variability.

A key metric for the performance of a QAOA algorithm is the expected MaxCut value. A higher expected MaxCut value means the sampler is placing more weight on bitstrings that cut more edges, which makes expected MaxCut value one of the clearest "overall quality" measures. The maximum value for the given MaxCut problem is 10 since the graph is bipartite.[27]. Other metrics explored include:

- Success probability - the fraction of shots that hit the exact optimal cut, or the best-known optimum. In QAOA, this is a very strict metric showing whether the circuit is landing on the best solution with nontrivial probability.
- Cost distribution - the probability distribution of sampled MaxCut costs. In QAOA, this metric measures whether the output is sharply concentrated near correct solutions, broadly spread, or multimodal.
- Cumulative cost distribution - showing how much probability mass has accumulated up to a given cost threshold. In QAOA, it is useful for comparing devices or parameter settings because it tells you how quickly the sampler reaches high-quality solutions across the full cost range.
- Single qubit marginals - the individual probabilities of each qubit being measured as 0 or 1. In QAOA, strong structure in the marginals can indicate that the optimizer has learned variable-level biases implied by the problem graph; marginals near 0.5 indicate noise-dominated behaviour.
- Ring-edge Pauli correlators $< Z_i Z_j >$ - are pairwise ZZ-correlations on neighboring edges of the ring graph. In QAOA, they diagnose whether the circuit is learning the local interaction structure encoded by the cost Hamiltonian. Large-magnitude, patterned correlators usually mean the ansatz is capturing real problem geometry.

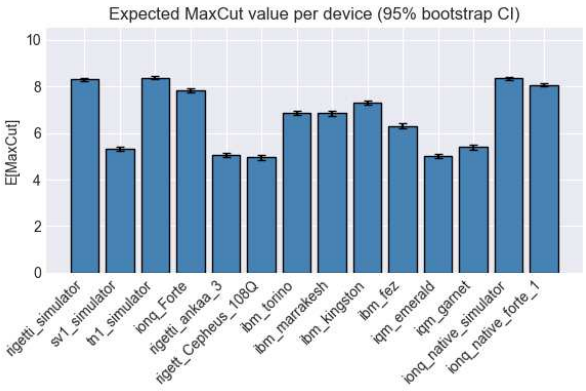
3.2 Results

We can see from Figure 1a that certain devices perform noticeably better with regards to the expected MaxCut value: Rigetti simulator, TN1 Simulator, IonQ native simulator * and IonQ Forte-1 as well as IBM Kingston following closely behind. Most of these devices are simulators. Since simulators are not subjected to the quantum noise, we would often expect a behaviour less prone to noise.[31,32] IonQ Forte-1 (represented here twice, run through the Amazon Braket interface and natively through IonQ platform) is a physical device operating on trapped-ion modality. It is interesting to see most quantum hardware outperform simulator model SV1. Outperforming simulators is the long-term aim for higher qubit-count devices, but with persistent improvement of both the quantum hardware and simulator technologies simultaneously, this is hardly ubiquitous. SV1 is state-vector simulator processing the operations sequentially, making it less efficient. Thus, we can see that a real quantum computing device, even granted a modest qubit count, outperforms state-vector simulators in certain tasks.

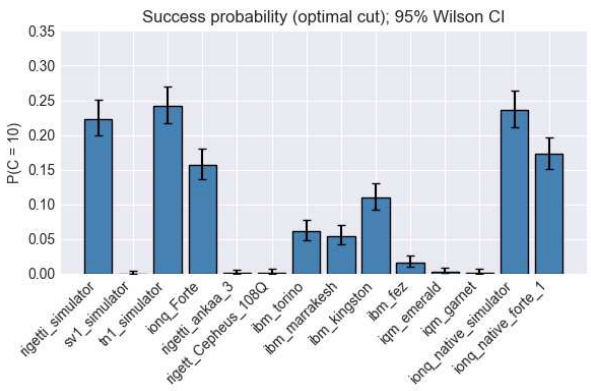
In Figure 1b, one can see the success probability for each circuit. Across the range of tested devices, the success probability of an optimal cut does not exceed 25%, as expected for NISQ devices. It is notable that the devices that displayed the highest expectation MaxCut value also show higher success probability. While the simulators outperform the leading hardware device, IonQ Forte and IBM Kingston also show non-trivial success probabilities. As can be seen from figure 2a, their bitstring entropy is, on average, also lower, i.e. they are concentrating probability on fewer outcomes thus converging on their best solution faster. Although still below 30%, the percentage of most popular bit strings is higher, showing less random behaviour (Figure 2b).

In Figure 3a, the cost distribution is shown. Comparing the plots in Figure 1a and 3a, we can see that the cost peaks at the estimated MaxCut value point for each device. Cost distribution graph reveals how spread out the solutions for each device are.

*We used three simulators: Amazon's SV1 and TN1 (on Amazon Braket) and Rigetti's QVM (on Azure Quantum). SV1 is a state-vector simulator. It stores all 2^n amplitudes and applies each gate in turn, so its runtime grows linearly with gate count and exponentially with qubit count, which limits it to 34 qubits[28]. TN1 represents the circuit as a tensor network. It first searches for an efficient contraction order, then performs the contraction. Its cost depends on the circuit's treewidth rather than its qubit count, so it can handle up to about 50 qubits on low-treewidth circuits but struggles with highly connected ones[29]. The QVM is also a state-vector simulator. It adds density-matrix evolution and Pauli noise channels, which we did not use here, and it has the same exponential scaling as SV1[30].

