Generative quantum learning of joint probability distribution functions

Elton Yechao Zhu,[1,*] Sonika Johri,[2,*] Dave Bacon,[2] Mert Esencan,[1] Jungsang Kim,[2] Mark Muir,[1] Nikhil Murgai,[1] Jason Nguyen,[2] Neal Pisenti,[2] Adam Schouela,[1] Ksenia Sosnova,[2] and Ken Wright[2]

[1]*Fidelity Center for Applied Technology, FMR LLC, Boston, Massachusetts 02210, USA*

[2]*IonQ Inc, 4505 Campus Dr, College Park, Maryland 20740, USA*

(Received 26 October 2021; accepted 1 September 2022; published 8 November 2022)

Modeling joint probability distributions is an important task in a wide variety of fields. One popular technique for this employs a family of multivariate distributions with uniform marginals called copulas. While the theory of modeling joint distributions via copulas is well understood, it gets practically challenging to accurately model real data with many variables. In this paper, we show that any copula can be naturally mapped to a multipartite maximally entangled state. Thus, the task of learning joint probability distributions becomes the task of learning maximally entangled states. We prove that a variational ansatz we christen as a "qopula" based on this insight leads to an exponential advantage over classical methods of learning some joint distributions. As an application, we train a quantum generative adversarial network (QGAN) and a quantum circuit Born machine (QCBM) using this variational ansatz to generate samples from joint distributions of two variables in historical data from the stock market. We demonstrate our generative learning algorithms on trapped ion quantum computers from IonQ for up to eight qubits. Our experimental results show interesting findings such as the resilience against noise, outperformance against equivalent classical models and 20–1000 times less iterations required to converge as compared to equivalent classical models.

DOI: 10.1103/PhysRevResearch.4.043092

I. INTRODUCTION

Understanding the statistical relationship between several random variables is critical to all data-based analysis and decision-making. A few examples of its diverse applications include risk management [1], portfolio optimization [2], reliability analysis [3], recommender systems [4], climate research [5], and medical imaging [6]. Traditionally, single-parameter quantities such as the Pearson correlation or Spearman's correlation have been used to model dependence between variables. However, such measures are good for monotonic dependence, which is frequently too simplistic for real data. Data, such as that from the financial markets, engineering reliability studies, earth/atmospheric sciences tends to exhibit tail dependence. This means they do not appear to have much correlation but exhibit dependence in extreme deviations, as in the case of a black swan event [7].

Due to the reasons above, the relationship between random variables is now commonly modeled using a dependence function between uniformly distributed variables, called a "copula". Sklar's theorem [8], which will be explained in the text, states that any multivariate joint distribution can be written in terms of univariate marginal distributions and a copula that describes the dependence structure between the variables. This provides the theoretical foundations for the use of copulas. Since then, copulas have found many applications in quantitative finance [9], engineering [10], and medicine [11].

Some of the commonly used copulas include elliptical and Archimedean. However, empirical copulas (copulas from real data) tend to be a mixture of copulas and are commonly modeled using parametric methods like maximum likelihood estimation [12]. As a rule, the more complex the copula and the more completely it describes the data, the more computationally challenging it becomes to extend it to higher dimensional data.

More recently, generative models have been proposed for statistical modeling. These learn to generate data with the same statistics as a given training dataset, effectively learning its distribution. The model can be used to output new samples that could plausibly have belonged to the original dataset [13,14]. One of the most successful of these is the generative adversarial network (GAN). In GAN, two neural networks compete with one another in a minimax game. In Ref. [13], the authors showed theoretically that a GAN will learn the data distribution if given enough capacity, data, and training time. Since then, GANs have found applications in areas like data augmentation [15], fashion design [16], and super resolution [17], etc. However, there is still debate over whether GANs will successfully learn a given distribution [18]. In particular, problems like vanishing or unstable gradient, mode collapse and nonconvergence, sensitivity to hyperparameters, and either the generator or discriminator overpowering the other, can make them challenging to train [19].

[*]These authors contributed equally to this work.

In this context, the question arises whether quantum computers can provide any advantage over classical models for generating samples that reproduce the interdependence between multiple variables of a given dataset. Quantum wavefunctions naturally represent probability distributions, which can be hard to sample from by classical algorithms. Quantum computers can also be used to generate correlations, which cannot be efficiently reproduced by classical means [20,21]. Therefore, intuitively, learning dependence between multiple random variables is something at which a quantum computer may excel. Further, an application that requires sampling from a probability distribution is likely one of the near term applications of quantum computers. As evidence, note that this is already the basis of the quantum supremacy demonstration [22].

Quantum generative models such as the quantum circuit Born machine (QCBM) [23,24] and quantum generative adversarial network (QGAN) [25,26] have been proposed as quantum algorithms for the learning of both classical and quantum data. Experimental demonstrations include training a QCBM to produce sample from the bars and stripes dataset [27] and to approximately prepare many-body states [24]. QGANs have been used in state preparation [28], learning of qubit channels [29], and image generation [30]. In Ref. [31], the output from a quantum circuit is used as the prior to a classical GAN. Since quantum algorithms are known to achieve speedup over classical algorithms in a variety of tasks [32,33], it has been hypothesized that quantum generative models can do better than classical generative models in terms of expressivity, stability during training, and shorter training time [26].

In the existing literature, generative quantum learning has either been restricted to learning quantum data, univariate distributions, or multivariate distributions without separating the correlation structure from the marginal distributions. Here, we extend this prior work using a natural connection between entanglement and the copula function. We proposed an architecture based on sampling of variational circuits. This allowed us to argue quantum advantage based on shallow circuits. This also extends previous quantum advantage argument in QGAN based on fault-tolerant quantum algorithms like the factoring algorithm [25,33] or the quantum algorithm for systems of linear equations [26,34]. As the model uses shallow circuits to learn classical data, it is NISQ-friendly and of wide applicability. Another significant difference to previous QGAN architectures is that we consider measurement samples as outputs, which saves a lot of computational resources when compared to other schemes that uses expectation values as outputs. Our tests on quantum simulators and hardware show advantages for the stability of the training process as well, which is relevant for practical applications.

This paper is especially relevant as quantum hardware has also been advancing rapidly implying that useful applications of quantum algorithmic advances may not be far off. A variety of noisy intermediate scale quantum (NISQ) computers are now available over the cloud, of which we use the trapped ion quantum processing units (QPU) from IonQ.

The outline of our paper is as follows. We begin by introducing the basic concepts of GAN, QGAN and QCBM necessary for this paper in Sec. II. In Sec. III, we introduce copulas and show how they can be learned using maximally entangled quantum states. Section IV gives a brief overview of the quantum hardware used for our experiments. In Secs. V and VI, we devise training routines for our quantum generative models that are implementable on a NISQ device. We then demonstrate via numerical simulation and executions on the IonQ QPUs that our generative learning algorithms can learn from empirical data. Section VII compares the performance of our algorithms with a classical GAN of similar number of parameters as well as classical parametric methods, revealing advantages for the quantum methods. We argue via communication and computational complexity that our quantum learning algorithms have a computational advantage over classical algorithms for learning joint distributions in Sec. VIII. Section IX provides an overview of future work. We present a discussion and conclusion in Sec. X.

II. CLASSICAL AND QUANTUM GENERATIVE ADVERSARIAL NETWORK, QUANTUM CIRCUIT BORN MACHINE

Generative modeling is an unsupervised learning task that involves the learning of patterns in the training data such that the model can generate new examples as if they are drawn from the original data. A generative adversarial network is a framework to train a generative model by transforming it to a supervised learning task, in which two neural networks are trained alternatively in a minimax game.

GAN employs two submodels, a generator G and a discriminator D. $G_{\theta_g}: Z \to X$ represents a mapping from a latent space Z to the data space X, with θ_g denoting the parameters of G. $D_{\theta_d}: X \to [0, 1]$ represents a mapping from the data space X to a scalar, with θ_d denoting the parameters of D. For each $x \in X$, $D(x)$ can be interpreted as the probability that x belongs to the original training data. D is trained to maximize the probability of differentiating true data from samples generated by G. Simultaneously, G is trained to "fool" the discriminator by generating realistic data samples. In other words, the two models D and G are trained in a two-player minimax game with value function

$$\begin{aligned}&\min_G \max_D V(D, G)\\&\quad = \mathbb{E}[\log D(x)] + \mathbb{E}[\log(1 - (D(G(z))))]. \qquad (1)\end{aligned}$$

After training, the generator is retained to generate new data samples for downstream use.

In practice, the minimax optimization is realized with two loss functions, one for the generator and one for the discriminator. For a batch of m samples from the real data $\{x^{(l)}\}$ and m noise vectors $\{z^{(l)}\}$, the loss function of the generator is

$$L_G(\theta_g, \theta_d) = -\frac{1}{m}\sum_{l=1}^{l=m}[\log D(G(z^{(l)}))] \qquad (2)$$

and the loss function of the discriminator is

$$L_D(\theta_g, \theta_d) = -\frac{1}{2m}\sum_{l=1}^{l=m}[\log D(x^{(l)}) + \log(1 - D(G(z^{(l)})))]. \qquad (3)$$

A typical GAN training schedule is show in Algorithm I.

Algorithm 1 GAN Training Loop

```
Result: Trained Generator $G$
Initialize Network
for $i \leftarrow 1$ to iterations do
    Sample minibatch of $m$ data examples $\{x^{(l)}\}$;
    Sample minibatch of $m$ data examples $\{x^{(l)}\}$;
    Calculate Discriminator Loss $L_D$;
    Update the discriminator by descending its stochastic gradient
      $\nabla_{\theta_d} L_D(\theta_g, \theta_d)$;
    Calculate Generator Loss $L_G$;
    Update the generator by descending its stochastic gradient
      $\nabla_{\theta_g} L_G(\theta_g, \theta_d)$;
end
```

The stochastic gradient descent is usually performed using adaptive algorithms like Adam [35] with a suitable learning rate. Some researchers also update the discriminator/generator multiple times in a training iteration or use different learning rates, depending on the problem.

Intuitively, the generator is trying to get better at generating data samples and thus fool the discriminator. The discriminator is trying not to be fooled. The discriminator provides positive feedback to the generator, such that the generator continues to improve. However, since it is a minimax optimization, it is very common for GAN training to suffer from nonconvergence (GAN fails to learn well from the training data) and mode collapse (GAN generates data with limited variety). While many techniques have been proposed to improve the stability of GAN training, these are still common problems researchers face.

Recently, quantum generative adversarial network (QGAN) [25,26] has been proposed as a generalization of classical GAN that can run on a quantum computer. QGAN can be used to learn either classical or quantum data. For the learning of quantum data, each training sample x becomes a quantum state $|x\rangle \in \mathcal{H}_X$. $G_{\theta_g} : \mathcal{H}_Z \to \mathcal{H}_X$ represents a quantum circuit from the latent Hilbert space $\mathcal{H}_Z$ to the data space $\mathcal{H}_X$, and $D_{\theta_d} : \mathcal{H}_X \to [0,1]$ can be either a quantum circuit or a classical neural network. Note that if D is a classical neural network, initial measurement is needed to convert $|x\rangle$ to some classical data.

