Robust implementation of generative modeling with parametrized quantum circuits

Vicente Leyton-Ortega1,2 · Alejandro Perdomo-Ortiz3,2,4 · Oscar Perdomo5,2

Received: 21 February 2020 / Accepted: 8 February 2021 / Published online: 3 May 2021
© The Author(s), under exclusive licence to Springer Nature Switzerland AG 2021

Abstract
Although the performance of hybrid quantum-classical algorithms is highly dependent on the selection of the classical optimizer and the circuit ansätze (Benedetti et al, npj Quantum Inf 5:45, 2019; Hamilton et al, 2018; Zhu et al, 2018), a robust and thorough assessment on-hardware of such features has been missing to date. From the optimizer perspective, the primary challenge lies in the solver’s stochastic nature, and their significant variance over the random initialization. Therefore, a robust comparison requires one to perform several training curves for each solver before one can reach conclusions about their typical performance. Since each of the training curves requires the execution of thousands of quantum circuits in the quantum computer, such a robust study remained a steep challenge for most hybrid platforms available today. Here, we leverage on Rigetti’s Quantum Cloud Services (QCSTM) to overcome this implementation barrier, and we study the on-hardware performance of the data-driven quantum circuit learning (DDQCL) for three different state-of-the-art classical solvers, and on two-different circuit ansätze associated to different entangling connectivity graphs for the same task. Additionally, we assess the gains in performance from varying circuit depths. To evaluate the typical performance associated with each of these settings in this benchmark study, we use at least five independent runs of DDQCL towards the generation of quantum generative models capable of capturing the patterns of the canonical Bars and Stripes dataset. In this experimental benchmarking, the gradient-free optimization algorithms show an outstanding performance compared to the gradient-based solver. In particular, one of them had better performance when handling the unavoidable noisy objective function to be minimized under experimental conditions.

Keywords NISQ · Quantum machine learning · Generative modeling · Unsupervised machine learning · Quantum circuit Born machine

1 Introduction
With the advent of several quantum computing technologies available to date, a significant effort is devoted to finding algorithmic strategies to cope with the noise in such early hardware architectures. In this domain, hybrid quantum-classical (HQC) algorithms, such as the variational quantum eigensolver (VQE) (Peruzzo et al. 2014; McClean et al. 2016) and the quantum approximate optimization algorithm (QAOA) (Farhi and Goldstone 2014), provide ways to use quantum computing hardware for practical applications. On the other hand, characterization of these so-called noisy-intermediate scale quantum (NISQ) devices is also one of the significant endeavors, since it paves the way to realizing the power of these devices. Covering both of these aspects, we had introduced quantum circuit-based generative models referred as quantum circuit Born machines (QCBMs) (Benedetti et al. 2019) and which can be trained with the so-called data-driven quantum circuit learning (DDQCL) algorithm. Training of these quantum generative models provide not only a framework to probe the power of NISQ devices on a useful machine learning
setting (i.e., the case of generative modeling in unsupervised machine learning) but also providing a way to measure the power of HQC variants. Some of these might include but not be limited to the choice of the circuit ansatz where the entangling connectivity layout and type of gates need to be specified, as well as the selection among different optimizers to be used in the classical processing side.

Although used for different tasks, DDQCL and other variational algorithms approaches have common components that need to be fine-tuned towards a successful experimental implementation (see Fig. 1). The quantum processing side is represented by a parametrized quantum circuit (PQC) whose parameters are updated via an optimizer running on a classical processing unit. Fine tuning of both of these components is essential to the performance of the overall HQC implementation.

Experimental implementations of QCBMs and of DDQCL where the entire learning process performed on-hardware have been recently demonstrated in both, superconducting qubits (Hamilton et al. 2018) and ion-trap quantum computers (Zhu et al. 2018). In the superconducting qubits implementation, emphasis was given to the exploration of different circuit ansätze, while in the ion-trap experiments, in addition to consideration of different proposals for the quantum processing unit (QPU) side of the pipeline, emphasis was given as well to the importance of the optimizer used.