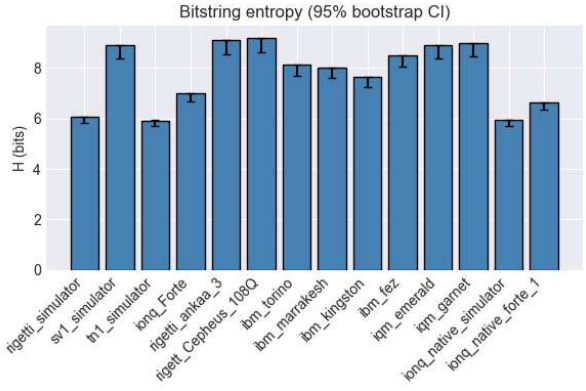


(a) Expected MaxCut value for each of the tested devices

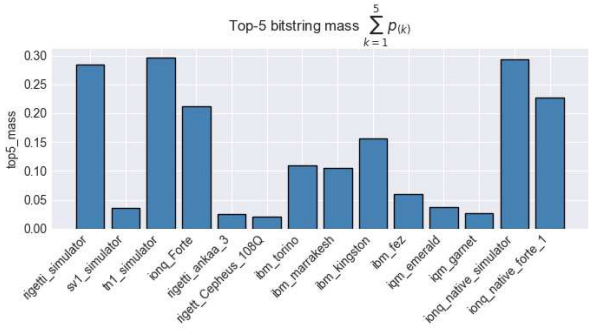


(b) Success probability of the QAOA algorithm across devices

Figure 1. Metrics defining the performance of the QAOA algorithm across devices

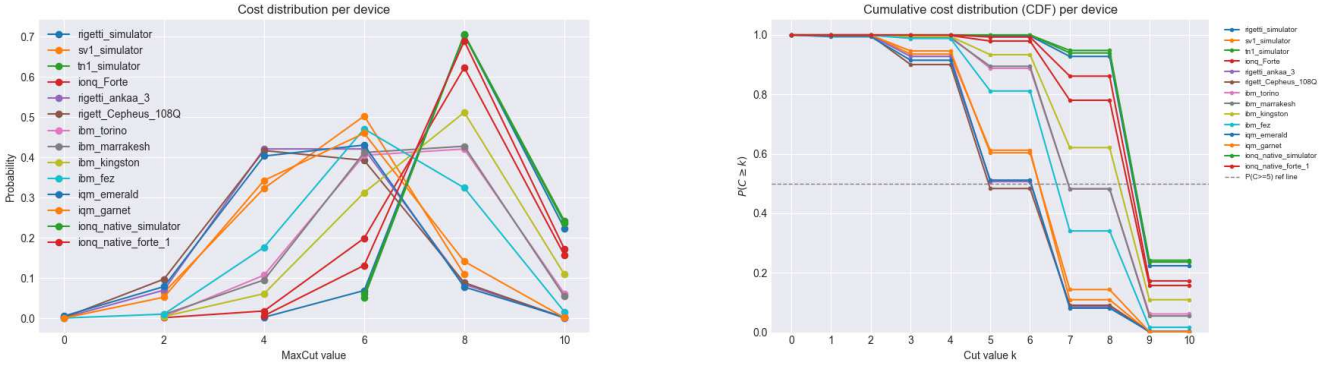


(a) Bitstring entropy per device



(b) Top 5 bitstrings mass detected in each device run

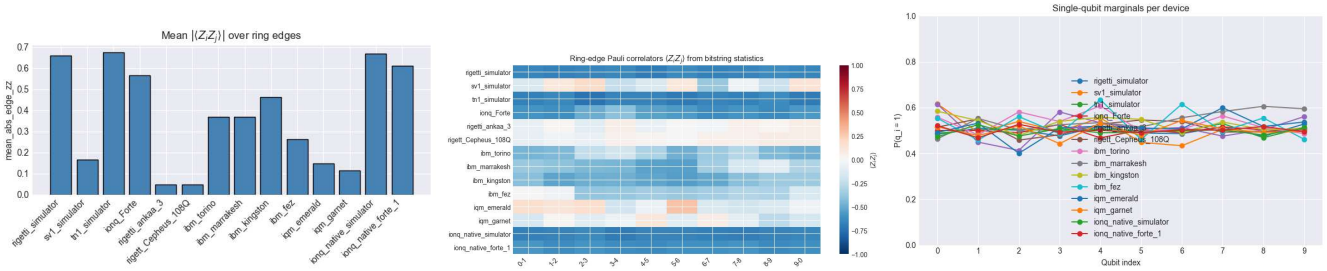
Figure 2. Bitstring distribution on QAOA algorithm across devices



(a) Linear cost distribution per device

(b) Cumulative cost distribution per device

Figure 3. Cost distribution on QAOA



(a) Mean $< Z_i Z_j >$ per device

(b) Ring-edge Pauli correlators

(c) Single qubit marginals per device

Figure 4. Accumulation of ring edge metrics

The cumulative version of the cost distribution 3b graph demonstrates that higher fidelity devices have the fallout at a later point in the graph.

Even though there are clear leaders in the performance, it is useful to see that the devices still display a lot of noise. Single qubit marginals graph in Figure 4c show mean expectation values $< Z_i >$ oscillating between 0.4 and 0.63, which is akin to random circuit behaviour.

In the ring-edge Pauli correlators graph in Figure 4b) we can see that the better performing devices also were better at learning from the local interactions, and that the Pauli correlators for them were strong negatives. The weaker-performing devices did not show strong magnitude correlation, oscillating around zero, which can be attributed to noise. This can be further seen from Figure 4a, showing the average of the edge correlators $<Z_iZ_j>$ over all ring edges. In QAOA, it compresses the local structure into one scalar: if it moves in the expected direction as optimization improves, that usually means the circuit is learning the graph's local constraints consistently rather than only on a few edges. This graph shows the discriminative performance variance of the devices very clearly.

3.3 Discussion

Most physical devices outperformed SV1, but not TN1 or the Rigetti QVM. This is a meaningful milestone for NISQ devices,since it validates the use of quantum computers for more complex circuits as opposed to simulator use. Analyzing the performance of quantum hardware devices, one of the main purposes was to determine the possible reasons for difference in performance. One obvious culprit is the architecture of the device. For example, the devices from the same provider (eg. Emerald and Garnet from IQM) have been demonstrating performance within 5% of one another across all metrics. The biggest outlier in this has been IBM. Their Kingston model has outperformed the others consistently. This discrepancy in the performance of the models from the same IBM Heron family with the same architecture opens doors to evaluating novel improvements on the same broad architecture.[33] For example, IBM Kingston supports 340K CLOPS (Coherence Layer Operations Per Second), which makes it further optimised for throughput. Higher CLOPS means the device executes circuit layers faster. This matters for result quality because quantum states decohere over *real time*, not over gate count. So if Kingston can execute the same 10-qubit circuit in shorter time than Fez or Marrakech, the qubits spend less time decohering between gates generating cleaner output distributions. CLOPS also captures classical control latency, qubit reset time, and measurement

throughput. A device with high CLOPS has effectively tighter integration between its classical control stack and quantum layer. However, CLOPS alone doesn't fully explain Kingston's advantage over Fez and Marrakech because the VQC circuit is not built to register mid-circuit feedback. Other factors included could involve variations in device performance from day to day.

Due to tiny atomic variations during manufacturing, each individual machine has its own distinct daily calibration map. A device might have slightly better gate fidelity on one side of its chip, while Marrakesh might have better readout accuracy on another. This is further enhanced by daily changes in environmental factors and noise. This creates an important limitation in the device analysis. A clear example of this here is the IonQ Forte-1. The exact same QAOA circuit has been run twice on this device: via Amazon Braket and naively through IonQ's own quantum platform. While it is true that the there is a small discrepancy between the expectation value between those two distinct runs (Figure 1a), the deviation is less than $|0.1|$ and does not affect the big picture.

4 Variational Quantum Circuit (VQC)

4.1 Methods

A quantum machine learning algorithm typically consists of two parts: generating weights in a classically run algorithm, and then building a circuit that uses these weights for quantum computations. Here, the classical algorithm was configured to generate different sets of weights and create OpenQASM circuits from them. 10 devices are tested with the VQC algorithm as opposed to the 13 devices for QAOA because IBM Torino, Rigetti Ankaa-3 and Rigetti simulator became deprecated or unavailable in the 3 week span between the experiments. None of them represent critical data points as their architectures are represented with other devices. Each device was tested with 11 circuits with different weights. The circuits were carried through on all the devices and executed with 1024 shots for each job. The results were then averaged out before being used in comparison work.

While we do care more about the comparative performance of devices, this Methodology section will represent a detailed study of a singular device, IBM Heron Marrakech, in order to present an example of analysis. The comparison will then be picked up in the Results section. Results for each of the other individual device are available in the paper's GitHub repository.

In this paper, we are evaluating the performance of the Variational Quantum Circuit as a Machine Learning classification circuit. The classification circuit presented to the model is rather simple: to classify the images with a vertical line and a horizontal line into buckets. The dataset and the inspiration for the problem definition came from the IBM workshop on Variational Quantum Circuits, but scaled for a 10 qubit system instead of an 8 qubit one.[34] The circuit has achieved a 100% on IBM Marrakech, as can be see in Figure 6. As this result is based on only 11 test circuits with an 8:3 class split, the 95% Wilson interval for the observed 11/11 outcome is [74.1%, 100%], and the 72.7% majority-class baseline should be borne in mind when interpreting this figure. You can see a per-job outcome in Figure 8. This is perhaps unsurprising, granted the simplicity of the task, but it does show that quantum computers even in NISQ are able to solve simple problems. It is important, however, to consider how the noisy nature of these systems affects the results at this level. If you look at Figure 7, the outcomes we have from the parity of the real jobs keep close to the ideal diagonal and the circuit does not show high bias, however, the entropy levels are very high. What this means is that while the circuit comes to correct solutions, it is not a sustainable behaviour due to much higher entropy and could fail with a larger circuit.

It is important to consider is a confidence margin for each job. It gauges the distance between a data point and the model's decision boundary. From Figure 9a, we can see that the confidence margins across the jobs are relatively low, with the highest one being below 50%.