For the learning of classical data, one way of realizing the quantum generator $G_{\theta_g} : Z \to X$ is to encode a noise vector z as some quantum state $|z\rangle$, apply a quantum circuit U_G, and then measure the expectation of some observable O_G, i.e., $G(z) = \langle z|U_G^\dagger O_G U_G|z\rangle$. Another way, as noted in Ref. [28], is to represent G_{θ_g} as a quantum circuit U_G followed by some POVM (positive operator-valued measure) M_x in X space. In this case, the dependence of G on a latent space Z is optional as POVM measurement is inherently probabilistic and data samples are drawn from the probability distribution

$$q(x) = \mathrm{tr}(M_x U_G|0\rangle\langle 0|U_G^\dagger). \tag{4}$$

A quantum circuit Born machine (QCBM) is a quantum generative model similar to the generator part of QGAN, except the training objective is different. In the QCBM case, if we suppose that the target distribution is p, a QCBM seeks to minimize the distance between distributions p and q directly, through either Kullback-Leibler (KL) divergence [24] or maximum mean discrepancy [36]. Here, we use the KL divergence as a cost function, defined as

$$d_{\mathrm{KL}} = \sum_x p(x) \log \frac{p(x)}{q(x)}. \tag{5}$$

III. COPULA AND QUANTUM ENTANGLEMENT

Given random variables $(\mathcal{X}_1, \ldots, \mathcal{X}_d)$, probability integral transform states that the marginal cumulative distribution functions defined as $F_i(x) = \Pr[\mathcal{X}_i \leqslant x]$ satisfies $\Pr[F_i(\mathcal{X}_i) \leqslant u] = u$. Therefore the random variables

$$(\mathcal{U}_1, \ldots, \mathcal{U}_d) = (F_1(\mathcal{X}_1), \ldots, F_d(\mathcal{X}_d)) \tag{6}$$

have marginals, which are uniformly distributed on [0,1].

A copula C of $(\mathcal{X}_1, \ldots, \mathcal{X}_d)$ is defined as the joint cumulative distribution function of $(\mathcal{U}_1, \ldots, \mathcal{U}_d)$,

$$C(u_1, \ldots, u_d) = \Pr[\mathcal{U}_1 \leqslant u_1, \ldots, \mathcal{U}_d \leqslant u_d]. \tag{7}$$

Reversing the above steps gives a way to generate samples from multivariate distributions. Given a procedure to generate samples $(\mathcal{U}_1, \ldots, \mathcal{U}_d)$ from the copula function C,

$$(\mathcal{X}_1, \ldots, \mathcal{X}_d) = \left(F_1^{-1}(\mathcal{U}_1), \ldots, F_d^{-1}(\mathcal{U}_d)\right) \tag{8}$$

would be samples from the original multivariate distribution.

Sklar's theorem [8,37] states that every multivariate cumulative distribution function on random variables $(\mathcal{X}_1, \ldots, \mathcal{X}_d)$ can be expressed in terms of their marginal cumulative distribution functions F (defined as $F_i(x) = \Pr[\mathcal{X}_i \leqslant x]$) and a copula C [defined in Eq. (7)], i.e.,

$$\Pr[\mathcal{X}_1 \leqslant x_1, \ldots, \mathcal{X}_d \leqslant x_d] = C(F_1(x_1), \ldots, F_d(x_d)). \tag{9}$$

Copulas are used to simulate correlated variables because they remove the structure of the marginal distributions, and capture just the pointwise correlation between variables. Simple copula formulas are extensively used in finance, engineering and medicine. However these simple formulas often fall short in accurately capturing the relationship between variables. Instead more sophisticated or empirical copulae can be used but these become hard to model and sample from as the number of variables increases. Very recently, generative models have been successfully used to model copulas [38].

For our study, we test our generative models using the following framework:

(1) Use Eq. (6) to transform training data into copula space

(2) Fit known copula model/GAN/QGAN/QCBM in copula space

(3) Sample from known copula model/GAN/QGAN/QCBM in copula space

(4) Use Eq. (8) to transform synthetic data from copula space back to original space

Next, we show that every copula with density can be represented by a maximally entangled state.

Suppose $C : [0,1]^d \to [0,1]$ is a d-dimensional copula with density $c : [0,1]^d \to [0,1]$. c can be obtained as

$$c(u_1, \ldots, u_d) = \frac{\partial^d C(u_1, \cdots, u_d)}{\partial u_1 \cdots \partial u_d}. \tag{10}$$

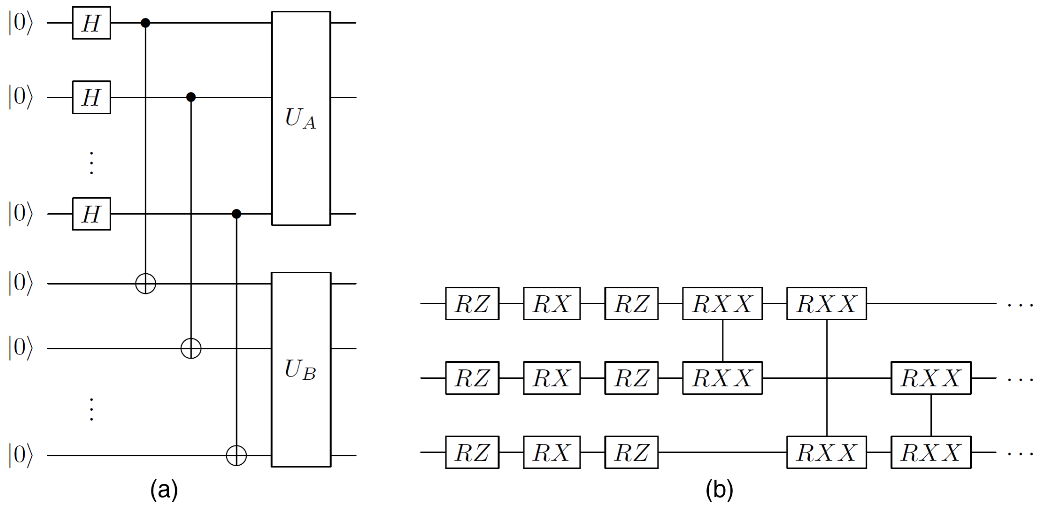


FIG. 1. (a) The qopula circuit for 2 random variables. The top (bottom) half of the circuit consists of qubits that belong to the register, which provides samples of the first (second) random variable. (b) The ansatz for U corresponding to 3 qubits. All gates are parametrized by angles, which are optimized during the learning process. The structure can be repeated for multiple layers each with different parameters. Here the 2-qubit RXX gates represent $\exp(i\theta X_i X_j)$ acting on qubits i and j, which can be executed as $\mathrm{CNOT}(i, j)\exp(i\theta X_i)\mathrm{CNOT}(i, j)$ using standard gates available on gate-model quantum computers. The RZ and RX gates represent rotations around the Z and X axes.

This can be represented as a quantum state

$$|c\rangle = \sum_{k_1,\dots,k_d=0}^{1} \int_{[0,1]^d} e^{i\phi(u_1,\cdots,u_d,k_1,\cdots,k_d)} \frac{\sqrt{c(u_1,\cdots,u_d)}}{2^{d/2}} \times |u_1,k_1\rangle\cdots|u_d,k_d\rangle d^d u. \tag{11}$$

Since $\mathcal{U}_1,\dots,\mathcal{U}_d$ have uniform marginals,

$$\int_0^1 \cdots \int_0^1 c(u_1,\cdots,u_d)du_1\cdots du_{i-1}du_{i+1}\cdots du_d = 1 \quad \forall i, \tag{12}$$

and by setting the phases ϕ appropriately, the reduced density matrix $\rho_i = \mathrm{Tr}_{1,\cdots,i-1,i+1,\cdots,d}(|c\rangle\langle c|)$ for partition i is completely mixed and $|c\rangle$ is a maximally entangled state with respect to each of the d partitions. (See Appendix A for the detailed proof.)

Here we construct a quantum circuit that can prepare such maximally entangled states, which we christen as a qopula circuit. The quantum circuit for two random variables is shown in Fig. 1(a). The first part of the circuit consists of Hadamard gates applied to all qubits in register A, followed by CNOTs between A and B. This creates $N_q/2$ Bell pairs, which are distributed between A and B. This results in the creation of a state of the form

$$|\psi_{AB}\rangle = \frac{1}{2^{N_q/4}} \sum_{i=0}^{2^{N_q/2}-1} |i_A\rangle|i_B\rangle. \tag{13}$$

In this state, the reduced density matrix of the states in each register is completely mixed ($\rho_A = I/|A|$). In the next step, unitaries act locally on registers A and B. Since $U^\dagger I U = I$, the action of the unitaries does not change the reduced density matrix, but it does change the correlations between the registers A and B. Note that the unitaries on each register can be set by parameters that are independent of the other.

If each register is measured in the computational basis with output $x_1,\dots,x_n,y_1,\dots,y_n$, the measured bitstrings can represent discretized values of each random variable in the domain [0,1] through the following linear transformation

$$x = \frac{1}{2^{1+n}} + \sum_{i=0}^{n} \frac{1}{2^n} x_n. \tag{14}$$

Therefore this ansatz can model the correlations between uniform random variables and capture the copula dependence structure.

The number of qubits in each register determines the discretization of the probability distribution. The more the qubits, the more finely the pointwise correlation can be learnt. If the number of qubits is restricted, for practical application, one can use the available qubits to learn the significant bits and then append random bits in the position of the less significant bits.

The circuit can be extended to more than two variables by preparing GHZ states instead of Bell pairs as the starting point on which the operators U act. The exact structure of U is not restricted and a subject of future research. Here we use a structure that consists of layers alternating between a "driver" layer that consists of parametrized single qubit rotations RZ and RX and an "entangler" layer that consists of RXX gates as shown in Fig. 1(b) [39]. This structure can be repeated many times to approximate general unitaries. We also note that the uniform marginal condition can be satisfied by many states other than the simple Bell or GHZ states [40].

The dataset that we use is the daily return of AAPL and MSFT between 2010–2018. Figure 2(a) represents the hypothetical growth of capital if \$10 000 is invested in both stocks (assuming dividends are reinvested). The daily returns are calculated as the percentage change of close prices between adjacent trading days. Suppose on Day 1 a stock's close price is P_1 and its close price is P_2 on Day 2, then its daily return on Day 2 is $r = (P_2 - P_1)/P_1$. The daily return data is plotted both in $\mathcal{X}$ space [real data space, Fig. 2(b)] and $\mathcal{U}$ space [copula space, Fig. 2(c)]. In model training, no time series

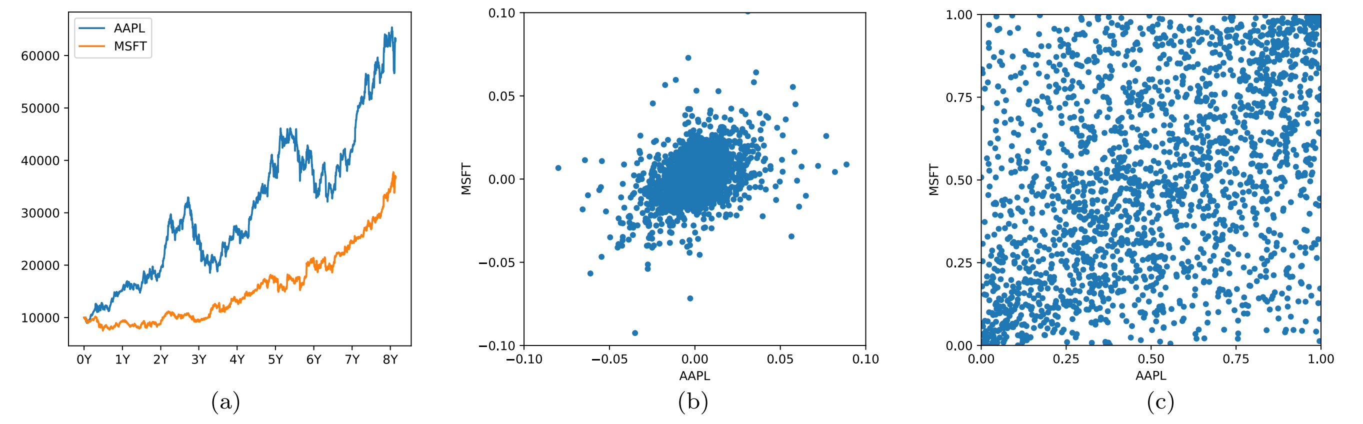


FIG. 2. (a) Hypothetical growth of \$10 000 invested in AAPL and MSFT between 2010–2018. (b) Scatter plot of daily returns. (c) Scatter plot of data after probability integral transform.

model is assumed and hence the daily returns are assumed to be independent events.

While the dataset being used for this research comes from the financial market, we would like to emphasize that our quantum generative algorithms are completely general and should be able to learn from any dataset that has correlations among different fields.