[Diagram of Hybrid Quantum-Classical algorithm]

Each of the independent training curves of DDQCL and any HQC algorithm requires several iterations which correspond to the updates of the PQC. This procedure, in turn, represents typically thousands of circuits that need to be executed in hardware, per training curve. As seen in any of the aforementioned experimental implementation of DDQCL on-hardware, usually only one learning curve is performed, for each of the components to be optimized, e.g., circuit ansätze or optimizer. As shown in this work and elsewhere (Benedetti et al. 2019), for most of the optimizers previously considered in the literature, the variance from the solver’s random initialization is one of the most significant sources of uncertainty when reporting the performance of the HQC algorithm. Thus, one successful run does not necessarily correlate with the typical performance of the solver; none of the studies to date seems to afford multiple runs per each of the settings studied. To assess the impact of any choice within the HQC algorithm and to draw robust conclusions about the impact of several entangling connectivities or the performance of any optimizer against another, several runs per setting need to be performed.

In this work, we present the first experimentally robust comparison of each of the aspects of HQC algorithms, using the training of QCBMs as the working framework since it provides a quantitative figure of merit to assess for the performance of each of the knobs explored. To achieve this systematic and robust study, we leverage on Rigetti’s Quantum Cloud Services (QCS™) and its enhanced framework for handling the requirements of HQC algorithms efficiently. Some unique features of this platform include pre-compilation of the PQC programming cycles, active reset allowing for faster repetition rates, and low-latency due to co-location of the quantum and classical hosts.

2 DDQCL execution on Rigetti’s QPU

As in previous on-hardware implementations of DDQCL (Hamilton et al. 2018; Zhu et al. 2018), for the machine learning task, we consider a small artificial classical dataset, Bars and Stripes (BAS), which consists of patterns of \( n \times m \) binary images of bars or stripes, and denoted here as \( (n, m)\text{BAS} \). As shown in Fig. 2, the \((2,2)\text{BAS}\) dataset contains six different binary images. We encode the binary images into bitstrings, associating dark pixels to “1” and white pixels to “0”, i.e. \( \Box = 0 \) and \( \square = 1 \). In Fig. 2, we present the patterns for \((2,2)\text{BAS}\) and the label convention to use for binary image encoding. Following that convention the \((2,2)\text{BAS}\) dataset contains \( \{0000, 1010, 0101, 0011, 1100, 1111\} \) that leads to a simple distribution \( P_X(x) = 1/6 \) for \( x \in (2,2)\text{BAS} \). The goal for the circuit learning approach is to prepare a state \( |\psi\rangle = \sum_{x \in \{0,1\}^2} a_x |x\rangle \) with \( |a_x|^2 = 1/6 \) for \( x \in (2,2)\text{BAS} \) and
$|\alpha_x|^2 = 0$ otherwise. As shown in Ref. (Benedetti et al. 2019), encoding this classical dataset into a QCBM requires significant quantum resources in the form of entanglement, since the entropy of entanglement, $S_{(2,2)\text{BAS}}$, of the set of potential quantum states capturing the distribution over these patterns is $1.25 < S_{(2,2)\text{BAS}} \leq S_{\text{max}}$, with $S_{\text{max}} \approx 1.79248$ is the maximum entanglement entropy known for any 4-qubit states. (Higuchi and Sudbery 2000). Note $S = 1.0$ corresponds to the entanglement entropy of the 4-qubit GHZ state. Other challenging probability distributions constructed from real-world data have been recently explored in Refs. (Alcazar et al. 2020; Coyle et al. 2020).

For the experimental implementation of DDQCL, we consider a 4-qubit subunit connected in star and line entangling topology, according to the connectivity constraints in the 16-qubit Aspen quantum computer (see insets in Fig 3). The main building blocks for the construction of the quantum circuit ansätze (see, e.g. top right in Fig. 3) relies on two observations. (i) The use of CZ entangling gates is native to the Rigetti QPU. This sets our preference towards these gates over either CNOTs or Mølmer-Sørensen gates which were used in previous DDQCL implementations on either the IBM Tokyo (Hamilton et al. 2018) or ion-trap (Zhu et al. 2018) quantum computer, respectively. (ii) Instead of using arbitrary single-qubit gates interspersed or alternating with two-qubit gates, we used here just $R_y(\theta)$ rotations. Given that all the entries of the $R_y(\theta)$ are real, along with the CZ gates produces quantum states with real amplitudes only. Note this is a not limitation towards the quantum model used in the generative task since it is enough to consider $\alpha_x \in \mathbb{R}$. In Appendix B.1, we provide more intuition of the layout for this circuit ansätze. This last choice of going from single-arbitrary rotations to just $R_y$ rotations is desirable since it reduces the number of parameter in the quantum model, therefore helping the classical optimizer in this high-dimensional search space.