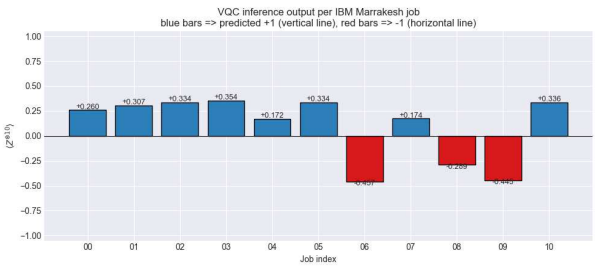


Figure 8. Prediction of the VQC circuit on IBM Heron Marrakech per job

Supplementing the traditional Machine Learning metric, we can use specific metric applicable to Quantum Machine Learning: the global output parity. The parity of shots reference to the ratio of 1s and 0s in the output string. Thus, an even

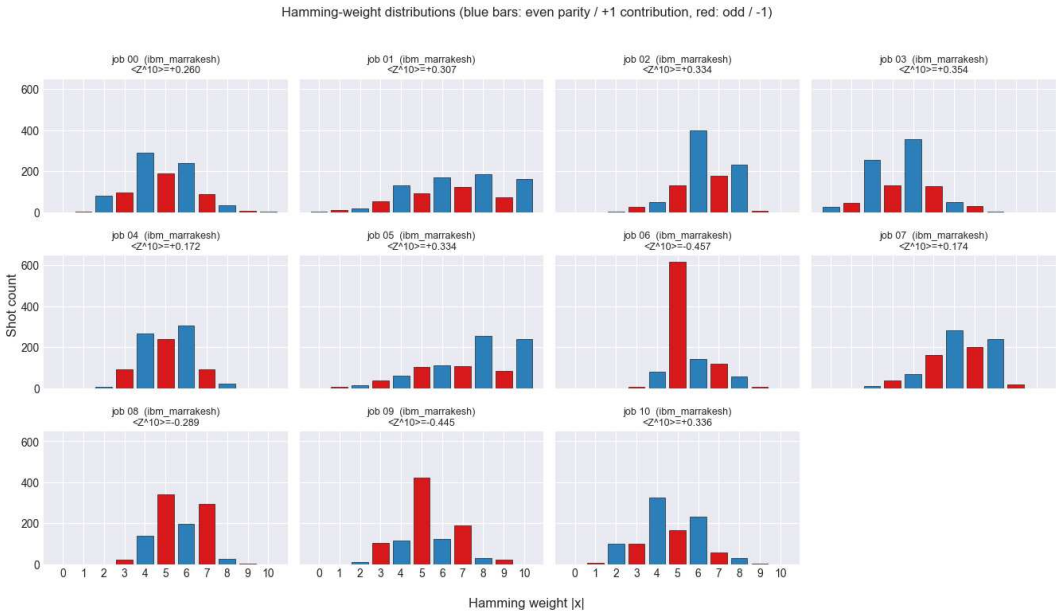


Figure 5. Hamming-weight distribution between jobs for IBM Heron Marrakech

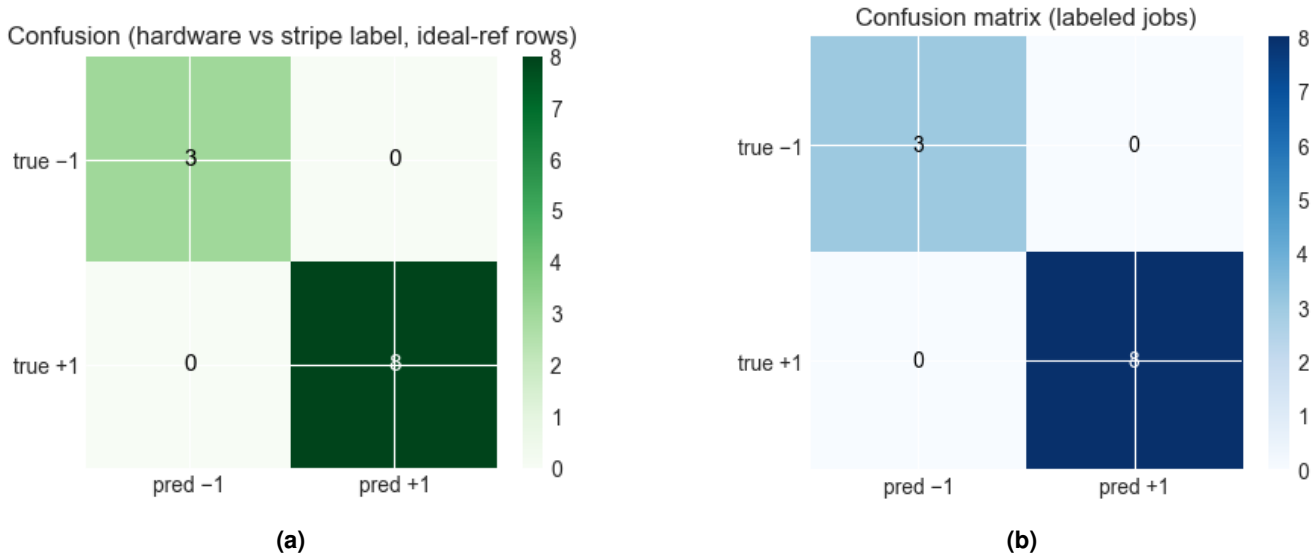


Figure 6. Reference confusion matrix and true confusion matrix on IBM Heron Marrakech coincide

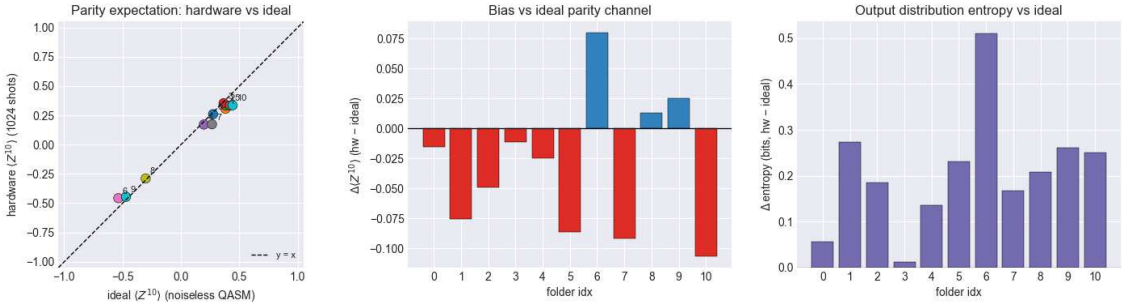
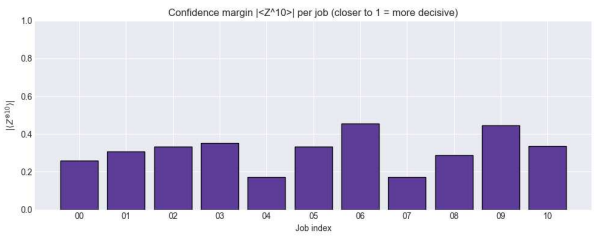
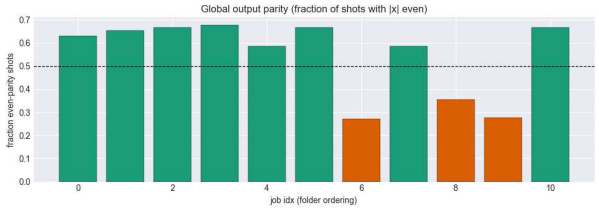


Figure 7. Comparison of the performance of Marrakech vs. ideal outputs. Shows that the behaviour is not far from ideal in accuracy (low bias and very close parity expectation) but has a very large entropy difference due to noise