IV. QUANTUM HARDWARE

The experiments were run on IonQ trapped ion quantum processing units (QPUs), which utilize stable ground states of trapped ytterbium atomic ions as qubits. Two ground states of these atoms with hyperfine splitting are used as the two qubit states, and they are initialized, manipulated and detected with adequately tailored laser lights applied to them. The quantum circuit is executed by applying quantum logic gates driven by Raman transitions using up to 32 individually addressed beams of 355 nm light. These Raman transitions are programmed to provide a universal gate set by applying individual rotations of a given qubit's internal states to realize a single-qubit gate, as well as two-qubit gates by inducing Molmer-Sorenson type interaction to generate entanglement between a pair of qubits in the system. IonQ's QPUs feature all-to-all connectivity, where a two-qubit gate can be applied directly to any pair of qubits in the system regardless of their physical location [41]. IonQ's QPUs available on the cloud are self-calibrating, which means that the system monitors the quality of the quantum logic gates constantly and makes sure they are fully calibrated before the computational tasks are executed. The hardware used in this work utilizes IonQ QPUs that are available over the cloud, as well as a next-generation QPU.

V. QUANTUM LEARNING THROUGH QGAN

In our QGAN implementation, the quantum generator comprises of 6 qubits, and each unitary uses 1 layer of ansatz as described in Fig. 1. Therefore the quantum generator has 24 trainable parameters. A measurement in the computational basis is performed at the end of the circuit and Eq. (14) is applied to get the 2-dimensional sample in copula space. To deal with the issue of discretization, the measurement output of $(x_0, x_1, x_2, y_1, y_2, y_3)$ is extended to $(x_0, x_1, \cdots, x_{22},\ y_0, y_1, \cdots, y_{22})$ with the additional 40 bits uniformly randomly generated, and then the linear transformation is applied.

The discriminator is a feed-forward classical neural network with input dimension 2, a hidden layer of dimension 32 and an output layer of dimension 1. The input and hidden layer comprises of a linear layer and leaky rectified linear unit (ReLU), whereas the output layer is a linear layer followed by sigmoid function.

To compare the expressivity of QGAN vs GAN, we design the generator of classical GAN to have the same number of trainable parameters. This can be achieved by a 2-layer feed-forward neural network with input dimension 6 and output dimension 2. The input layer consists of a linear layer followed by batch norm and ReLU. The output layer consists of a linear layer followed by a sigmoid function. A sigmoid function is needed to transform samples back to the unit square.

Neural-network models are typically trained using gradient-based optimizers. In the case where the cost function has a quantum circuit component, the gradient with respect to circuit parameters can be calculated using the parameter-shift rule [42]. However, this procedure requires two executions of the original circuit (each with shifted parameters), and thus the number of circuit executions scales two times as the number of circuit parameters. For example, in our case, it would take 48 circuit executions to compute the gradient with respect to the 24 parameters in the quantum generator. This is executed sequentially in most software for quantum computers. In addition to slower evaluation of the cost function itself, since commercially available quantum computers (including IonQ's) are typically cloud based, sequential evaluation will also be slow due to network latency. For this reason, we avoided the computation of quantum gradients and used the simultaneous perturbation stochastic approximation (SPSA) algorithm [43]. SPSA algorithm updates the parameter in an iterative process, with the kth step

$$\theta_{k+1} = \theta_k - a_k \hat{g}_k(\theta_k), \tag{15}$$

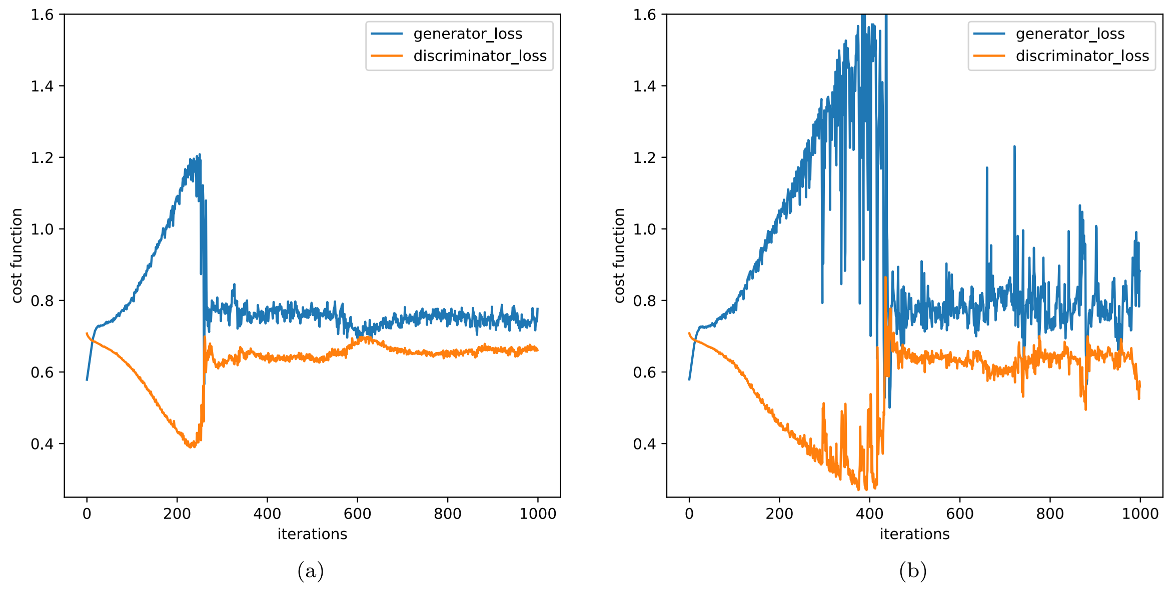


FIG. 3. Plot of the loss functions from QGAN training for $N_q = 6$ qubits. (a) Loss plot from simulator. (b) Loss plot from experiment.

where the ith component of the gradient estimator $\hat{g}_k(\theta_k)$ is

$$(\hat{g}_k(\theta_k))_i = \frac{C(\theta_k + c_k\Delta_k) - C(\theta_k - c_k\Delta_k)}{2c_k(\Delta_k)_i}. \qquad (16)$$

Here Δ_k is a random perturbation vector with each element drawn uniformly at random from $\{-1, 1\}$. For initial learning rate a and step size c, a_n, and c_n are updated as $a_k = a/k$ and $c_k = c/k^\gamma$, where γ is chosen from [1/6, 1/2]. Through numerical simulation, we observe that running SPSA algorithm with 3–5 iterations in place of a gradient descent step for the generator, yields good performance in our QGAN training schedule. Since each iteration step only requires 2 circuit executions, an optimization step using SPSA would only require 6–10 circuit executions in total.

Therefore, our QGAN training schedule will be as shown in Algorithm 2.

Algorithm 2 QGAN Training Loop

```
Result: Trained Generator $G$
Initialize Network and angles of the Quantum Circuit ansatz;
for $i \leftarrow 1$ to $iterations$ do
    Run the quantum generator with $m$ shots to generate $m$ measurements;
    Sample minibatch of $m$ data examples $\{x^{(l)}\}$;
    Calculate Discriminator Loss $L_D$;
    Update the discriminator by descending its stochastic gradient $\nabla_{\theta_d} L_D(\theta_g, \theta_d)$;
    Calculate Generator Loss $L_G$;
    Update the Quantum Generator by using SPSA algorithm;
end
```

We first test our model in simulation before running it on IonQ's cloud QPUs. All simulations and experiments of QGAN are trained with 1000 iterations and use the parameter $a = 0.008$, $c = 0.01$, number of iterations $n_{\text{iter}} = 5$, $\gamma = 0.101$ for SPSA, learning rate 0.0015 for the discriminator, batch size $m = 2048$. To reduce the experiment time, we used the maximum learning rates possible without breaking down the learning. To better understand the effect of hardware noise, random initialization with the same seed is used for comparing simulation and experiment results. However, other random initializations were also used in simulation and results are summarized in Sec. VII.

Results from the simulator, Fig. 3(a) indicates that in our training process, the losses from the generator and discriminator first diverge before converging around iteration 250. This implies that, the discriminator learns faster than the generator in the beginning, but then the generator experiences an Aha! moment and quickly catches up. This is also evident when we examine the quality of synthetic data produced (Fig. 4), where the KS statistics rapidly decreases around the convergence point, and then enters a stable phase. (KS statistics is explained more in Sec. VII where we evaluate the performance of the models.)

As quantum computers at the moment still have significant noise, it is essential to investigate how noise would affect our training algorithm. We ran the same training algorithm on a mixed state simulator with the addition of up to 4% depolarizing noise following each two-qubit gate, which is an appropriate model for noise on IonQ's cloud QPUs [44]. We observe that the losses of the generator and discriminator diverge a bit further. This is to be expected that noise makes the generator a weaker learner and therefore it needs more time to learn. However, in most of the cases, our QGAN converges eventually (anywhere between iterations 300–500), indicating robustness against noise and suitability to run on NISQ machines. We observe that in our experiment running on IonQ's cloud QPUs, the losses converged at around iteration 400, agreeing with results from noisy simulations.

QGAN-generated data is visualized in both $\mathcal{X}$ space [real data space, Fig. 5(a)] and $\mathcal{U}$ space [copula space, Fig. 5(b)]. We observe that the data in copula space [Fig. 5(b)] are concentrated around a few clusters and are less evenly distributed than the original data [Fig. 2(c)]. This is because we used only 3 qubits to represent each dimension and therefore the

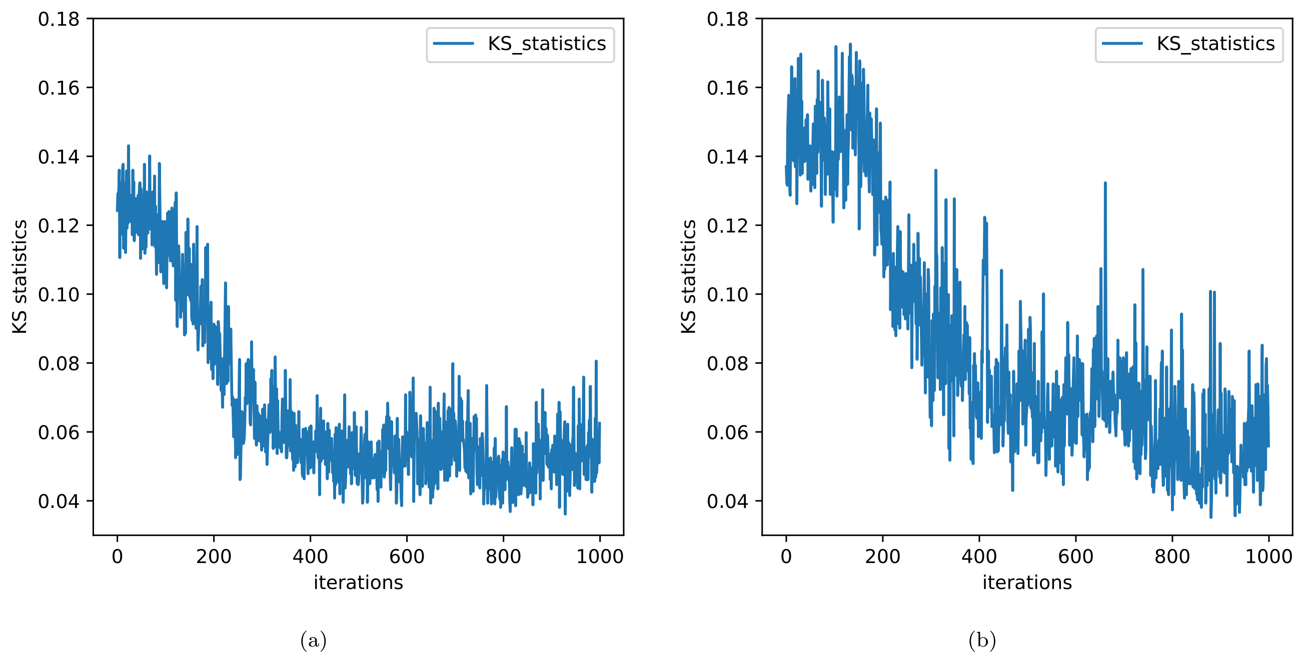


FIG. 4. Plot of the KS statistics from QGAN training for $N_q = 6$ qubits. (a) KS statistics plot from simulator. (b) KS statistics plot from experiment.

output is very discrete. One would only see 64 unique data points had we not added the random bits after measurements. The random bits have the effect of creating small fluctuations around the original discrete value, hence the clusters.