The progress of this machine learning task is evaluated by the Jensen-Shannon divergence cost function, denoted by $D_{JS}$ (details in Appendix A), that compares the probabilistic distribution of the circuit output with the target distribution. With the learning score record, the quantum circuit parameters are updated by an optimizer running on a classical processor; this optimization in pursuit of a minimum score is the classical part of the generative model approach.

3 Results

For our experimental and numerical benchmarking study, we used three solvers: the Zeroth-Order Optimization package (ZOOPT) (Liu et al. 2017), the Stochastic Variation of Hill-Climbing type algorithm (SVHC) (Perdomo and Wei 2021), and a classical stochastic gradient descent based solver (ADAM) (Kingma and Ba 2014) (for configuration details, see Appendix B.2).

We consider a specific region of the QPU to run the experiments (see QPU layout inside Fig. 3). For each of the settings to be explored, i.e., the choice of solver or the entangling connectivity topology or the varying depth of the circuit, we performed five independent DDQCL from random initializations of the parameters, and subsequently we bootstrap the outcome to obtain a more robust median for the final reported value of the $D_{JS}$ cost function (see Appendix C.1 for details). In each DDQCL, 2000 different realizations of the PQC were evaluated, and from each of them, 3000 computational basis measurements or shots were taken as readouts in the computational basis. Before every batch of these five independent runs, we performed the characterization of the transition matrix to be used for readout correction and use it to post-process the experimental histograms built from every 3000 shots (see Appendix B3 for details). In Fig. 3, we present the main results, where the best learned probabilistic model is depicted with the Kullback-Leibler divergence, $D_{KL}$, values 0.44 and 0.14 for line and star topology, respectively. Those values were reached using the ZOOPT optimizer. $D_{KL}$ corresponds to the Kullback-Leibler divergence and it is the gold standard when comparing the closeness between two probability distributions, with $D_{KL} = 0.0$ meaning that the quantum model and the target distribution match. Estimation of $D_{KL}$ from the experimental histogram also allows us to compare to previous experimental realizations (Hamilton et al. 2018; Zhu et al. 2018). From these, it can be seen that our DDQCL implementation on the Rigetti QPU ($D_{KL} = 0.14$) is better than the...
Experimental implementation of DDQCL for the training of 4-qubit QCBM models on the (2, 2) BAS and Stripes dataset, using different solvers and topologies. In the upper/lower panel (a/b) we show the bootstrapped median and 90% confidence interval over the distribution of medians of the $D_{JS}$ divergence as learning progresses for 2000 iterations, for two layers of entangling circuits with a line/star topology connectivity using ADAM, SVHC, and ZOOPT classical optimizers. Besides, in the panels are shown the quantum circuit model and the comparison between the probability distribution corresponding to the lowest cost and the target distribution. In that comparison, it is shown the Kullback-Leibler divergence cost to get to the target distribution from the QPU best result ($D_{KL} = 0.36$) (Hamilton et al. 2018) and comparable to the best results obtained in the trapped-ion quantum computer ($D_{KL} = 0.09$) (Zhu et al. 2018).

In addition to testing the performance of different types of optimizers, we consider different circuits depths. For these experiments, we consider a circuit with one round of entangling gates with 3 CZs and 10 local rotations, and the circuit used for the optimizers comparison corresponding to two rounds of entangling gates with 16 local rotations and 6 CZs (see $L_1$ and $L_2$ in Fig. 4, respectively). In star topology configuration, a significant improvement is observed with the addition of a second entangling round, while for the line topology configuration, there is significantly less improvement (or at least slower convergence) from the addition of a second entangling round. To understand this apparent faster convergence and better results from the star entangling topology compared to its line analog, we performed in silico simulations with the same optimizers. From these simulations on noiseless qubits (see Appendix C), it is hard to advocate for a clear advantage of the star connectivity over the line entangling ansatz. Therefore, this discrepancy in results between in silico simulations and the on-hardware ones is not to the topology itself, but most likely it is due to the quality of the qubits used for each experiment, with better quality for the subset used for the star experiments.