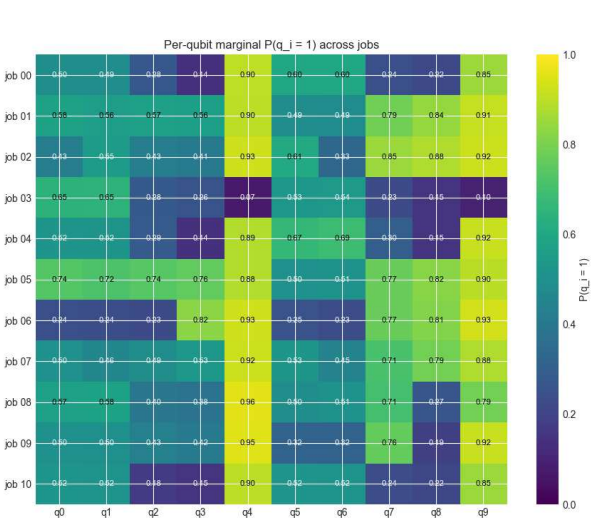


(a) Confidence margins per job

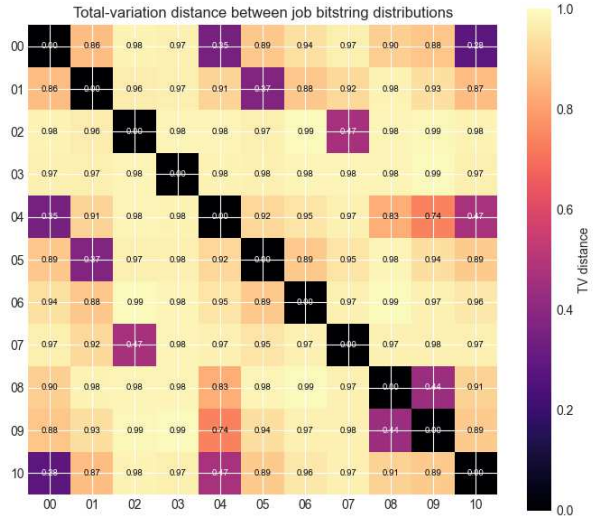


(b) Global output parity - the fraction of even parity shots per job

Figure 9. Confidence vs. Parity output on IBM Heron Marrakech



(a) Per qubit marginals matrix per job and per qubit



(b) Variational difference between jobs on the qubit level

Figure 10. IBM Heron Marrakech metrics concerning qubit marginals and variational distance

parity shot will be a shot with an even number of 1s and, conversely, an odd parity shot will be a shot with an odd number of 1s in the output. In QML systems, fraction of even-parity shots is used as a proxy for confidence. Since we are considering a classification problem, the output should be either 1 or 0 depending on the predicted class. If the fraction of even-parity shots is higher than 0.5, the model will predict Class A. The confidence margin then shows how far the result is from the confidence margin. Another use for this metric is for error identification and correction. While we have a healthy even-parity proportion in the dataset, jobs 6, 8 and 9 lag well below the confidence margin for 1. These are the 3 jobs that are correctly identifying -1 as the classification value.

In Figure 5 you can see the Hamming weights per job. The Hamming weight (W_H) is the total count of `1`s in a binary string. The graphs show what was the per-qubit Hamming weight based on the number of shots. Looking at the qubit-level behaviour of the circuit allows us to decipher where the most weight of the certainty lies. Deeper than computing the results, this can open the window at the way the system is operating.

Zooming in deeper at per-qubit behaviour, we can see the local behaviour of individual qubits in Figure 10a. While the mean P(q=1) across jobs and qubits is 0.555, very close to complete randomness, the standard deviation is 0.145, also showing a high spread (Figure 11).

While the spread is relatively high, what is important is that the consensus is reached toward the correct outcome. As can be seen in Figure 12, the value of top bitstring values trumps the overall noise and also coincides with the overall returned classification (Figure 8).

We can see the jobs largely similar, which is an encouraging sign. While cross-job similarity is the product not only of the circuit itself but also of its environment, i.e the specific moment in time the job was executed, it does nonetheless shows that the circuit is stable as of the time of testing, as well as demonstrates the potential of IBM Heron Marrakesh of sustaining a circuit.

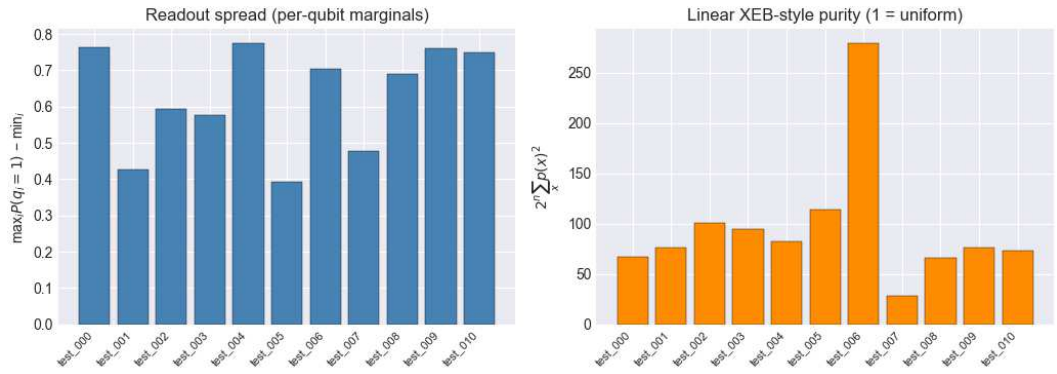


Figure 11. Cross entropy benchmarking of IBM Heron Marrakech.

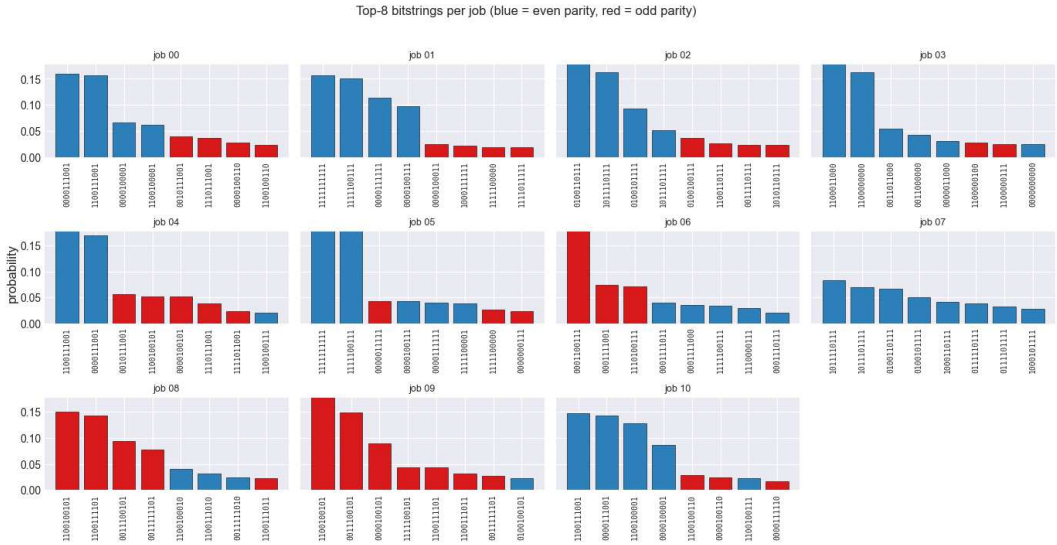


Figure 12. Top 8 bitstring values per job on IBM Heron Marrakech. Top 2 bitstrings outweigh the others and determine the final job decision.