VI. QUANTUM LEARNING THROUGH QCBM

For the QCBM, we use the KL divergence as a cost function. As the target, we bin the data in copula space into $2^{N_q/2}$ bins, where N_q is the number of available qubits, generating a discrete probability distribution p. This is compared to the probability distribution q over the computational basis states measured at the output of the circuit. The KL divergence cost function is

$$d_{\mathrm{KL}} = \sum_{i,j} q_{i,j} \log \frac{q_{i,j}}{p_{i,j}}, \tag{17}$$

where i and j are for the first and second variable respectively. Numerically, we implement the clipped version of the KL divergence where $p_{i,j}$ are set to 10^{-6} if below that value. The

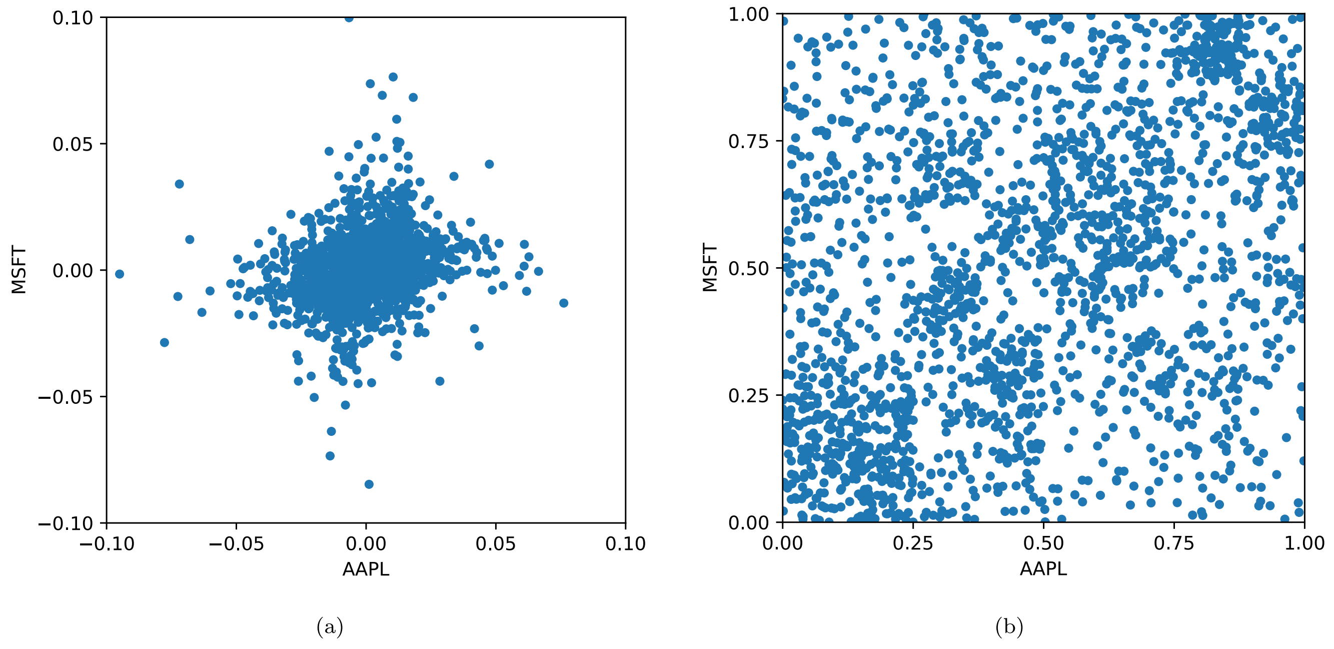


FIG. 5. (a) Scatter plot of QGAN-generated daily returns after reverse probability integral transform. (b) Scatter plot of QGAN-generated copula data. $N_q = 6$ qubits were used.

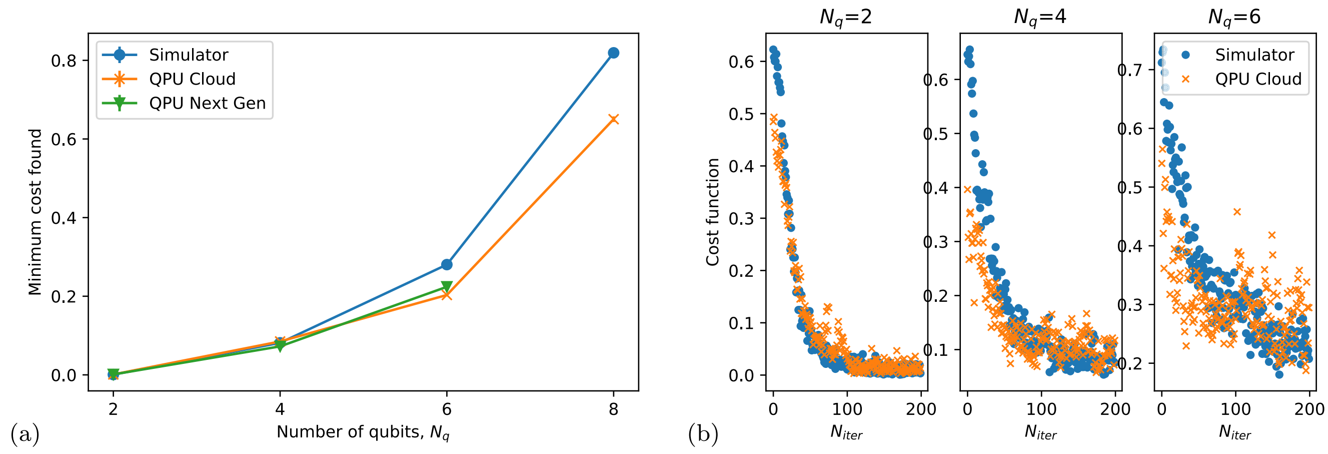


FIG. 6. QCBM training (a) average of the 10 minimum values of the cost function reached during training as a function of the number of qubits for optimization parameters $a = 0.5$, $c = 0.5$. The error bars are smaller than the marker sizes. (b) Convergence of the cost function as a function of the number of iterations for $a = 0.1$, $c = 0.1$. The panels indicate results from different numbers of qubits N_q.

QCBM training schedule is shown in Algorithm 3.

Algorithm 3 QCBM Training Loop

```
Result: Trained Quantum Circuit
Initialize angles of the Quantum Circuit ansatz;
for $i \leftarrow 1$ to $iterations$ do
    Run the quantum circuit with $m$ shots to generate $m$ measurements;
    Calculate the KL divergence $d_{KL}$;
    Update the circuit angles by using SPSA algorithm;
end
```

For converting the measured bits into samples, Eq. (14) is used along with the procedure of appending the random bits as described in the QGAN section.

The results of training are shown in Fig. 6. The results are shown for a series of system sizes consisting of 2, 4, and 6 qubits. A one-layer ansatz was used in each case. Each system size was initialized with the optimal angles from the previous smaller system size, which can be thought of as a form of transfer learning. The training was done on the simulator, IonQ's cloud QPUs, and its latest generation QPU, which is not yet accessible through the cloud.

The optimization algorithm used for training is the SPSA similar to the QGAN training. Two hundred iterations were used for all experiments. Figure 6(a) shows results for up to 8 qubits on the simulator and cloud quantum computer and up to 6 qubits on the next-generation quantum computer. The optimization parameters used were $a = 0.5$ and $c = 0.5$, for which the training finds the minimum within 20 iterations. As can be seen in this figure, which shows the average over the 20 smallest values of the measured cost function, for 2 and 4 qubits, all three backends converge to a similar value. For 6 qubits, somewhat surprisingly, we observe that the noisy hardware has a smaller value of the final cost function. This observation of the noise being favorable for converging to a low value of the cost function at the beginning of the optimization is further confirmed in Fig. 6(b), which shows the convergence for $a = 0.1$ and $c = 0.1$, where it is slow and hence can be seen clearly. We note that we do not expect that the noise will continue to remain beneficial at larger system sizes, however.

While vanishing gradient (barren plateau) has been observed in several variational quantum algorithms [45], we note that we see no evidence of it even as the system size increases. Instead, the training landscape becomes noisier even for the simulator as shown by the broadening of the cost function values as the number of qubits increases. This is to be expected since only a finite number of shots are used whereas the number of possible outputs increases exponentially with the number of qubits. We note that this would also be true for a classical generator. More discussion of the QCBM training can be found in Appendix B.

We further note that for the cases we tested, the QCBM technique requires fewer function calls to train the model than the QGAN approach. Which technique may end up performing better will depend on the details of the application. Finally, we also note that an improved convergence is not observed for the increased quantum gate fidelity of the next-generation QPU for either approach, implying that the noise plays some positive role in the statistical machine learning models. The major advantage of the next-generation QPU in this application is its increased stability that leads to reduced runtime of the entire optimization task by about a factor of 2.

VII. EVALUATION OF MODEL PERFORMANCE

We use the 2-dimensional 2-sample Kolmogorov-Smirnov (KS) test [46] to evaluate our generative models together with classical GAN and a parametric model. The null hypothesis of the 2-sample KS test is that the two samples are drawn from the same distribution, and the alternative hypothesis is that they are drawn from different distributions. It is typically performed on 1-dimensional distributions but here we are performing on 2-dimensional distributions. We use a significance level of 0.05 here. If the p-value produced by the test is more than the significance level specified, then we accept the null hypothesis. In the case where multiple models produce samples, which are accepted by the KS test, we use the associated KS statistics to compare among the models. The KS statistic

TABLE I. KS statistics and p-value of KS test across multiple models. The quantum models use $N_q = 6$ qubits.

Model	D_{KS} (the smaller the better)	p-value (threshold 0.05)
Parametric model	0.0449	0.117
Classical GAN	0.0363–0.0508	0.0530–0.309
QGAN simulation	0.0320–0.0396	0.226–0.473
QGAN experiment, QPU cloud	0.0352	0.3570
QCBM simulation	0.0425–0.0520	0.0511–0.1717
QCBM experiment, QPU cloud	0.0373–0.0515	0.0548–0.3030
QCBM experiment, QPU Next Gen	0.0330–0.0510	0.0578–0.4465

quantifies the distance between the empirical distributions of two samples. For 1-dimensional data, it can be defined as

$$D_{KS}(P, Q) = \sup_x |P(X \leqslant x) - Q(X \leqslant x)|, \tag{18}$$

where P and Q are the empirical cumulative distributions. Please refer to Ref. [46] for the 2-dimensional generalization.

For the classical models and QGAN, 2048 samples are collected from our generative models, and for the QCBM, 2000 samples are collected from 4 circuits with minimal cost function that were run with 500 shots each. These are compared with a bootstrapped version of the training data. The architecture of classical GAN is described earlier. The parametric model we used involves fitting the copula data with a Gaussian copula.

For classical GAN and QGAN on simulator, we were able to run the model multiple times with different initializations and thus there is a range of results. For QGAN, we initialized the parameters uniformly at random from $[0, 2\pi]$ and note that our model converges regardless of initialization. For classical GAN, we used the default initialization method from PyTorch [47] with different seeds. We note that about 40% of our model instances are accepted by the 2D KS test with threshold 0.05. The rest failed to learn and were rejected by the test. In Table I, the metrics of those accepted instances were included. For the QCBM, in order to see the range of performance, we first generated 5 datasets, each of which consisted of data from running the QCBM with parameters from 4 unique points where the cost function values were minimized during the optimization. For each dataset, we calculated the performance by affixing less significant random bits to the ones from the generator. We then also averaged over 20 such sets of random bits. The range shown in Table I is then the minimum and maximum of the D_{KS} and p-values of these 5 datasets compared to the target distribution.