For completeness, in Table 1, we report the values for the qBAS score, calculated as detailed in Ref. (Benedetti et al. 2019).
In this figure, we show the bootstrapped median and 90% confidence interval over the distribution of medians of the $D_{JS}$ divergence as learning progresses for 2000 iterations, and the learning improvement with the circuit depth on the line and star topology. We define an entangling layer as a set of entangling gates $U_2$ overall qubits according to the topology. For line topology, after simplification, a layer for 4 qubits consists of 6 rotations $R_y$ and 3 CZs, see the quantum circuit over panel a. The same amount of rotations $R_y$ and CZ's stands for the star topology, see the quantum circuit over panel b. A relative improvement due to circuit depth is presented using star topology against the line topology, in which the variance of the learning curve for 1 layer overlaps the variance of the learning curve for 2 layers. In the experiments we consider qubits in similar regions of the device (see layouts inside) according to the topologies.

4 Summary and outlook

We performed a robust implementation and a comparison on-hardware of several components of primal importance affecting the performance of hybrid quantum-classical algorithms. The factors tested include the circuit entangling layout and its depth, the selection of the classical optimizer, and the impact of post-processing strategies such as readout correction.

Although results here show that the choice of each of these components affects the performance significantly, the study presented here is far from exhaustive. We considered ansätze that prepare an extensive set of 4-qubit QCBMs with real amplitudes, which under noiseless condition, have an arbitrary excellent performance in the learning of the $2 \times 2$ BAS dataset considered here. In contrast, these same ansätze and training settings have suboptimal performance which is limited by the different noise sources affecting the quantum processor output. Although we cannot disentangle the various noise sources, these application-inspired benchmarks offer valuable insights into how much device noise affects the performance of the machine in the generative model.

Based on our experiments, we show that to generate the $(2,2)$ BAS dataset, optimizing parameterized quantum circuits that assume the star topology using the ZOOPT solver yielded the best results. This result is contrary to our simulation predictions that suggested the use of SVHC to optimize the circuits. In addition to providing favorable performance, the use of ZOOPT allowed us to explore a wider range of ansätze and training settings, which is crucial for practical applications.

Table 1 Performance figures of merit from main experimental features explored in this work related to different optimizers and circuits ansätze (star versus line)

| Solver | $D_{KL}$ | qBAS score | $D_{KL}$ | qBAS score |
|--------|----------|------------|----------|------------|
| Line topology | | | | |
| ADAM | 0.76 | 0.61 | 0.783 | 0.61 |
| SVHC | 0.52 | 0.67 | 0.740 | 0.58 |
| ZOOPT | 0.46 | 0.76 | 0.497 | 0.75 |
| Star topology | | | | |
| ADAM | 0.55 | 0.70 | 0.579 | 0.66 |
| SVHC | 0.42 | 0.75 | 0.496 | 0.70 |
| ZOOPT | 0.14 | 0.89 | 0.186 | 0.86 |
experimental conditions for the DDQCL algorithm, more importantly, our work highlights the need for similarly robust benchmark studies (beyond only simulations) to facilitate experimental designs on near-term quantum computers.

Another interesting direction would be to try some recent extensions of DDQCL experimentally, proposing a gradient-based training approach (Liu and Wang 2018) or by adding ancillary qubits to enhance the power of the quantum generative model (Du et al. 2018). Additionally, it is known that optimizing each of the classical solvers in its own is a hard task, and it would be left to future work to address this selection, for example, by using an automatic routine for hyperparameter setting (see, e.g., (Snoek et al. 2012)).

For the small number of qubits and small circuits depths used in this manuscript, we are not affected by barren plateaus (McClean et al. 2018). The argument behind the existence of barren plateaus point to limitations hindering the trainability of PQCs. Such difficulties have been proven to affect the training of PQCs under the following four main assumptions: (1) random parameter initialization strategy, (2) gradient-based training methods, (3) circuits of significant depth (due to the 2-design conditions in the ansätze constructions), and (4) characteristics of the cost function with local preferred over global cost functions (for the latter please see Ref. (Cerezo et al. 2020)). In our work, we do not have circuits with an extensive depth and we are not using gradient-based methods either, and therefore we are not directly affected by the barren plateaus theorems proved to date. For the number of qubits, choice of cost function, and training details including the circuit ansätze and the classical optimizers used, we demonstrate here, numerically and experimentally, that all of our experiments succeeded in finding the expected global minima.

We hope hybrid approaches on concrete datasets and real-world applications, such as the generative modeling task highlighted here, become a more significant way to benchmark and measure the power of quantum devices. The qBAS score described in Ref. (Benedetti et al. 2019) and measured here for our experimental results represents a way to measure the power of these NISQ devices, beyond values reported for the fidelity of single and two-qubit gates. To move to a quantum ready stage, it is essential we move towards evaluating the capacity and performance of the device as a whole. We believe developing the infrastructure to develop robust benchmarking studies like the one presented here is an essential step towards a meaningful comparison among different algorithmic and hardware proposals.