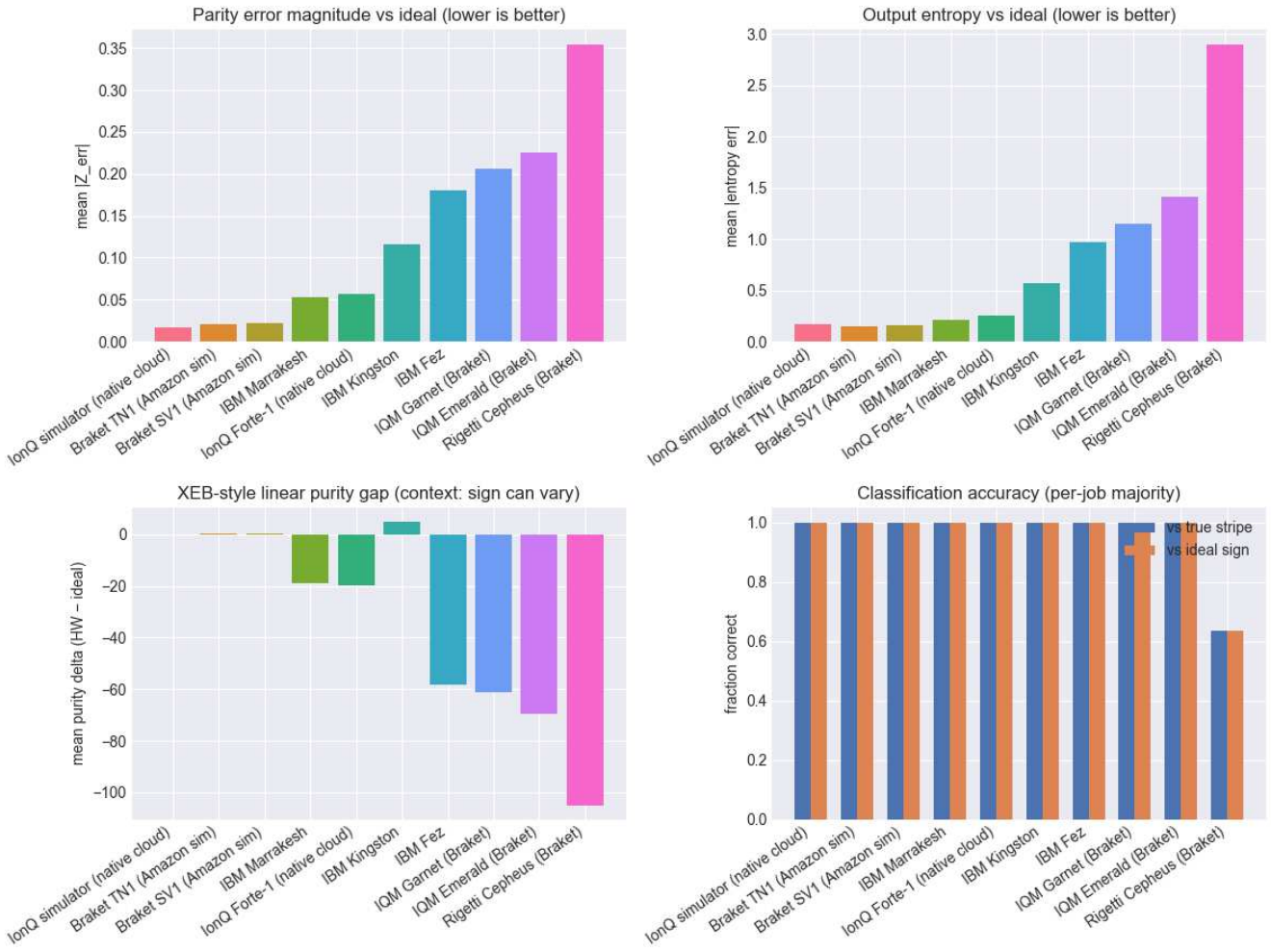


Figure 13. Comparative overview tested quantum backends in performing VQC circuits, including the XEB purity assessment

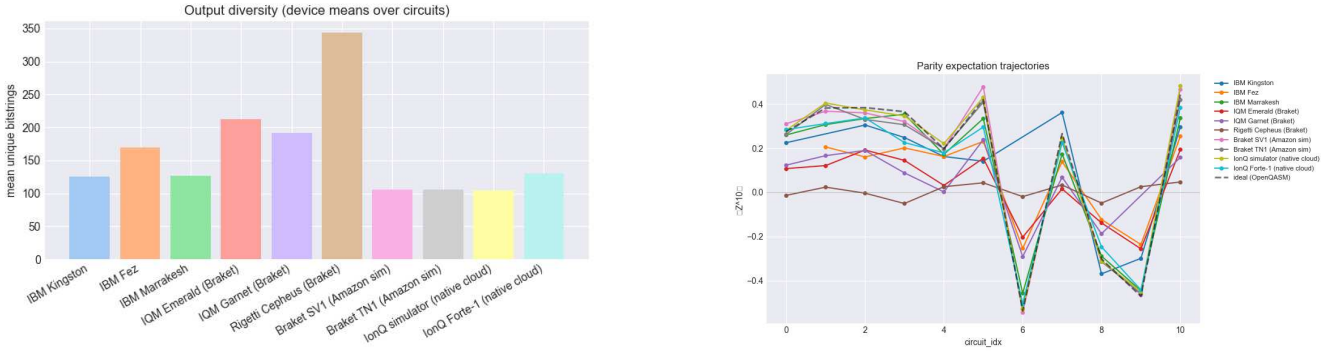
4.2 Results

Figure 13 demonstrates the performance of 10 tested quantum devices following the same metrics as have been discussed previously. All devices, apart from Rigetti Cepheus-108Q, have displayed 100% accuracy, but, at the same time, high entropy and parity error magnitude compared to ideal of up to 35%. This is further supported in Figure 14a: the higher number of unique bit-strings means less certainty in the correct output, which suggests that the circuit is less likely to come to the correct outcome.

However, it is important to note that the lack of certainty, mostly due to the noisy nature of the hardware, does not condemn the device. In the NISQ era, we would expect for the devices to have a degree of entropy, but it is promising to see meaningful outcomes dominate over the noisy floor. The parity deviation from the ideal diagonal in Figure 15: the device with higher entropy deviate further, with Cepheus flowing the furthest.

Since all the device ran the same circuits in their respective jobs, it is valuable to compare the jobs between themselves. In figure 16, most jobs have a consistent and reasonable parity standard deviation, and jobs 6 and 9 have the highest divergence, which is interesting because they both have the -1 prediction. The core reason for it is that the -1 jobs output a different parity signature rather than the +1 jobs that represent the majority. The +1 jobs outnumbered the -1 jobs in the training set, making the circuit biased towards the +1 toward, thus producing even-parity output distributions.

There is a secondary reason specific to superconducting qubits (represented here widely with IBM, IQM and Rigetti devices): the NISQ hardware asymmetry pushing outputs toward higher Hamming weights. The hardware itself has a slight structural bias toward even parity. When the true label is -1, the circuit goes against that bias as well as the hardware settings. This brings all the more weight to the system's and the circuit's sensitivity to follow the the data even in a bit-flip scenario, but it does imply higher variance in the cross-shot parity.



(a) Unique bitstring count per device

(b) Parity expectation trajectories per device

Figure 14. Unique bitstring count and parity expectations over VQC

When it comes to parity error matrix in Figure 17b, there doesn't seem to be any meaningful difference between jobs apart from the -1 jobs having the parity sign reversed - a natural outcome.

In Figure 14b, there is particular pattern the parity trajectory is following. This trajectory is determined by the reading outputs, which is why the ideal graph has a roughly consistent line at positive 0.4 for jobs with "1" output and sharp troughs for the "-1" jobs (6, 8, 9). Parity doesn't exceed 0.4, revealing confidence levels of the circuit. IBM Marrakech and IonQ Forte-1, as well as the simulators, follow the ideal very closely. IQM devices and IBM Fez follow the shape of the line, but do not maintain same amplitudes. Rigetti Cepheus oscillates lowly around the 0.0 line, showing an almost random behavior, and IBM Kingston misses one of the troughs.

4.3 Discussion

The majority of devices achieves a 100% accuracy with high entropy, demonstrating the NISQ devices can solve simple classification tasks correctly but noisily. This could lead to failure on harder tasks where the signal-to-noise margin is smaller. It is also evident that the ranking of device performance being metric-invariant, suggesting holistically better performance by certain devices. The performance hierarchy is task-agnostic at this circuit depth and qubit count, leading to the conclusions that overall noise floor/calibration quality dominates over circuit-specific factors. For example, Rigetti Cepheus not only had the highest parity error, but also was the only one with less than 100% accuracy and had the highest mean distance from the expectation diagonal across its jobs.