As one can see, the parametric model does a good job in modeling the copula but generative models are able to do better. The result from QGAN is consistent and outperforms classical GAN with similar number of parameters. The QCBM performs slightly worse than the QGAN but comparable to the classical GAN. Figure 7 shows the results for the D_{KS} as a function of number of qubits for the QCBM training. Up to 6 qubits, the quantum computers perform better for QCBM training than the simulator as is to be expected from their better performance in copula space as well. We note that D_{KS} decreases up to 6 qubits as expected but increases after that indicating the need for more shots and iterations for the optimization to converge.

We note that it is entirely possible for classical GAN to achieve better results with a deeper neural network as generator. However, the same can be said about our quantum generative models as they are only learning the significant bits at the moment. With better hardware, more qubits, more layers and better design for the ansatz U, our quantum generative models will also do better. Nevertheless, our results provide compelling evidence that our quantum generator has more expressivity than classical generative models.

We also note that we are able to train QGAN/QCBM at a much faster learning rate and therefore conclude the training with much fewer iterations than classical GAN. In classical GAN, the learning rate used is 0.0001 and model training concludes after 20 000 iterations. Attempts to increase the learning rate failed due to nonconvergence in model training. In QGAN, model training concludes after 1000 iterations. In QCBM for 6 qubits, the training converges to a good value for as little as 20 iterations.

Currently the gate speeds of commercially available QPUs are typically slower than those of classical computers. Additionally, due to the current access mode of IonQ QPUs over the cloud that includes queued access priority, network latency and calibration time on the machine, our QGAN experiment took 2 weeks to complete. At this time, this is much slower as compared to other models. The training time for our classical GAN is 6 minutes on a CPU-only machine and that of QGAN simulation is 4 minutes. For the QCBM on the next-generation quantum hardware, the training for 6 qubits with 200 iterations took approximately 5 hours whereas it took 9 hours on

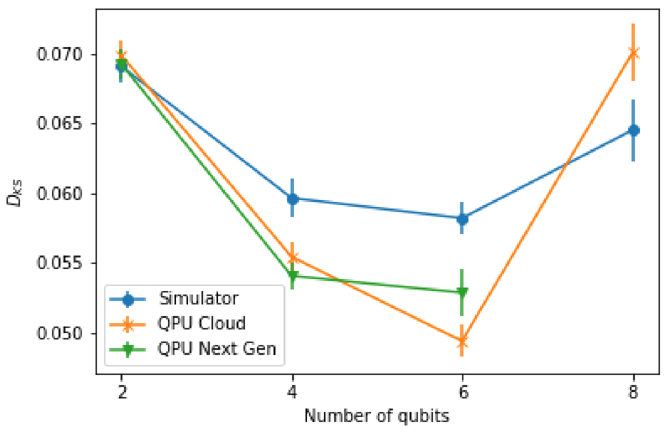


FIG. 7. KS metric averaged over the 10 best converged values of the cost function as a function of system size for the QCBM.

the cloud system. We expect many of the network latencies and calibration procedures to improve on QPUs in the future, as well as dedicated access to be available. Further, as shown in the next section, there will be certain joint probability distributions for which the run time for the classical models and the simulations are expected to grow exponentially with the number of qubits, while the QPU time will only grow polynomially, ultimately making the classical and quantum technologies competitive for learning applications.

VIII. QUANTUM COMPUTATIONAL ADVANTAGE

We next present arguments for the presence of computational advantage in our quantum machine learning algorithms over classical machine learning algorithms such as neural networks.

The first source of advantage stems from the quantum supremacy argument of "instantaneous quantum polynomial" (IQP) circuits [48], which is a family of nonuniversal quantum circuits. The output distribution of IQP circuits is of the form

$$q^{IQP}(z) = |\langle z|H^{\otimes n}U_D H^{\otimes n}|0^n\rangle|^2 \tag{19}$$

where $U_D = \exp(iD)$ and

$$D = \sum_{k,l} J_{k,l}Z_kZ_l + \sum_k M_kZ_k. \tag{20}$$

Here we quote some results from Ref. [48].

Lemma 1 (Corollary 1 of Ref. [48]). If the output probability distributions generated by uniform families if IQP circuits could be weakly classically simulated to within multiplicative error $1 \leqslant c \leqslant \sqrt{2}$ then the polynomial hierarchy would collapse to its third level, *i.e.*, $PH = \Delta_3$[49].

Hence, this type of distributions cannot be efficiently sampled classically, assuming $PH = \Delta_3$. The same argument underlies quantum supremacy arguments of boson sampling [20] and sampling outputs of QAOA [51].

Here we briefly sketch the original arguments. We encourage interested readers to refer to Ref. [48] for detailed proofs. A quantum circuit with postselection has output registers O and postselection registers P. Instead of sampling measurement results x directly from measurement on the output registers, we consider only those runs of the process for which a measurement on the postselection registers P yield $0\dots0$. In this construction, we require the circuit to have the property $\Pr[P = 0\dots0] \neq 0$ so that the conditional distribution $\Pr[O = x|P = 0\dots0]$ is well defined. If A and B are complexity classes, A^B denotes the class A with an oracle for B [50].

While IQP $\subsetneq$ BQP (BQP = bounded error quantum polynomial time), Ref. [48] argued that with postselection, they have the same power, i.e., post-IQP = post-BQP. If there is an efficient classical algorithm that can sample outputs from IQP, then IQP $\subseteq$ BPP (BPP = bounded error probabilistic polynomial time). The same relation would hold with postselection, i.e., post-IQP $\subseteq$ post-BPP. This would imply post-IQP = post-BQP $\subseteq$ post-BPP. Since we know a universal quantum computer can simulate a classical computer efficiently, i.e., BPP $\subseteq$ BQP, the same would hole with postselection, i.e., post-BPP $\subseteq$ post-BQP. This would immediately imply post-BQP = post-BPP. The equality of these two complexity classes will imply $PH = \Delta_3$, which we believe will not happen. Therefore, there is no efficient classical algorithm that can sample outputs from IQP, assuming the polynomial hierarchy does not collapse to the third level. Here we present a similar result for qopula circuits.

Theorem 1. If the output probability distributions generated by uniform families if qopula circuits could be weakly classically simulated to within multiplicative error $1 \leqslant c \leqslant \sqrt{2}$ then the polynomial hierarchy would collapse to its third level, *i.e.*, $PH = \Delta_3$.

Proof. We note that the output distribution of IQP circuits is a special case of the conditional output distribution from our circuit. The output distribution of our circuit can be written as

$$q(z_A, z_B) = \left|\langle z_A, z_B|U_A \otimes U_B\left(\frac{|0_A0_B\rangle + |1_A1_B\rangle}{\sqrt{2}}\right)^{\otimes N_q/2}\right|^2. \tag{21}$$

If we set all the parameters in U_B to 0 such that U_B is trivial, then

$$q(z_A|z_B = 0\cdots0) = |\langle z_A|U_A|0^{\otimes N_q/2}\rangle|^2. \tag{22}$$

If we set U_A to use 1 layer of the ansatz and set angles of all the RZ rotations to 0, then we are left with RX and RXX gates. Since they commute, we can write $U_A = \exp(i\tilde{D})$ where $\tilde{D} = \sum_{k,l} J_{k,l}X_kX_l + \sum_k M_kX_k$. We can see that Eq. (22) is of the same form as Eq. (19), as $HZH = X$. Since postselecting the qopula circuit with $B = 0\dots0$ allows us to generate distributions identical to that of IQP circuits, we see that IQP $\subseteq$ post-Qopula (the complexity class of qopula circuits with postselection) and therefore post-IQP $\subseteq$ post-Qopula $\subseteq$ post-BQP. Since post-IQP = post-BQP [48], we get post-Qopula = post-BQP. If the output distribution of qopula circuits can be sampled efficiently classically, we would have Qopula $\subseteq$ BPP and therefore post-Qopula $\subseteq$ post-BPP. This would imply post-BQP $\subseteq$ post-BPP, which would result in $PH = \Delta_3$. ■

Reference [52] argued quantum advantage of QCBM based on IQP circuits.

The second source of advantage stems from the exponential separation between shallow quantum circuits and shallow classical circuits [53]. Reference [53] defines a problem called the Parity Halving Problem PHP$_n$ as outputting a string $y \in \{0, 1\}^n$ given an input $x \in \{0, 1\}^n$ of even parity, such that $|y| \equiv |x|/2 (\mathrm{mod} 2)$. Here we present a result in Ref. [53] that is relevant for our discussion. We encourage interested readers to refer to Ref. [53] for detailed proofs.

Lemma 2 (Theorem 2 in Ref. [53]). The Parity Halving Problem (PHP_n) can be solved exactly by a $QNC^0/|CAT\rangle$ circuit. But on the uniform distributions over all valid inputs (even parity strings), any $AC^0/rpoly$ circuit of depth d and size at most $\exp(n^{1/10d})$ only solves the problem with probability $\frac{1}{2} + \exp(-n^{\alpha})$ for some $\alpha > 0$.

Here $|CAT\rangle$ denotes the GHZ-type state $\frac{1}{2}(|0^n\rangle + |1^n\rangle)$. AC^0/rpoly is the family of Boolean circuits of depth $O(1)$ and polynomial size, with unlimited-fanin gates, with the ability to sample from any probability distribution on polynomially many bits that is independent of the input, but that can depend on the input size. The quantum circuit that solves the problem is given as $H^{\otimes n}RZ(\vec{x}\pi/2)$ acting on $|CAT\rangle$, followed by measurement in the standard basis. Here $RZ(\vec{x}\pi/2)$ is understood as a layer of RZ gates where each gate is $RZ(x_i\pi/2)$. The measurement outcome is interpreted as the $y \in \{0, 1\}^n$ required by the problem.

Based on the above results, we derive the following.

Theorem 2. There exists n sets of measurements (each of size 2) determined by the optimizer (as defined by the corresponding machine learning model) such that a classical circuit of depth d with randomized advice of size poly(n) with size at most $\exp(n^{1/10d})$ can only produce samples from the output distribution produced by qopula circuits with probability $\frac{1}{2} + \exp(-n^{\alpha})$ for some $\alpha > 0$.

Proof. The quantum circuits proposed in the proof of Lemma 2 are special cases of qopula circuits, because

$$H = iRZ(\pi/4)RX(\pi/4)RZ(\pi/4) \tag{23}$$

and thus the circuit is equivalent to qopula circuit for n random variables, with 1 qubit for each random variable and 1 layer. The angle parameters are $\pi/4$ for the RZ's followed by $\pi/4$ for the RX's and $\pi/4 + \vec{x}\pi/2$ for the RZ's. There is no two-qubit gate after the initial entangling procedure.

The quantum generative models described in this paper involves parameterized quantum circuits and an optimizer producing the corresponding measurements (SPSA based on KL divergence for the case of QCBM and SPSA based on binary cross entropy and the discriminator for the case of QGAN). Replicating the proof of Lemma 2, it would mean that, in the case the parameters provided by the optimizer comes from the uniform distribution of even parity binary strings, the quantum circuits would produce a quantum state that is the uniform superposition of the halved-parity binary strings. If the corresponding quantum circuit is replaced by a classical circuit of depth d with randomized advice of size polynomial in the input size, such classical circuits with size at most $\exp(n^{1/10d})$ can only produce samples from the output distribution produced by qopula circuits with probability $\frac{1}{2} + \exp(-n^{\alpha})$ for some $\alpha > 0$. If the classical circuit is in the form of a classical neural network, it would mean the neural network with constant depth would need exponential dimension (either the input or the hidden dimension) to simulate the output distribution exactly.

The third source of advantage stems from Bell's theorem. The discussion below in summarized in Appendix C in the form of a theorem. Consider a Bell pair of two qubits. One qubit is given to an entity (say Alice) and the other to a different entity (say Bob). At this stage, they each also have access to random variables, which share arbitrary correlations. Next, suppose they are given measurements x and y, which are not shared, and based on these they produce outcomes a and b respectively. Then quantum mechanics implies that the joint conditional probability distribution, $P(a, b|x, y)$, cannot be reproduced by classical means without communication between Alice and Bob. Further, when n Bell pairs are shared between Alice and Bob, some quantum correlations quantified by $P(a, b|x, y)$, where x and y are selected from a set of $\{0, 1\}^{2^n}$ measurements, can only be reproduced by classical means if $\mathcal{O}(2^n)$ number of bits are exchanged between Alice and Bob [54].