Appendix A: DDQCL pipeline

In this work, we implement a hybrid quantum-classical algorithm for unsupervised machine learning tasks introduced in (Benedetti et al. 2019) at Rigetti’s superconducting quantum computer (QPU). In the following, we present a short introduction of the approach, for a complete discussion see (Benedetti et al. 2019). The algorithm generates a probability distribution model for a given classical dataset $X' = \{x^{(1)}, x^{(2)}, \ldots, x^{(D)}\}$ with distribution $P_{X'}$. Without loss of generality, the elements of $X'$ can be considered as $N$-dimensional binary vectors $x^{(i)} \in \{0,1\}^N$ for $i = 1, \ldots, D$, allowing a direct connection with the basis of an $N$-qubit quantum state, i.e., $x^{(i)} \rightarrow |x^{(i)}\rangle = |x_1^{(i)}, \ldots, x_N^{(i)}\rangle$ for $i = 1, \ldots, D$. The goal is to prepare a quantum state $|\psi\rangle = \sum_{x \in \{0,1\}^N} \alpha_x |x\rangle$, with a probability distribution that mimics the dataset distribution, $|\langle x | \psi \rangle|^2 = |\alpha_x|^2 = P_{X'}(x)$ for $x \in \{0,1\}^N$.

The required quantum state $|\psi\rangle$ is prepared by tuning a PQC composed of single rotations and entangling gates with fixed depth and gate layout. In general, the circuit parameters of amount $L$ can be written in a vector form $\theta = (\theta_1, \ldots, \theta_L)$, that prepares a state $|\psi_{\theta}\rangle$ with distribution $P_{\theta}$. The algorithm varies $\theta$ to get a minimal loss from $P_{\theta}$ to $P_{X'}$. Here, we consider the Jensen-Shannon divergence ($D_{JS}$) to measure the loss, and determine how to update $\theta$ using a classical optimizer.

The $D_{JS}$ divergence is a symmetrized and smoothed version of the Kullback-Leibler divergence ($D_{KL}$) (Weng 2019), defined as

$$D_{JS}(P|Q) = \frac{1}{2} D_{KL}(P|M) + \frac{1}{2} D_{KL}(Q|M),$$

where $P$ and $Q$ are distributions, and $M = (P + Q)/2$ their average. The Kullback-Leibler divergence is defined as

$$D_{KL}(X|M) = \sum_{s \in \{0,1\}^N} X(s) \ln (X(s)) - \sum_{s \in \{0,1\}^N} X(s) \ln (M(s)),$$

for $X \in \{P, Q\}$.

After the algorithm evaluates the cost from $P_{\theta}$ to $P_{X'}$, it follows a quantum circuit parameters update in pursuit to a minimal cost. This procedure is done by an optimizer that runs on a classical processor unit (CPU). In summary, the algorithm chooses a random set of parameters $\theta$, evaluates the cost from $P_{\theta}$ to the target $P_{X'}$ through the Jensen-Shannon divergence $D_{JS}(P_{\theta}|P_{X'})$, and updates $\theta$...
minimizing the cost. This procedure is repeated several times (iterations) until a convergence criterion is met, for instance, a maximum number of iterations.

Appendix B: Experimental details

B.1 Quantum circuit model design

Since the quantum state probabilistic distribution does not depend on local quantum state phases, we chose a layout of the model circuit that prepares real amplitude quantum states. We consider the following two-qubit ansätze as the base to design $N$-qubit the circuit layout for DDQCL,

$$U_2(\theta, \gamma, \beta)|00\rangle = R_0^\beta R_0^\gamma CZ_{01} R_0^\alpha |00\rangle = \sum_{s \in \{0,1\}^2} \alpha_s |s\rangle$$ (3)

therein $R_0^\alpha = \exp[-i \theta \sigma_y^s / 2]$ is a local rotation around $y$-axis in the Bloch sphere on the $k$th qubit and $CZ_{kl}$ the control phase shift gate between $k$th and $l$th qubits. The gate $U_2$ defines a one-to-one map $[0, 2\pi)^3 \rightarrow S^3$, that ensures the preparation of any 2 qubit state with real probability amplitudes (Perdomo et al. 2021), i.e. $\alpha_s \in \mathbb{R}$ for $s \in \{0,1\}^2$. For setups with more than 2 qubits, we use $U_2$ to entangle different pairs of qubits depending on their connectivity, until getting a fully controlled entangled state. In this study, we consider four qubits with different connections (topologies) according to the Rigetti’s quantum computer architecture. We designed several circuits for line and star topologies using up to $L = 16$ local rotations and 6 CZ’s (see Fig. 3).