While IQM Garnet and Emerald performed similarly, as was the case with QAOA, IBM devices showed higher discrepancies in their behaviour. IonQ Forte-1 has strongly outperformed it's peers, following the patterns only slightly diverging from the simulators. With IonQ being the only representative from the trapped-ion modality, it is hard to conclude whether this finding implies the trapped-ion modality being the most suited for VQC tasks from a single data point. However, it is coherent with the trapped-ion modalities demonstrating strong gate fidelities, long coherence times and all to all-to-all connectivity important for VQC([35]).

5 Discussion of combined results

QAOA and VQC share the same hybrid variational structure (Section 3) that underlies essentially all near-term QML models. Read jointly, the two experiments show that device ranking is stable across this shared structure regardless of the specific loss function being optimized (MaxCut expectation versus classification parity). The devices that produced the highest expected MaxCut value and success probability in QAOA (IonQ Forte-1, IBM Kingston, the simulators) are the same devices that produced the lowest parity error and entropy deviation in VQC. Since QAOA and VQC stress different aspects of a device — QAOA depends on iterative two-qubit entangling depth to encode graph structure, while the VQC circuit here is shallower and more reliant on single-qubit rotation fidelity and readout — this consistency suggests that whatever separates the top and bottom performers is not circuit-specific. This behaviour demonstrates disparity stemming from global noise floor or calibration quality that dominates over algorithm-specific effects such as entangling gate count or mixer structure. This has a direct implication for the CLOPS finding. A QML training loop requires many circuit evaluations per optimization step, so the real-time cost of each shot batch compounds across iterations far more than it would in a single inference call. IBM Kingston's throughput advantage over Fez and Marrakech is therefore not just a per-circuit fidelity effect but a plausible trainability lever: less accumulated decoherence per iteration should translate into cleaner gradient estimates during classical parameter updates, which is precisely the kind of hardware property that the accuracy metric alone cannot expose but which

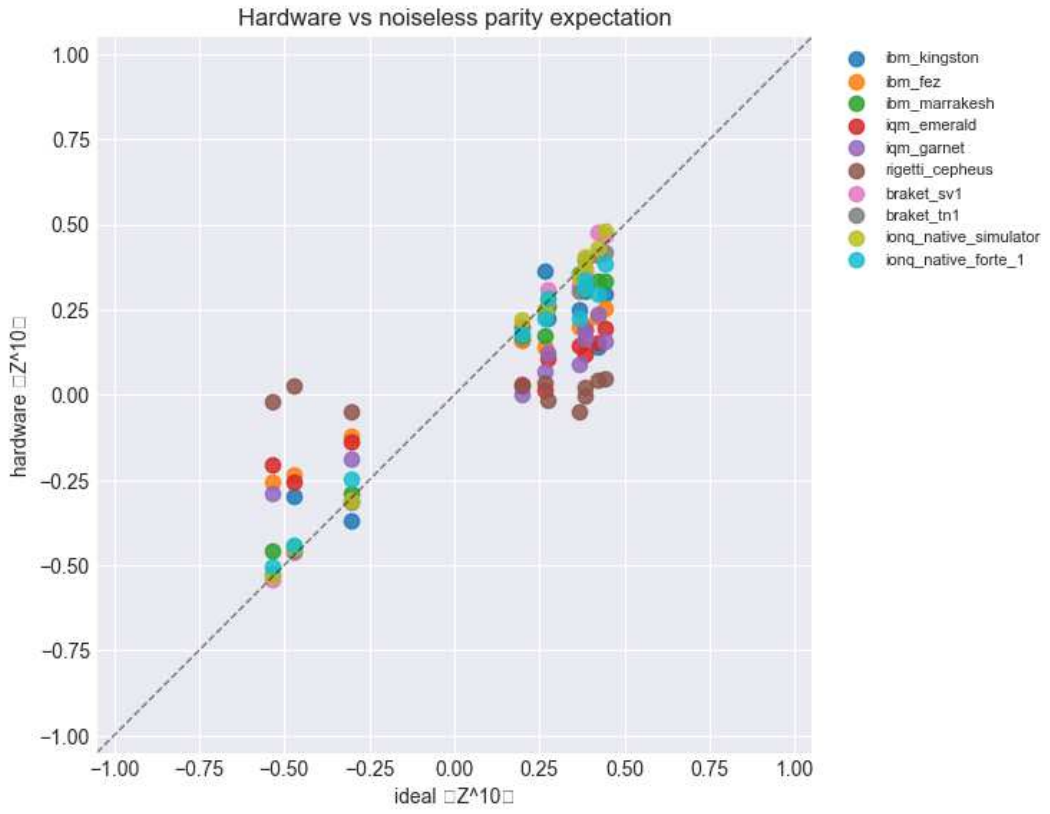


Figure 15. Noiseless expectation diagonal and hardware jobs deviations from it

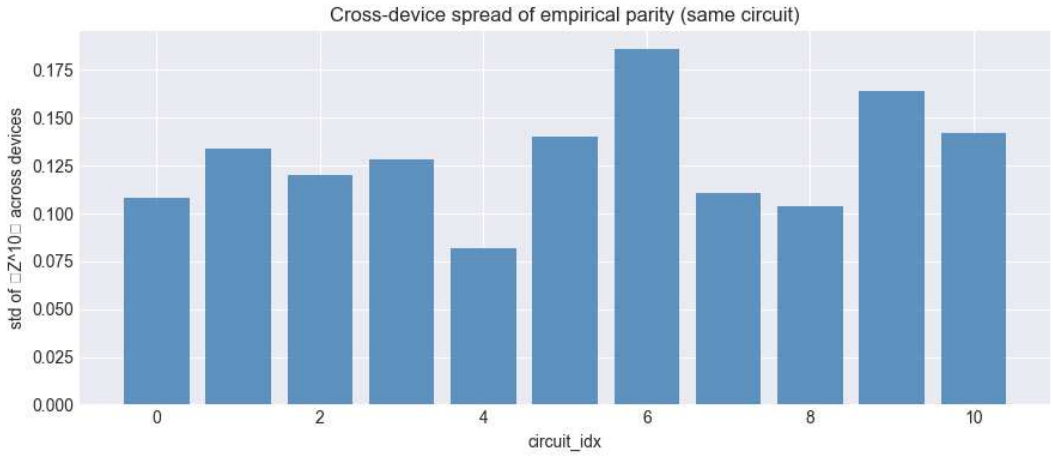


Figure 16. Comparison on the job by parity standard deviation across devices



(a) Parity error distribution between jobs

(b) Parity error heat map by device and job

Figure 17. Error distribution in the VQC runs across devices

matters most for whether a variational loop converges usefully rather than stalling in noise. The same-family IBM Heron discrepancy (Kingston versus Fez/Marrakech) versus the tight IQM Garnet/Emerald agreement across both algorithms is the more informative asymmetry. IQM devices are nominally the same architecture and track each other within 5% on both tasks, whereas the Heron-family IBM devices diverge substantially despite similar architecture. Architecture alone, therefore, is not enough to predict performance, although does serve as one of the key factors. Daily calibrations (supported by the small, but non-zero discrepancy between the Amazon Braket and IonQ Cloud runs on the IonQ Forte-1), appears to contribute as well. It is also notable that the pure modality does not appear to play a key role, exemplified by the fact that IBM, Rigetti and IQM are all superconducting modalities, but display vastly ranging results.