Let us analyze the implication of this for the quantum learning algorithms described in the paper versus classical learning schemes. The communication flow in the quantum

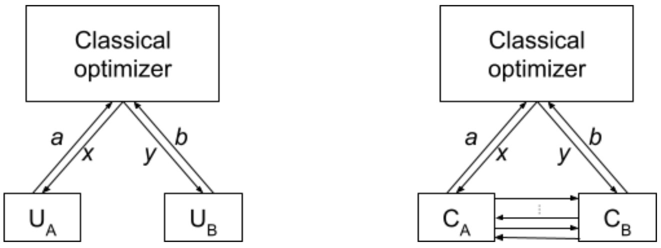


FIG. 8. Comparison of quantum (left) vs classical (right) learning. U_A and U_B are unitaries that act on entangled Bell pairs distributed between Alice and Bob. C_A and C_B are operations on random classical bit strings belonging to Alice and Bob.

learning scheme is shown in the left part of Fig. 8. Once the Bell pairs are created, the classical computer executing the optimization scheme can separately send x and y to Alice and Bob respectively. The measurements a and b are then collectively processed by the classical optimizer to produce the input to the quantum computer for the next iteration.

Now consider how one would simulate the quantum learning algorithm classically. Let us assume that, as is usually the case, the classical algorithm is deterministic in nature, i.e., for every unique input, there is one unique output. In general, the input to the classical generator is a vector of random bits, which has to contain at least n bits for each random variable to produce as many unique outputs as produced in the quantum case. Distribute these bits equally between Alice and Bob. Any classical operation can be written as a combination of operations that only acts on the bits of Alice or Bob separately, and operations that act on both sets of bits. The latter can be further modeled as operations on the bits belonging to Alice, followed by communication to Bob and then operations on just on the bits belonging to Bob, and vice versa (right part of Fig. 8). Then, without loss of generality, we can restrict Alice's operation C_A to be of the form $C_A(x, d_{AB})$, i.e., parametrized by x from the optimizer, and the sequence of bits d_{AB} communicated from Bob to Alice. Similarly, for Bob we have $C_B(y, d_{BA})$. Note that d_{AB} can contain information about y and d_{BA} about x, so we have not restricted the form of the classical generator in any fashion. Then, in order to simulate $P(a, b|x, y)$ for arbitrary x and y sent by the classical optimizer, it follows that the amount of communication between Alice and Bob, $d_{BA} + d_{AB}$, will scale as $\mathcal{O}(2^n)$. Next, let us see the concrete implications of this argument by considering a 2-layer feed-forward neural network architecture as shown in Fig 9. We assume Alice and Bob share the neurons at each layer and their job is to jointly compute $o = W\chi + b$. Writing it in blocks as

$$\begin{bmatrix} o_A \\ o_B \end{bmatrix} = \begin{bmatrix} W_{\mathcal{A}\mathcal{A}} & W_{\mathcal{A}\mathcal{B}} \\ W_{\mathcal{B}\mathcal{A}} & W_{\mathcal{B}\mathcal{B}} \end{bmatrix} \begin{bmatrix} \chi_{\mathcal{A}} \\ \chi_{\mathcal{B}} \end{bmatrix} + \begin{bmatrix} b_A \\ b_B \end{bmatrix},$$

where $W_{\mathcal{A}\mathcal{A}}, W_{\mathcal{B}\mathcal{A}}, \chi_{\mathcal{A}}, b_A$ belongs to Alice and $W_{\mathcal{A}\mathcal{B}}, W_{\mathcal{B}\mathcal{B}}, \chi_{\mathcal{B}}, b_B$ belongs to Bob, Alice locally stores $W_{\mathcal{A}\mathcal{A}}\chi_{\mathcal{A}}$ and sends $W_{\mathcal{B}\mathcal{A}}\chi_{\mathcal{A}}$ to Bob. Similarly, Bob locally stores $W_{\mathcal{B}\mathcal{B}}\chi_{\mathcal{B}}$ and sends $W_{\mathcal{A}\mathcal{B}}\chi_{\mathcal{B}}$ to Alice. After the communication, Alice receives $W_{\mathcal{A}\mathcal{B}}\chi_{\mathcal{B}}$ and computes $o_A = W_{\mathcal{A}\mathcal{A}}\chi_{\mathcal{A}} + W_{\mathcal{A}\mathcal{B}}\chi_{\mathcal{B}} + b_A$, and Bob does so similarly. In this process, Alice sends $\mathcal{O}(|B|)$

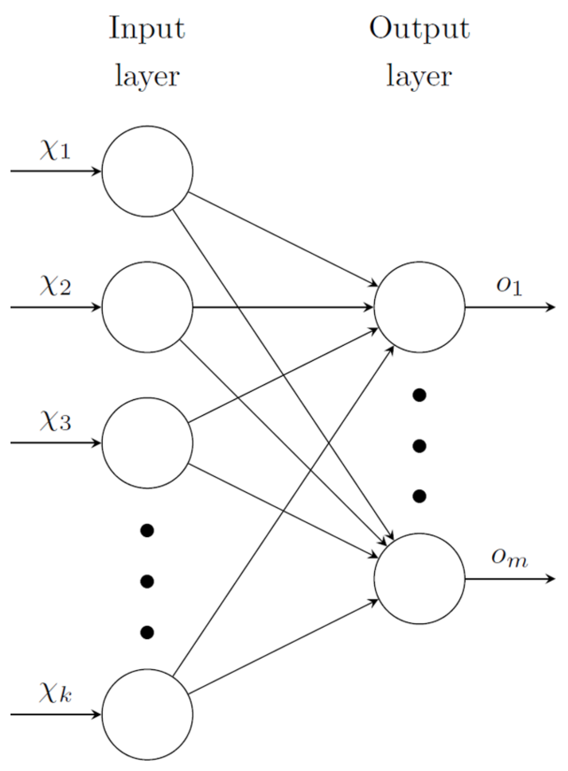


FIG. 9. Feed-forward neural network with dimension $[k, m]$.

bits to Bob and Bob sends $\mathcal{O}(|A|)$ bits to Alice. Since $|A| + |B| = m$, $O(m)$ bits of communication occur.

Thus, if the generator is represented by a deep feed-forward neural network where each layer has at most m neurons, then the number of layers has to scale exponentially asymptotically as $\mathcal{O}(2^n/m)$ to reproduce certain probability distributions generated by experiments on the Bell pairs. This ultimately implies that a neural network will have to have exponentially scaling depth in the precision of the output.

In the argument above, x and y belong to $\{0, 1\}^{2^n}$. For the quantum generator, they can be more compactly specified as angles to the ansatzes specified by U_A and U_B respectively, which can even consist of a constant number of layers. While the full set of measurements takes a circuit of up to depth $\mathcal{O}(2^n)$ to realize, the number of accessible measurements also grows exponentially with depth.

We note that while it is true that qopula circuits exhibit the complexity separations outlined above against classical computation, these are arguments for the worst case and it may not apply to the ones relevant for a particular learning problem. While determining the generality of these exponential communication and computational separations is a topic of ongoing research, the reasoning above clearly shows that the quantum computer expands the space of joint probability distributions that can be efficiently explored for learning.

IX. FUTURE WORK

GANs are prone to mode collapse and nonconvergence, and as such many improvements have been made to mitigate these issues. While our pedagogical case did not suffer from them, they may occur with more complex models and problems. Therefore it would be interesting to see if our framework can benefit from the GAN enhancements discussed in this section. For instance, one may use a critic (C) in place of a discriminator. Instead of using binary cross entropy (BCE) as loss function, one may instead use Wasserstein loss (W-Loss) [55]

$$\min_G \max_C V(G, C) = \mathbb{E}[C(x)] - \mathbb{E}[C(G(z)], \tag{24}$$

which tends to make model training more stable. W-Loss solves the vanishing gradient problem of the discriminator, in which the discriminator does not provide enough information for the generator to make progress in the BCE case.

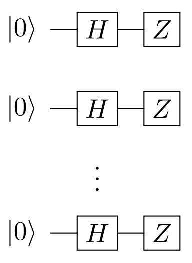
WGAN can be implemented straightforwardly in our case as we can replace the sigmoid function with a linear activation function such that the discriminator becomes the critic and replace BCE with W-Loss as the loss function. We note that this is different from Ref. [56] as we are learning classical data and the discriminator is classical, whereas in Ref. [56] both the discriminator and the generator are quantum.

Another way to enhance our model is the analog of the addition of noise vectors from the latent space as input to the classical GAN. Recall that in classical GAN, once a model is trained, this input noise vector completely determines the model output (see Sec. II). A significant improvement with the addition of input noise vectors is that it allows the controlled generation of data, such as changing specific features of the output by tweaking the noise vector. In our example, this would mean that one can generate samples with desired characteristics such as where all absolute returns are larger than a preset value, as in the case of a black swan event. Controlled generation allows for the emergence of desired features by updating the noise vector according to pre-trained feature classifiers. This would also allow more complex GAN architectures such as styleGAN [57].

A proposed circuit for such a generator from Sec. II allows one to calculate the expectation value of an observable with respect to the noise vector in the latent space,

$$G(z) = \langle z|U_G^\dagger O_G U_G|z\rangle. \tag{25}$$

$|z\rangle$ may be realized in the following way, where the Z gates are parametrized rotations defined by the noise vector z:



It is important to note that this model might be more costly to implement on quantum hardware. A circuit execution with 1000 shots corresponds to 1000 data samples in our current model architecture, whereas if using expectation values this would only correspond to one data sample. However, in the long run, such a model might outperform our current model due to the mitigation of problems mentioned above.

Moreover, it is not clear if such a model will be advantageous against classical cases in terms of complexity. While we have argued that producing the distribution in our case is difficult for a classical generator, it might not be the case for producing expectation values as this procedure is deterministic (up to small statistical noise from the measurement). For example, it has been argued that even the lowest depth version of the quantum approximate optimization algorithm (QAOA) can produce distributions, which are hard for classical computers [51]. However, the computational task of QAOA is to optimize the expectation value $\langle \boldsymbol{\gamma}, \boldsymbol{\beta}|C|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle$ of some objective function C and it is unclear if the low depth version of QAOA can outperform classical algorithms [58].

Another area of potential exploration is the development of higher-level ansatzes. Currently, the number of trainable parameters in our ansatz scales quadratically with the number of qubits. This is undesirable as we use more qubits to represent each dimension of data. It may be beneficial to develop ansatz analog to building blocks in classical neural networks (e.g., linear, batch norm), where operations are applied to the variables instead of the bits representing those variables.

Lastly, as discussed in Sec. V, instead of utilizing SPSA, the generator can be updated using gradient-based optimizers with the parameter-shift rule. This could allow for more efficient optimization through the cost function. Implementing the parameter-shift rule in parallel would reduce the runtime for quantum gradient estimation. However, this is only possible if the quantum hardware has enough qubits and low crosstalk noise. A networked quantum computer architecture would also make this possible.

X. DISCUSSION AND CONCLUSIONS

In our view, the theory of quantum machine learning (QML) has gone through three development phases. The first phase consists of using quantum computers to speed up subroutines of classical learning algorithms [59]. These algorithms most often used the quantum linear systems algorithm [34] as the basis of quantum advantage. The resource requirements for this class of algorithms are very high and it often requires loading classical data in a superposition. The second stage, which started with the advent of cloud-based machine learning, consists of heuristic techniques utilizing variational quantum circuits, which are applied to discriminative or generative problems [27,60]. While these have been experimentally demonstrated on very small problem sizes, these are heuristic in nature without theoretical support for their advantage over classical methods. Since they cannot be tested at scale, often their efficacy cannot even be numerically predicted.