In silico simulations on noiseless qubits show that this circuit layout can learn a quantum model capable of successfully reproducing the target probability from the $(2, 2)\text{BAS}$ (see Appendix C).

As pointed in the main text, one of the advantages of this circuit layout compared to the one considering arbitrary single-qubit rotations is the reduction in the number of parameters. On the other hand, it is important to note that having the additional parameters could in principle help the search since these open more paths in the Hilbert space, resulting in an increased number of states that could lead to a perfect $P_X$ (see e.g., the discussion in the “Entanglement entropy of BAS(2,2)” section in the Supp. Material of Ref. (Benedetti et al. 2019)). This trade-off between flexibility in the quantum model and difficulty in optimization is beyond the scope of this work, and it would be an interesting research direction to explore.

B.2 Classical optimizers

For the experimental and numerical benchmarking study, we consider the following solvers and configurations.

- **ZOOPT**: The Zeroth-Order Optimization algorithm is a package that collects state-of-the-art zeroth-order optimization methods. The algorithm used by ZOOPT does not use the gradient of the objective function, it learns from samples of the search space. This method is convenient for optimizing cost functions that are not differentiable, with several local minima like the cost function we used in this work. Due to the stochastic nature of the cost function, we consider a value-suppression setting in the ZOOPT configuration. Additionally, we configured the initial population or initial number of evaluations as $N_{\text{ini}} = 3L$, with $L$ as the number of parameters in the parametric quantum circuit.

- **SVHC**: This algorithm is the stochastic version of the ‘hill-climbing’ solver, with a variational searching size region. This algorithm is gradient-free, the criterion for new candidate solutions is based in samples of the search space. We consider an initial population $N_{\text{ini}} = 3L$, with $L$ as the number of parameters in the quantum circuit.

- **ADAM**: This algorithm is an extension of classical stochastic gradient-descent algorithm, a combination of Adaptive Gradient Algorithm and Root Mean Square Propagation. Its performance depends on the derivative implementation of the objective function. We consider the setup used in Ref. (Hamilton et al. 2018), with learning rate $\alpha = 0.2$, decay rates $\beta_1 = 0.9$, and $\beta_2 = 0.999$.

B.3 Improvement via readout correction

DDQCL is based on the probability distribution of the $N$ qubit quantum state, which is in turn transformed into resulting bitstrings $\{0, 1\}^N$, denoted as shots in the main text. Unfortunately, no device is 100% free of errors in this transformation from quantum state to bitstring output. Next, we describe a simple model to cope with this classical readout channel. Although there have been other procedures which are scalable (Kandala et al. 2017; Dumitrescu et al. 2018; Magnard et al. 2018), here we exploit our small number of qubits to make the least number of assumptions on the channel and perform an exponential number of experiments, which need to be done once to characterize the channel.

The simple error readout correction implemented here consist of the measurement characterization of all the $2^N$
Fig. 5 Comparison of DDQCL with and without readout correction. Full learning curve is shown (left) with and without readout correction (blue and green line, respectively). Without error correction, the minimal value reached for the KL divergence is $\sim 1.0$, which is closer to a random distribution, and far from the target one (see histograms on the upper right). On the other hand, the training curve with readout correction reached a value of $KL \sim 0.13$, which is a significant improvement in the DDQCL projectors $\Pi_x = |x\rangle\langle x|$, with $x \in \{0, 1\}^4$ for the (2,2)BAS realization here. Suppose you trivially prepare any of the $2^4$ states of the computational basis, e.g., $x = 0010$. In a noiseless scenario that measurement should yield a bitstring $x = 0010$ in the classical register for every single shot, allowing the computation of the $P_x = 1.0$, as expected from preparation. However, in the experiment the readouts correspond to different elements $y \in \{0, 1\}^4$ with a distribution $p(y|x)$ due to assignment errors. In our procedure, we use a large number of readouts (set to 10,000 shots) to compute the distribution $p(y|x)$, after trivial preparation of each $x \in \{0, 1\}^4$. Each of the $p(y|x)$