The dissociation between "task accuracy" and "distributional quality" is the second cross-cutting theme. In VQC, nearly every hardware device except Cepheus reached 100% classification accuracy, yet entropy and parity-error-magnitude differences of up to 35% relative to ideal persisted. In QAOA, no device exceeded 25% probability of landing on the exact optimal cut, yet expected MaxCut value and bitstring concentration (Figures 1, 3, 4) still cleanly separated devices. This confirms the paper's stated premise: binary success/accuracy is an insufficient metric on its own, since it saturates quickly for simple problems like the horizontal/vertical line classification of VQC and is too strict to be discriminating for harder tasks like QAOA. Distributional metrics — entropy, top-k bitstring mass, parity deviation - at the same time remain sensitive across the full performance range in both regimes. For QML specifically, this matters because a classifier that is "100% accurate but high-entropy" is fragile: it is arriving at the right label without confidently excluding the wrong one, which the paper's own confidence-margin and parity-trajectory analyses (Figures 17b, 14a) show directly, and which would be expected to break down as circuit depth or class separability decreases.

6 Conclusions

This benchmarking exercise across up to 13 devices and two canonical QML-adjacent workloads supports three conclusions. First, the relative ranking of devices is broadly stable across algorithm types at this qubit count and depth, indicating that overall noise floor and calibration quality, rather than pure modality and architecture, are the dominant determinant of NISQ performance in this regime. Secondly, architecture is a weak predictor of that ranking in isolation: same-vendor, same-family devices can diverge as much as, or more than, cross-vendor comparisons (IBM Heron family versus IQM Garnet/Emerald), implicating daily calibration state and possibly execution speed as at least co-equal factors. Third, binary accuracy is an inadequate standalone benchmark for either algorithm class — it saturates in VQC and is overly strict in QAOA — so entropy, bitstring concentration, and parity/expectation-value deviation from ideal are necessary complementary metrics to characterize a device's actual usability for downstream QML tasks, particularly because they capture trainability-adjacent behavior (susceptibility to noise-induced randomization) that a single pass/fail label cannot.

The clearest limitation of this study is that each device was sampled once (or twice, for IonQ Forte-1), so architecture-versus-calibration effects cannot be formally separated from day-to-day drift without repeated-measures data across multiple calibration cycles. The IonQ Forte-1 cross-platform comparison is suggestive but is a single data point. The circuits tested are also relatively simple, a deliberate decision made to confidently compare vastly different devices at early development stage. Future work should prioritize repeated longitudinal runs per device to quantify the calibration-drift component directly, and test whether the entropy/parity metrics used here retain their discriminating power as problem difficulty increases toward the barren-plateau regime flagged in the literature review. It could also be interesting to expand the benchmarking into more individual and architecturally different trapped-ion devices as well as photonic quantum computing infrastructure.

7 Data Availability

The datasets generated and/or analysed during the current study are available in the GitHub repository: https://github.com/vlcode110/Benchmarking-analysis-of-various-Quantum-computing-hardware-units-for-VQC-through-QAOA and archived in Zenodo at 10.5281/zenodo.23723053.

8 Acknowledgement

We thank IonQ, Inc. for providing the research team with credits for running the algorithms tested on their hardware through the IonQ Cloud platform.

9 Funding

This research received no specific grant from any funding agency in the public, commercial or not-for-profit sectors. IonQ, Inc. provided in-kind support in the form of cloud computing credits for access to its quantum hardware; IonQ had no role in the study design, data collection, analysis, interpretation of results or the decision to submit the manuscript.

References

1. Bhoite, R. V. & Sreekuttan, S. L. Benchmarking quantum algorithms on Xanadu, IBM, and Google quantum computers. In *2022 International Conference on Trends in Quantum Computing and Emerging Business Technologies (TQCEBT)*, 1–5, DOI: 10.1109/TQCEBT54229.2022.10041689 (2022).

2. Lolur, P., Rahm, M., Skogh, M., García-Álvarez, L. & Wendin, G. Benchmarking the variational quantum eigensolver through simulation of the ground state energy of prebiotic molecules on high-performance computers. 030005, DOI: 10.1063/5.0054915 (2021).

3. Lorenz, J. M. *et al.* Systematic benchmarking of quantum computers: status and recommendations. *arXiv* DOI: 10.48550/arXiv.2503.04905 (2025). 2503.04905.

4. Jałowiecki, K., Lewandowska, P. & Pawela, Ł. PyQBench: a Python library for benchmarking gate-based quantum computers. *SoftwareX* **24**, 101558, DOI: 10.1016/j.softx.2023.101558 (2023).

5. Proctor, T., Young, K., Baczewski, A. D. & Blume-Kohout, R. Benchmarking quantum computers. *Nat. Rev. Phys.* **7**, 105–118, DOI: 10.1038/s42254-024-00796-z (2025).

6. Meirom, D., Mor, T. & Weinstein, Y. Benchmarking quantum computers via protocols. *arXiv* DOI: 10.48550/arXiv.2505.12441 (2025). 2505.12441.

7. Miessen, A., Egger, D. J., Tavernelli, I. & Mazzola, G. Benchmarking digital quantum simulations above hundreds of qubits using quantum critical dynamics. *PRX Quantum* **5**, 040320, DOI: 10.1103/PRXQuantum.5.040320 (2024).

8. Mesman, K. J., van der Schoot, W., Möller, M. & Neumann, N. M. P. QuAS: Quantum application score for benchmarking the utility of quantum computers. *arXiv* DOI: 10.48550/arXiv.2406.03905 (2024). 2406.03905.

9. Hashim, A. *et al.* Practical introduction to benchmarking and characterization of quantum computers. *PRX Quantum* **6**, 030202, DOI: 10.1103/PRXQuantum.6.030202 (2025).

10. IonQ. Supercharging AI with quantum computing: Quantum-enhanced large language models. Accessed: Jun. 18, 2026.

11. The living QAOA reference. Accessed: May 22, 2026.

12. Cerezo, M. *et al.* Variational quantum algorithms. *Nat. Rev. Phys.* **3**, 625–644, DOI: 10.1038/s42254-021-00348-9 (2021).

13. Benedetti, M., Lloyd, E., Sack, S. & Fiorentini, M. Parameterized quantum circuits as machine learning models. *Quantum Sci. Technol.* **4**, 043001, DOI: 10.1088/2058-9565/ab4eb5 (2019).

14. Du, Y. *et al.* Quantum machine learning: A hands-on tutorial for machine learning practitioners and researchers. *arXiv* DOI: 10.48550/arXiv.2502.01146 (2025). 2502.01146.

15. Cosentino, A. *et al.* Metriq: A collaborative platform for benchmarking quantum computers. *arXiv* DOI: 10.48550/arXiv.2603.08680 (2026). 2603.08680.

16. Cirillo, F. & Esposito, C. Quantum machine learning for intrusion detection on noisy quantum computers. In *2025 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 01, 1665–1675, DOI: 10.1109/QCE65121.2025.00184 (2025).

17. McClean, J. R., Boixo, S., Smelyanskiy, V. N., Babbush, R. & Neven, H. Barren plateaus in quantum neural network training landscapes. *Nat. Commun.* **9**, 4812, DOI: 10.1038/s41467-018-07090-4 (2018).

18. Larocca, M. *et al.* Barren plateaus in variational quantum computing. *Nat. Rev. Phys.* **7**, 174–189, DOI: 10.1038/s42254-025-00813-9 (2025).

19. Farhi, E., Goldstone, J. & Gutmann, S. A quantum approximate optimization algorithm. *arXiv* DOI: 10.48550/arXiv.1411.4028 (2014). 1411.4028.

20. Farhi, E., Goldstone, J. & Gutmann, S. A quantum approximate optimization algorithm applied to a bounded occurrence constraint problem. *arXiv* DOI: 10.48550/arXiv.1412.6062 (2015). 1412.6062.

21. Herrman, R., Lotshaw, P. C., Ostrowski, J., Humble, T. S. & Siopsis, G. Multi-angle quantum approximate optimization algorithm. *Sci. Reports* **12**, 6781, DOI: 10.1038/s41598-022-10555-8 (2022).