The third phase of QML development is now under way. It consists of quantum algorithms with reasonable resource requirements. Moreover, the structure of circuits can often be shown to have advantage over the corresponding classical algorithms. For example, in Ref. [61], quantum advantage is proven for certain problems where the objective is achieving a specified worst-case prediction error. Reference [62] uses the hardness of the discrete log problem to show that a quantum classifier can achieve high accuracy and is robust to additive error where a classical classifier can do no better than random guessing. Perhaps the simplest example of advantage is the proof of separation in expressive power between commonly-used Bayesian networks and their minimal quantum extension shown in Ref. [63] based on results from quantum foundations. Our paper falls in this third phase. We anticipate that the future of practical QML will follow this path.

For the particular algorithm presented here, we can project when it will become competitive with classical techniques as follows. Typical structured datasets have dimension of order 10–100. With each dimension represented by 5-10 qubits, we expect our model to have production value if running on quantum hardware of 50 qubits or more.

In conclusion, we have demonstrated quantum generative learning for multivariate data using a QGAN and a QCBM. Our learning models utilize a variational ansatz that we have designed. Our ansatz has provable advantage in capturing correlations between the data over classical methods. We note that our training of the quantum generative models does not show the barren plateau problem and is resilient to noise. The outcome of the QGAN training reaches a better performance and is more stable than training of the equivalent classical GAN. We anticipate that our observation of a fundamental connection between modeling correlated distributions and quantum entanglement will become the standard technique for modeling multivariate data with quantum computers, and will be used as a basis for a wide variety of learning applications.

ACKNOWLEDGMENTS

This work is a collaboration between Fidelity Center for Applied Technology, Fidelity Labs, LLC., and IonQ Inc.

E.Z. and N.M. initiated the idea of doing generative modeling of a copula on quantum computers. E.Z. trained the QGAN, worked on the proof, and worked on the argument for quantum advantage. S.J. designed the qopula circuits, trained the QCBM, and worked on the argument for quantum advantage. D.B. worked on the argument for quantum advantage and contributed in all aspects. M.E. contributed to benchmarking the hardware and contributed in all aspects. J.K., J.N., N.P., K.S., K.W. were responsible for the quantum hardware. M.M. and A.S. contributed to scientific discussions. After E.Z. and S.J., all authors are listed in alphabetical order. The Fidelity publishing approval number for this paper is 995073.2.0.

APPENDIX A: REDUCED DENSITY MATRIX OF A COPULA STATE

Given a quantum state of the form

$$|c\rangle = \sum_{u_1,\cdots,u_d=0}^{m-1} \sum_{k_1,\cdots,k_d=0}^{1} e^{i\phi(u_1,\cdots,u_d,k_1,\cdots,k_d)} \frac{\sqrt{c(u_1/m,\cdots,u_d/m)}}{2^{d-1}} |u_1,k_1\rangle\cdots|u_d,k_d\rangle. \tag{A1}$$

with c a given discretized copula density, i.e.,

$$\sum_{u_1,\cdots u_{i-1},u_{i+1},u_d=0}^{m-1} c(u_1/m,\ldots,u_d/m) = \frac{1}{m} \quad \forall i. \tag{A2}$$

Since

$$\langle v_1, l_1|c\rangle = \sum_{u_2,\cdots,u_d=0}^{m-1} \sum_{k_2,\cdots,k_d=0}^{1} e^{i\phi(v_1,u_2,\cdots,u_d,l_1,k_2,\cdots,k_d)} \frac{\sqrt{c(v_1/m,\cdots,u_d/m)}}{2^{d-1}} |u_2,k_2\rangle\cdots|u_d,k_d\rangle, \tag{A3}$$

we can see that after tracing over the first register u_1, k_1,

$$\mathrm{Tr}_1(|c\rangle\langle c|) = \sum_{u_1,k_1}\sum_{u_2,\cdots,u_d}\sum_{k_2,\cdots,k_d} e^{i\phi(u_1,u_2,\cdots,u_d,k_1,k_2,\cdots,k_d)} \frac{\sqrt{c(u_1/m,u_2/m\cdots,u_d/m)}}{2^{d-1}} |u_2,k_2\rangle\cdots|u_d,k_d\rangle \tag{A4}$$

$$\sum_{u_2',\cdots,u_d'}\sum_{k_2',\cdots,k_d'} e^{-i\phi(u_1,u_2',\cdots,u_d',k_1,k_2',\cdots,k_d')} \frac{\sqrt{c(u_1/m,u_2'/m,\cdots,u_d'/m)}}{2^{d-1}} \langle u_2',k_2'|\cdots\langle u_d',k_d'|. \tag{A5}$$

Therefore,

$$\rho_d = \mathrm{Tr}_{1,\cdots,d-1}(|c\rangle\langle c|) \tag{A6}$$

$$= \sum_{u_1,\cdots,u_{d-1},k_1,\cdots,k_{d-1}} \sum_{u_d}\sum_{k_d} e^{i\phi(u_1,\cdots,u_{d-1},u_d,k_1,\cdots,k_{d-1},k_d)} \frac{\sqrt{c(u_1/m,\cdots,u_{d-1}/m,u_d/m)}}{2^{d-1}} |u_d,k_d\rangle \tag{A7}$$

$$\sum_{u_d'}\sum_{k_d'} e^{-i\phi(u_1,\cdots,u_{d-1},u_d',k_1,k_2,\cdots,k_{d-1},k_d')} \frac{\sqrt{c(u_1/m,\cdots,u_{d-1}/m,u_d'/m)}}{2^{d-1}} \langle u_d',k_d'|. \tag{A8}$$

We see that the requirement of $\rho_d = I/m$ is equivalent to the coefficient of $|u_d,k_d\rangle\langle u_d',k_d'|$ being 0 when $u_d \neq u_d'$ or $k_d \neq k_d'$. This can always be achieved by setting the phases $\phi(u_1,\cdots,u_d,k_1,\cdots,k_d)$ appropriately.

In total, by enforcing $\rho_i = I/m$ for all i, we will end up with $dm(2m-1)$ number of equations. However, the number of degrees of freedom that we have in setting the phases is $(2m)^d - 1$. Since $(2m)^d - 1 > dm(2m-1)$ whenever $d \geqslant 2$ and $m \geqslant 2$, we will always have enough degrees of freedom to ensure maximal entanglement in $|c\rangle$.

In practice, the extra qubits $|k_1\rangle,\cdots,|k_d\rangle$ are usually not needed unless the copula function c has extreme spikes in certain regions.

APPENDIX B: QCBM TRAINING

Here we discuss details of the QCBM training. Figure 10 shows the effect of changing the number of shots N_{shots} on the simulator. For $N_q = 2$, the cost function converges faster for fewer shots while this is reversed for $N_q = 6$. This further supports our observation in the text that for the QCBM at small system sizes, a noisy estimate of the cost function can play a favorable role in the optimization. For $N_q = 6$, one can also observe that the spread of the values towards the end of the optimization decreases due to the increased accuracy available from larger number of shots.

Figure 11 shows the marginal distributions obtained at the minimum value of the cost function achieved during optimization. Theoretically, the marginals should be uniform within statistical error. This is seen to be true even on the quantum hardware, revealing one reason for the robustness of the algorithm to noise.

APPENDIX C: PROOF OF QUANTUM ADVANTAGE BASED ON BELL'S THEOREM

Theorem 3. Simulating Algorithm II or Algorithm III on a classical deterministic computer using the qopula ansatz with n qubits per variable will take an amount of time $O(\exp(n))$.

In each iteration of the Algorithms II and III, the first step is to create n Bell pairs and distribute them between two registers A and B. Let us say the classical computer executing the optimization scheme sends parameter vectors x and y to the quantum computer. The unitary U_A acting on register A is a function of x and the unitary U_B is a function of y. From the

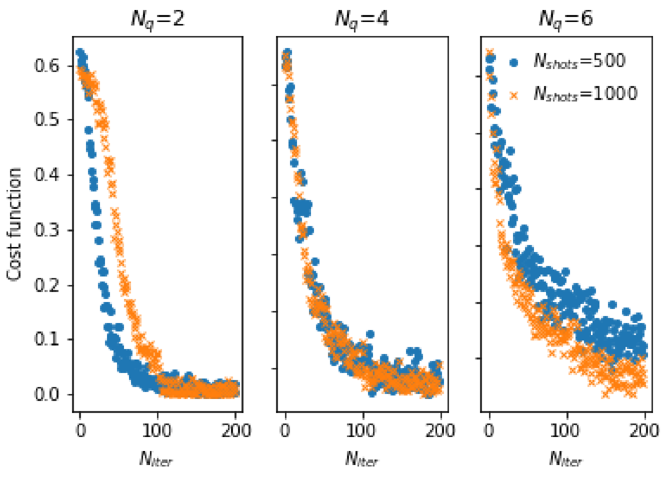


FIG. 10. Effect of increasing shots on the convergence of the QCBM for different system sizes.

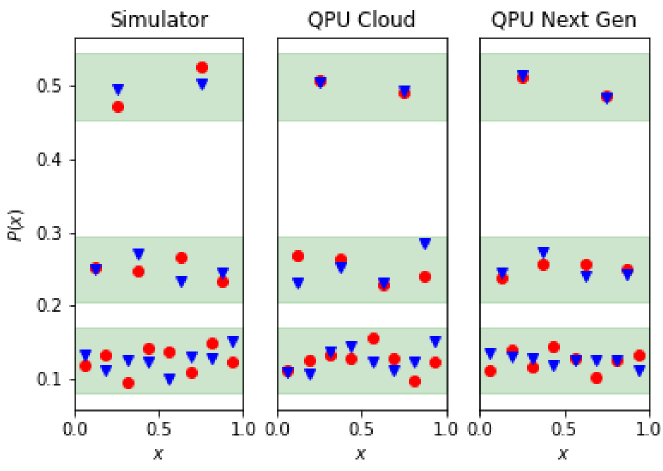


FIG. 11. Marginal distributions in the copula space on the three different backends used. The red circles correspond to the first variable and the blue triangles correspond to the second. The width of the green-shaded bands is the theoretical statistical uncertainty, which is $1/\sqrt{N_{\text{shots}}}$ and they are centered at $1/2^{N_q/2}$, where $N_q = 2$, 4, and 6 qubits from top to bottom in each subfigure. Here $N_{\text{shots}} = 500$.

point of view of the quantum computer, x and y are arbitrary. After execution of the qopula circuit, the measurements a and b from A and B respectively are then collectively processed by the classical optimizer to produce the input to the quantum computer for the next iteration. The measurements are then samples from a joint probability distribution, say $P(a, b|x, y)$.

Now consider how one would simulate these quantum learning algorithms classically. The classical algorithm is deterministic in nature, i.e., for every unique input, there is one unique output. In general, the input to the classical generator function is a vector of random bits, which has to contain at least n bits for each random variable to produce as many unique outputs as produced in the quantum case. Distribute these bits equally between two registers A and B. Any classical operation can be written as a combination of operations that only acts on the bits of A or B separately, and operations that act on both sets of bits. The latter can be further modeled as operations on the bits belonging to A, followed by communication to B and then operations on just on the bits belonging to B, and vice versa (right part of Fig. 8). Then, without loss of generality, we can restrict C_A to be of the form $C_A(x, d_{AB})$, i.e., parametrized by x from the optimizer, and the sequence of bits d_{AB} communicated from B to A. Similarly, for B we have $C_B(y, d_{BA})$. Note that d_{AB} can contain information about y and d_{BA} about x, so we have not restricted the form of the classical generator function in any fashion. Then, in order to simulate $P(a, b|x, y)$ for arbitrary x and y sent by the classical optimizer, it follows Theorem 4 of Ref. [54] that the amount of communication between A and B, $d_{BA} + d_{AB}$, will scale as $\mathcal{O}(2^n)$. Thus, the computation of C_A and C_B, which is required to simulate Algorithms 1 and 2 will also take time $\mathcal{O}(2^n)$.

[1] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: With Applications in R*, (Springer, New York, 2013).