![STAR TOPOLOGY](image)

![LINE TOPOLOGY](image)

Fig. 6 DDQCL simulations in silico, and under the assumption of noiseless qubits, but taking into account the stochasticity from finite readouts. In these simulations we used 3000 shots per circuit.
any quantum state with population $P_x$ with a 90% confidence interval, as the lower and upper limits, respectively, accounting for a replacement and compute the median $D_{JS}$ in independent DDQCL into 10,000 datasets of five with observed error bars from the 5th and 95th percentiles of the distribution of 10,000 medians, we computed the median DJS. To recover the observed output distribution $P_y$ as $P_y = M P_x$. To recover the original distribution from the quantum state, we invert the relation between $P_x$ and $P_y$, i.e., $P_x = M^{-1} P_y$. Before each batch of five of learning curves, we calculate the inverse transformation $M^{-1}$ and apply this to the readout distribution; this defines a post-measurement process before the score and optimization steps.

To test the efficacy of this procedure, in Fig. 5, we present the training with the bare and the readout corrected training. The latter shows significant improvement with a score and optimization steps.

| Solver   | $D_{KL}$ | qBAS score | Median results |
|----------|----------|------------|----------------|
| Line topology |         |            |                |
| ADAM     | 0.72     | 0.62       | 0.800          |
| SVHC     | 0.02     | 0.93       | 0.022          |
| ZOOPT    | 0.01     | 0.97       | 0.014          |
| Star topology |       |            |                |
| ADAM     | 0.38     | 0.83       | 0.553          |
| SVHC     | 0.01     | 0.96       | 0.037          |
| ZOOPT    | 0.01     | 0.96       | 0.048          |

is used as the $x$th column of the transition matrix $M$. Thus, any quantum state with population $P_x$ is related with actual observed output distribution $P_y$, as $P_y = M P_x$. To recover the original distribution from the quantum state, we invert the relation between $P_x$ and $P_y$, i.e., $P_x = M^{-1} P_y$. Before each batch of five of learning curves, we calculate the inverse transformation $M^{-1}$ and apply this to the readout distribution; this defines a post-measurement process before the score and optimization steps.

To test the efficacy of this procedure, in Fig. 5, we present the training with the bare and the readout corrected training. The latter shows significant improvement with a score and optimization steps.

To test the efficacy of this procedure, in Fig. 5, we present the training with the bare and the readout corrected training. The latter shows significant improvement with a score and optimization steps.

| Solver   | $D_{KL}$ | qBAS score | Median results |
|----------|----------|------------|----------------|
| Line topology |         |            |                |
| ADAM     | 0.72     | 0.62       | 0.800          |
| SVHC     | 0.02     | 0.93       | 0.022          |
| ZOOPT    | 0.01     | 0.97       | 0.014          |
| Star topology |       |            |                |
| ADAM     | 0.38     | 0.83       | 0.553          |
| SVHC     | 0.01     | 0.96       | 0.037          |
| ZOOPT    | 0.01     | 0.96       | 0.048          |

B.4 QPU specifics

In this work, we had access to the 16Q Aspen-1 quantum chip. This chip has $0.05\mu s$ and $0.15\mu s$ of coherence time for one and two qubit gates, respectively. With a $T_1 = 28.71\mu s$ and $T_2 = 20.34\mu s$, the quantum chip can support the quantum operations defined this work.

Appendix C: In silico simulations

C.1 Bootstrapping details

We resample the experimental outcome of the five independent DDQCL into 10,000 datasets of five with replacement and compute the median $D_{JS}$ for each. From the distribution of 10,000 medians, we computed the median and obtained error bars from the 5th and 95th percentiles as the lower and upper limits, respectively, accounting for a 90% confidence interval.

Acknowledgements The authors would like to thank Marcus P. da Silva for useful feedback and suggesting the readout correction used in our experiments here. Marcello Benedetti for general discussions on generative modeling and their evaluation, and the Rigetti’s QCS team for access and support during the beta testing of the Aspen processor. The authors also would like to thank the referees for all the suggestions to improve the Funding. V. Leyton-Ortega was supported as part of the ASCR Quantum Testbed Pathfinder Program at Oak Ridge National Laboratory under FWP #ERKJ332.

Declarations

Conflict of interest The authors declare no competing interests.