22. Larkin, J., Jonsson, M., Justice, D. & Guerreschi, G. G. Evaluation of QAOA based on the approximation ratio of individual samples. *Quantum Sci. Technol.* **7**, 045014, DOI: 10.1088/2058-9565/ac6973 (2022).

23. Giovagnoli, A. An introduction to the quantum approximate optimization algorithm (2025). 2511.18377.

24. Gaidai, I. & Herrman, R. Performance analysis of multi-angle QAOA for $p > 1$. *Sci. Reports* **14**, 18911, DOI: 10.1038/s41598-024-69643-6 (2024).

25. Zhou, L., Wang, S.-T., Choi, S., Pichler, H. & Lukin, M. D. Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Phys. Rev. X* **10**, 021067, DOI: 10.1103/PhysRevX.10.021067 (2020).

26. Schuld, M. & Petruccione, F. *Supervised Learning with Quantum Computers* (Springer, 2018).

27. Google Quantum AI. QAOA: Max-cut. Accessed: May 22, 2026.

28. Amazon Braket. Compare Amazon Braket simulators. Accessed: May 22, 2026.

29. Amazon Braket. Submitting quantum tasks to simulators. Accessed: May 22, 2026.

30. OpenQAOA. Rigetti quantum virtual machine. Accessed: May 22, 2026.

31. IonQ. The value of classical quantum simulators. Accessed: May 22, 2026.

32. Quantinuum. Quantinuum emulators. Accessed: May 22, 2026.

33. Mayo, N., Mor, T. & Weinstein, Y. Benchmarking quantum computers via protocols, comparing IBM's Heron vs IBM's Eagle. *arXiv* DOI: 10.48550/arXiv.2603.04377 (2026). 2603.04377.

34. IBM Quantum. QVCs and QNNs. Accessed: May 22, 2026.

35. IonQ. IonQ technology (2026). Accessed: Jul. 09, 2026.

36. Wang, Z., Hadfield, S., Jiang, Z. & Rieffel, E. G. Quantum approximate optimization algorithm for maxcut: A fermionic view. *Phys. Rev. A* **97**, 022304, DOI: 10.1103/PhysRevA.97.022304 (2018).

10 Appendix

10.1 Circuit definitions[36]

```
OPENQASM 2.0;
include "qelib1.inc";

gate cost_edge(gamma) a,b {
    cx a,b;
    rz(2*gamma) b;
    cx a,b;
}

// Mixer term for a single qubit: RX(2*beta).
gate mixer(beta) q {
    rx(2*beta) q;
}

// 10-qubit register for the ring graph and 10-bit classical register
qreg q[10];
```

```
creg c[10];

// 1. Initialize in |+>^10
h q[0];
h q[1];
h q[2];
h q[3];
h q[4];
h q[5];
h q[6];
h q[7];
h q[8];
h q[9];


// ------------------------------
// Layer 1 (gamma1, beta1)
// All QAOA angles taken from Wang et al. [36]
// analytic / numeric optima for ring of disagrees at p=2
// gamma1 = 0.2052*pi, beta1 = 0.1026*pi
// ------------------------------
// 2a. Cost layer 1 for MaxCut on a 10- vertex ring
cost_edge(0.2052*pi) q[0], q[1];
cost_edge(0.2052*pi) q[1], q[2];
cost_edge(0.2052*pi) q[2], q[3];
cost_edge(0.2052*pi) q[3], q[4];
cost_edge(0.2052*pi) q[4], q[5];
cost_edge(0.2052*pi) q[5], q[6];
cost_edge(0.2052*pi) q[6], q[7];
cost_edge(0.2052*pi) q[7], q[8];
cost_edge(0.2052*pi) q[8], q[9];
cost_edge(0.2052*pi) q[9], q[0];



// 2b. Mixer layer 1
mixer(0.1026*pi) q[0];
mixer(0.1026*pi) q[1];
mixer(0.1026*pi) q[2];
mixer(0.1026*pi) q[3];
mixer(0.1026*pi) q[4];
mixer(0.1026*pi) q[5];
mixer(0.1026*pi) q[6];
mixer(0.1026*pi) q[7];
mixer(0.1026*pi) q[8];
mixer(0.1026*pi) q[9];



// ------------------------------
// Layer 2 (gamma2, beta2)
// ------------------------------

// 3a. Cost layer 2
cost_edge(0.3974*pi) q[0], q[1];
cost_edge(0.3974*pi) q[1], q[2];
cost_edge(0.3974*pi) q[2], q[3];
cost_edge(0.3974*pi) q[3], q[4];
cost_edge(0.3974*pi) q[4], q[5];
cost_edge(0.3974*pi) q[5], q[6];
cost_edge(0.3974*pi) q[6], q[7];
cost_edge(0.3974*pi) q[7], q[8];
cost_edge(0.3974*pi) q[8], q[9];
cost_edge(0.3974*pi) q[9], q[0];



// 3b. Mixer layer 2
mixer(0.2948*pi) q[0];
mixer(0.2948*pi) q[1];
```

```
mixer(0.2948*pi) q[2];
mixer(0.2948*pi) q[3];
mixer(0.2948*pi) q[4];
mixer(0.2948*pi) q[5];
mixer(0.2948*pi) q[6];
mixer(0.2948*pi) q[7];
mixer(0.2948*pi) q[8];
mixer(0.2948*pi) q[9];



// 4. Measure in the computational basis
measure q -> c;
```

Listing 1. QAOA circuit definition

```

OPENQASM 2.0;
include "qelib1.inc";

qreg q[10];
creg meas[10];

h q[0];
h q[1];
h q[2];
h q[3];
h q[4];
h q[5];
h q[6];
h q[7];
h q[8];
h q[9];

u1(1.5556244972704971) q[0];
u1(pi) q[1];
u1(pi) q[2];
u1(0.0003731011605266864) q[3];
u1(0.29134016508307764) q[4];
u1(0.6031290679077852) q[5];
u1(1.4649088041311724) q[6];
u1(0.11000240455465495) q[7];
u1(0.014899684301701563) q[8];
u1(0.08275557603449549) q[9];

h q[0];
h q[1];
h q[2];
h q[3];
h q[4];
h q[5];
h q[6];
h q[7];
h q[8];
h q[9];

u1(1.5556244972704971) q[0];
u1(pi) q[1];
u1(pi) q[2];
u1(0.0003731011605266864) q[3];
u1(0.29134016508307764) q[4];
u1(0.6031290679077852) q[5];
u1(1.4649088041311724) q[6];
u1(0.11000240455465495) q[7];
u1(0.014899684301701563) q[8];
u1(0.08275557603449549) q[9];

ry(2.5225848505738764) q[0];
ry(5.589237354288658) q[1];
ry(5.772230445645033) q[2];
```

```
ry(4.783195011144127) q[3];
ry(2.593696861340901) q[4];
ry(-0.604486587833009) q[5];
ry(0.47389770993912156) q[6];
ry(5.821278882943438) q[7];
ry(3.386885254036975) q[8];
ry(5.670996835252272) q[9];

cx q[0],q[1];
cx q[1],q[2];
cx q[2],q[3];
cx q[3],q[4];
cx q[5],q[6];
cx q[6],q[7];
cx q[7],q[8];
cx q[8],q[9];

rx(0.05279726604313351) q[0];
rx(6.269951384408505) q[1];
rx(6.381684306215958) q[2];
rx(3.1376474414714326) q[3];
rx(0.3121376410336459) q[4];
rx(3.007938989864461) q[5];
rx(3.020603451616161) q[6];
rx(3.2705543255521654) q[7];
rx(3.166855470096143) q[8];
rx(2.81006705283337) q[9];

barrier q;

measure q[0] -> meas[0];
measure q[1] -> meas[1];
measure q[2] -> meas[2];
measure q[3] -> meas[3];
measure q[4] -> meas[4];
measure q[5] -> meas[5];
measure q[6] -> meas[6];
measure q[7] -> meas[7];
measure q[8] -> meas[8];
measure q[9] -> meas[9];
```

Listing 2. VQC sample circuit