[2] H. Markowitz, Portfolio selection, J. Financ. **7**, 77 (1952).

[3] S. Zacks, *Introduction to Reliability Analysis: Probability Models and Statistical Methods* (Springer-Verlag, New York, 1992).

[4] C. C. Aggarwal, *Recommender Systems: The Textbook*, 1st edition (Springer, New York, 2016).

[5] H. v. Storch and F. W. Zwiers, *Statistical Analysis in Climate Research* (Cambridge University Press, Cambridge, 1999).

[6] C. Palmer, Statistics of medical imaging, Health Phys. **104**, 332 (2013).

[7] R. Aloui, M. S. B. Aissa, and D. K. Nguyen, Global financial crisis, extreme interdependencies, and contagion effects: The role of economic structure?, J. Bank. Financ. **35**, 130 (2011); D. Kenourgios, A. Christopoulos, and D. I. Dimitriou, Asset markets contagion during the global financial crisis, Multinatl. Financ. J. **17**, 49 (2013); F. Serinaldi, A. Bárdossy, and C. G. Kilsby, Upper tail dependence in rainfall extremes: Would we know it if we saw it? Stoch. Environ. Res. Risk Assess.**29**, 1211 (2015).

[8] M. Sklar, *Fonctions de repartition a n dimensions et leurs marges*, Publications de l'Institut Statistique de l'Université de Paris (1959).

[9] R. W. van den Goorbergh, C. Genest, and B. J. Werker, Bivariate option pricing using dynamic copula models, Insur. Math. Econ. **37**, 101 (2005).

[10] R. T. Kilgore and D. B. Thompson, Estimating joint flow probabilities at stream confluences by using copulas, Trans. Res. Rec. **2262**, 200 (2011).

[11] J. Lapuyade-Lahorgue, J.-H. Xue, and S. Ruan, Segmenting multi-source images using hidden Markov fields with copula-based multivariate statistical distributions, IEEE Trans. Image Process. **26**, 3187 (2017).

[12] H. Joe, *Multivariate Models and Multivariate Dependence Concepts* (Chapman and Hall/CRC, New York, 1997).

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, Generative adversarial nets, in *Advances in Neural Information Processing Systems* (NIPS 27) (2014).

[14] D. P. Kingma and M. Welling, Auto-encoding variational Bayes, in 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings, edited by Y. Bengio and Y. LeCun.

[15] A. Antoniou, A. Storkey, and H. Edwards, Data augmentation generative adversarial Networks, arXiv:1711.04340.

[16] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling, in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain, NIPS'16* (Curran Associates Inc., USA, 2016), pp. 82–90.

[17] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, Photo-realistic single image super-resolution using a generative adversarial network, in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, 2017), pp. 105–114.

[18] S. Arora, A. Risteski, and Y. Zhang, Do GANs learn the distribution? Some theory and empirics, in *International Conference on Learning Representations* (ICLR, 2018).
[19] *Common problems—Generative adversarial networks—Google Developers*, https://developers.google.com/machine-learning/gan/problems.
[20] S. Aaronson and A. Arkhipov, The computational complexity of linear optics, in *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing, STOC '11* (Association for Computing Machinery, USA, 2011), pp. 333–342.
[21] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner, J. M. Martinis, and H. Neven, Characterizing quantum supremacy in near-term devices, Nat. Phys. **14**, 595 (2018).
[22] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell *et al.*, Quantum supremacy using a programmable superconducting processor, Nature (London) **574**, 505 (2019).
[23] S. Cheng, J. Chen, and L. Wang, Information perspective to probabilistic modeling: Boltzmann machines versus Born machines, Entropy **20**, 583 (2018).
[24] M. Benedetti, D. Garcia-Pintos, O. Perdomo, V. Leyton-Ortega, Y. Nam, and A. Perdomo-Ortiz, A generative modeling approach for benchmarking and training shallow quantum circuits, npj Quantum Inf. **5**, 45 (2019).
[25] P.-L. Dallaire-Demers and N. Killoran, Quantum generative adversarial networks, Phys. Rev. A **98**, 012324 (2018).
[26] S. Lloyd and C. Weedbrook, Quantum Generative Adversarial Learning, Phys. Rev. Lett. **121**, 040502 (2018).
[27] D. Zhu, N. M. Linke, M. Benedetti, K. A. Landsman, N. H. Nguyen, C. H. Alderete, A. Perdomo-Ortiz, N. Korda, A. Garfoot, C. Brecque *et al.*, Training of quantum circuits on a hybrid quantum computer, Sci. Adv. **5**, eaaw9918 (2019).
[28] C. Zoufal, A. Lucchi, and S. Woerner, Quantum generative adversarial networks for learning and loading random distributions, npj Quantum Inf. **5**, 103 (2019).
[29] L. Hu, S.-H. Wu, W. Cai, Y. Ma, X. Mu, Y. Xu, H. Wang, Y. Song, D.-L. Deng, C.-L. Zou, and L. Sun, Quantum generative adversarial learning in a superconducting quantum circuit, Sci. Adv. **5**, eaav2761 (2019).
[30] H.-L. Huang, Y. Du, M. Gong, Y. Zhao, Y. Wu, C. Wang, S. Li, F. Liang, J. Lin, Y. Xu, R. Yang, T. Liu, M.-H. Hsieh, H. Deng, H. Rong, C.-Z. Peng, C.-Y. Lu, Y.-A. Chen, D. Tao, X. Zhu, and J.-W. Pan, Experimental quantum generative adversarial networks for image generation, Phys. Rev. Appl. **16**, 024051 (2021).
[31] M. S. Rudolph, N. B. Toussaint, A. Katabarwa, S. Johri, B. Peropadre, and A. Perdomo-Ortiz, Generation of High-Resolution Handwritten Digits with an Ion-Trap Quantum Computer, Phys. Rev. X **12**, 031010 (2022).
[32] L. K. Grover, A fast quantum mechanical algorithm for database search, in *Annual ACM Symposium on Theory of Computing, STOC '96* (Association for Computing Machinery, USA, 1996), pp. 212–219.
[33] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM J. Comput. **26**, 1484 (1997).
[34] A. W. Harrow, A. Hassidim, and S. Lloyd, Quantum Algorithm for Linear Systems of Equations, Phys. Rev. Lett. **103**, 150502 (2009).
[35] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, in *ICLR (Poster)* (ICLR, 2015).
[36] J.-G. Liu and L. Wang, Differentiable learning of quantum circuit Born machines, Phys. Rev. A **98**, 062324 (2018).
[37] F. Durante, J. Fernández-Sánchez, and C. Sempi, A topological proof of Sklar's theorem, Appl. Math. Lett. **26**, 945 (2013).
[38] N. Patki, R. Wedge, and K. Veeramachaneni, The synthetic data vault, in *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)* (IEEE, 2016), pp. 399–410; N. Tagasovska, D. Ackerer, and T. Vatter, Copulas as high-dimensional generative models: Vine copula autoencoders, in *Advances in Neural Information Processing Systems*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett (Curran Associates, New York, 2019).
[39] D. Maslov, Basic circuit compilation techniques for an ion-trap quantum machine, New J. Phys. **19**, 023035 (2017).
[40] M. Enríquez, I. Wintrowicz, and K. Życzkowski, Maximally entangled multipartite states: A brief survey, J. Phys.: Conf. Ser. **698**, 012003 (2016).
[41] K. Wright, K. M. Beck, S. Debnath, J. M. Amini, Y. Nam, N. Grzesiak, J.-S. Chen, N. C. Pisenti, M. Chmielewski, C. Collins *et al.*, Benchmarking an 11-qubit quantum computer, Nat. Commun. **10**, 5464 (2019).
[42] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, Evaluating analytic gradients on quantum hardware, Phys. Rev. A **99**, 032331 (2019).
[43] J. C. Spall, A stochastic approximation technique for generating maximum likelihood parameter estimates, American Control Conference 1161 (1987); Multivariate stochastic approximation using a simultaneous perturbation gradient approximation, IEEE Trans. Autom. Control **37**, 332 (1992).
[44] S. Johri, S. Debnath, A. Mocherla, A. SINGK, A. Prakash, J. Kim, and I. Kerenidis, Nearest centroid classification on a trapped ion quantum computer, npj Quantum Inf. **7**, 122 (2021).
[45] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, Barren plateaus in quantum neural network training landscapes, Nat. Commun. **9**, 4812 (2018).; M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles, Cost function dependent barren plateaus in shallow parametrized quantum circuits, *ibid.* **12**, 1791 (2021).
[46] J. A. Peacock, Two-dimensional goodness-of-fit testing in astronomy, Mon. Not. R. Astron. Soc. **202**, 615 (1983).
[47] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, PyTorch: An imperative style, high-performance deep learning library, in *Advances in Neural Information Processing Systems 32*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, (Curran Associates, New York, 2019), pp. 8024–8035.
[48] M. J. Bremner, R. Jozsa, and D. J. Shepherd, Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy, Proc. R. Soc. London A **467**, 459 (2011).
[49] The polynomial hierarchy class PH [50] is defined to be the union of an infinite tower of increasing classes Δ_k, $k = 1, 2, ...$, in which $\Delta_1 = P$ and $\Delta_{k+1} = P^{N\Delta_k}$.
[50] S. Arora and B. Barak, *Computational Complexity: A Modern Approach*, 1st edition (Cambridge University Press, New York, 2009).

[51] E. Farhi and A. W. Harrow, Quantum supremacy through the quantum approximate optimization algorithm, arXiv:1602.07674.

[52] B. Coyle, D. Mills, V. Danos, and E. Kashefi, The Born supremacy: Quantum advantage and training of an Ising Born machine, npj Quantum Inf. **6**, 60 (2020).

[53] A. B. Watts, R. Kothari, L. Schaeffer, and A. Tal, Exponential separation between shallow quantum circuits and unbounded fan-in shallow classical circuits, in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing* (Association for Computing Machinery, New York, 2019), pp. 515–526.

[54] G. Brassard, R. Cleve, and A. Tapp, Cost of Exactly Simulating Quantum Entanglement with Classical Communication, Phys. Rev. Lett. **83**, 1874 (1999).

[55] M. Arjovsky, S. Chintala, and L. Bottou, Wasserstein generative adversarial networks, in *Proceedings of the 34th International Conference on Machine Learning*, edited by D. Precup and Y. W. Teh (PMLR, 2017), pp. 214–223.

[56] S. Chakrabarti, H. Yiming, T. Li, S. Feizi, and X. Wu, Quantum Wasserstein generative adversarial networks, in *Advances in Neural Information Processing Systems*.

[57] T. Karras, S. Laine, and T. Aila, A style-based generator architecture for generative adversarial networks, IEEE Trans. Pattern. Anal. Mach. Intell. **43**, 4217 (2021).

[58] M. B. Hastings, Classical and quantum bounded depth approximation algorithms, Quantum Inf. Comput. **19**, 1116 (2019).

[59] N. Wiebe, D. Braun, and S. Lloyd, Quantum Algorithm for Data Fitting, Phys. Rev. Lett. **109**, 050505 (2012); P. Rebentrost, M. Mohseni, and S. Lloyd, Quantum Support Vector Machine for Big Data Classification, *ibid.* **113**, 130503 (2014).

[60] I. Cong, S. Choi, and M. D. Lukin, Quantum convolutional neural networks, Nat. Phys. **15**, 1273 (2019); E. Farhi and H. Neven, Classification with quantum neural networks on near term processors, arXiv:1802.06002.

[61] H.-Y. Huang, R. Kueng, and J. Preskill, Information-Theoretic Bounds on Quantum Advantage in Machine Learning, Phys. Rev. Lett. **126**, 190505 (2021).

[62] Y. Liu, S. Arunachalam, and K. Temme, A rigorous and robust quantum speed-up in supervised machine learning, Nat. Phys. **17**, 1013 (2021).

[63] X. Gao, E. R. Anschuetz, S.-T. Wang, J. I. Cirac, and M. D. Lukin, Enhancing Generative Models via Quantum Correlations, Phys. Rev. X **12**, 021037 (2022).