References

Benedetti M, Garcia-Pintos D, Perdomo O, Leyton-Ortega V, Nam Y, Perdomo-Ortiz A (2019) A generative modeling approach for benchmarking and training shallow quantum circuits. npj Quantum Inf 5:45

Hamilton KE, Dumitrescu EF, Poozer RC (2018) Generative model benchmarks for superconducting qubits. arXiv:1811.09905

Zhu D, Linke NM, Benedetti M, Landsman KA, Nguyen NH, Alderete CH, Perdomo-Ortiz A, Korda N, Garfoot A, Brecque C, Egan L, Perdomo O, Monroe C (2018) Training of quantum circuits on a hybrid quantum computer. arXiv:1812.08862

Peruzzo A, McClean J, Shadbolt P, Yung M-H, Zhou X-Q, Love PJ, Aspuru-Guzik A, O’Brien JL (2014) A variational eigenvalue solver on a photonic quantum processor, vol 5. EP – McClean JR, Romero J, Babbush R, Aspuru-Guzik A (2016) The theory of variational hybrid quantum-classical algorithms. New J Phys 18:023023

Farhi SGE, Goldstone J (2014) A quantum approximate optimization algorithm. arXiv:1411.4028

Higuchi A, Sudbery AW (2000) How entangled can two couples get? Phys Lett A 273:213–217

Alcazar J, Leyton-Ortega V, Perdomo-Ortiz A (2020) Classical versus quantum models in machine learning: insights from a finance application. Mach Learn Sci Technol 1:035003

Coyle B, Henderson M, Le JCI, Kumar N, Paini M, Kashefi E (2020) Quantum versus classical generative modelling in finance. Quantum Sci Technol

Liu Y-R, Hu Y-Q, Hu Q, Yu Y, Qian C (2017) Toolbox for derivative-free optimization. arXiv:1801.00329

Perdomo O, Wei F (2021) The blade flapping motion using a new induced velocity model. In preparation

Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv:1412.6980

Liu J-G, Wang L (2018) Differentiable learning of quantum circuit born machines. Phys Rev A 98:062324

Du Y, Hsieh M-H, Liu T, Tao D (2018) The expressive power of parameterized quantum circuits. arXiv:1810.11922

Snoek J, Larochelle H, Adams RP (2012) Practical bayesian optimization of machine learning algorithms. arXiv:1206.2944

McClean JR, Boixo S, Smelyanskiy VN, Babbush R, Neven H (2018) Barren plateaus in quantum neural network training landscapes. Nat Commun 9:4812

Cerezo M, Sone A, Volkoff T, Cincio L, Coles PJ (2020) Cost-function-dependent barren plateaus in shallow quantum neural networks. arXiv:2001.00550

Weng L (2019) From gan to wgan. arXiv:1904.08994
Perdomo O, Leyton-Ortega V, Perdomo-Ortiz A (2021) Entanglement
types for two-qubit states with real amplitudes. Quantum Inf
Process 20:89
Kandala A, Mezzacapo A, Temme K, Takita M, Brink M, Chow
JM, Gambetta JM (2017) Hardware-efficient variational quantum
eigensolver for small molecules and quantum magnets. Nature
549:242
Dumitrescu EF, McCaskey AJ, Hagen G, Jansen GR, Morris TD,
Papenbrock T, Pooser RC, Dean DJ, Lougovski P (2018) Cloud
quantum computing of an atomic nucleus. Phys Rev Lett
120:210501
Magnard P, Kurpiers P, Royer B, Walter T, Besse J-C, Gasparinetti
S, Pechal M, Heinsoo J, Storz S, Blais A, Wallraff A (2018) Fast
and unconditional all-microwave reset of a superconducting qubit.
Phys Rev Lett 121:060502

Publisher’s note Springer Nature remains neutral with regard to
jurisdictional claims in published maps and institutional affiliations.

Affiliations

Vicente Leyton-Ortega¹,² · Alejandro Perdomo-Ortiz³,²,⁴ · Oscar Perdomo⁵,²

¹ Computer Science and Engineering Division, Oak Ridge National
Laboratory, One Bethel Valley Road, Oak Ridge, TN 37831 USA
² Rigetti Computing, 2919 Seventh Street, Berkeley, CA 94710-
2704, USA
³ Zapata Computing Canada Inc., 325 Front St W, Toronto, ON,
M5V 2Y1, Canada
⁴ Department of Computer Science, University College London,
WC1E 6BT London, UK
⁵ Department of Mathematics, Central Connecticut State University,
New Britain, CT 06050, USA