Generative model benchmarks for superconducting qubits

Kathleen E. Hamilton, Eugene F. Dumitrescu, and Raphael C. Pooser

Generative model benchmarks for superconducting qubits[*]

Kathleen E. Hamilton, Eugene F. Dumitrescu, and Raphael C. Pooser
Computer Science and Engineering Division, Oak Ridge National Laboratory,
One Bethel Valley Road, Oak Ridge, TN 37831 USA
(Dated: May 6, 2019)

In this work we experimentally demonstrate how generative model training can be used as a benchmark for small (< 5 qubits) quantum devices. Performance is quantified using three data analytic metrics: the Kullback-Leiber divergence, and two adaptations of the F_1 score. Using the 2×2 Bars and Stripes dataset, we train several different circuit constructions for generative modeling with superconducting qubits. By taking hardware connectivity constraints into consideration, we show that sparsely connected shallow circuits out-perform denser counterparts on noisy hardware.

I. INTRODUCTION

The increasing diversity of programmable noisy intermediate-scale quantum (NISQ) devices has exposed the need for a unified set of benchmark tasks which assess application-centric device capabilities. Quantum machine learning (QML) has been presented as a useful tool for benchmarking quantum hardware [1]. Generative model training was recently proposed as a benchmark task [2–4] for NISQ devices. In this work we use non-adversarial training of a generative model to benchmark superconducting qubit devices. This approach to generative modeling requires training of a single quantum circuit, making it more practical for implementation on current devices.

Generative models, such as adversarial networks [5], have recently spurred significant interest in the development of quantum circuit analogues [6, 7] and adversarial quantum circuits training [8–10]. The quantum-circuit Born machine (QCBM) is a generative model constructed as a quantum circuit [3, 4, 11]. Numerical simulation of QCBMs, constructed using the hardware efficient circuit ansatz [12] with many (> 10) entangling layers and trained with non-adversarial methods, using data-driven quantum circuit learning (DDQCL), introduced in [4] can reproduce several classes of discrete and continuous distributions [3]. Here we utilize the gradient-based DDQCL methods of [3].

In contrast, NISQ devices accumulate errors due to imperfect gates and environmental decoherence effects. As such, we expect that the depth of useful NISQ circuits to be limited. After this point, the output becomes random as dictated by the noise. QML-based benchmarking

is a practical method to establish the maximal circuit depth. To experimentally test this hypothesis we train a set of shallow circuits (< 3 entangling layers) which are deployed on IBM's Toyko chip which has 20 superconducting qubits. The entangling layers of all circuits considered can be embedded in a two-rung ladder geometry (e.g. IBM's Melbourne chip [13]) ensuring portability of our benchmark.

Guidelines for benchmarking digital QML algorithms have been proposed [14] in terms of the output correctness. For generative models, correctness refers to the model's ability to reproduce the target distribution. Performance is therefore naturally captured by statistical measures describing the similarity of two distributions, such as the Kullback-Leibler divergence and the F_1 score.

We evaluated several QCBM circuits on superconducting qubits accessed through the IBM Quantum Hub cloud interface. The QCBM circuit, training methodology, and performance metrics are described in Section II. In Section III we discuss the interplay between circuit design and QCBM performance. Noisy qubits are introduced into QCBM training in Section III B. While previous experimental results for machine-learning based benchmarks were executed on direct-access ion trap hardware which can implement all-to-all connectivity [4], our results show comparable performance in superconducting qubits as measured by the Kullback-Leiber divergence.

II. QUANTUM CIRCUIT BORN MACHINES

A parametrized quantum circuit defining a particular variational manifold of quantum states is referred to as an *ansatz*. In this work, as in [3], QCBM training is performed with circuits inspired by the hardware efficient ansatz originally applied in the context of the variational quantum eigensolver algorithm [12] (see Figure 1). The BAS(2,2) dataset contains six 2×2-pixel black and white striped images. Each image is represented in the computational basis of a 4-qubit register by fixing a qubit-pixel mapping, and associating black (white) pixels with the states $|0\rangle (|1\rangle)$ (see Appendix C).

While the entangling design introduced in [3] contains enough complexity to represent the dataset, for larger image sizes it can require a high degree of qubit connec-

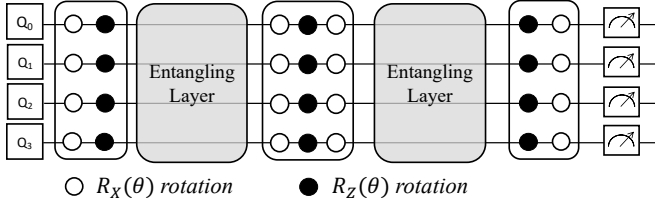tivity that is not available on current superconducting devices.

FIG. 1. The general circuit construction of a QCBM introduced in [4] is based on the hardware efficient variational quantum eigensolver ansatz of [12].

To generate BAS(2,2) we train three different ansatz (shown in Figure 1) whose entangling layers are illustrated in Figure 2. Each circuit is defined on a 4 qubit register and specified by the number of entangling layers (L) and the number of CNOT gates contained within each entangling layer (d_C). Current hardware's fixed connectivity presents a challenge when mapping arbitrary datasets. The $d_C = 2$ and $d_C = 4$ entangling layers conform to IBM's layout, i.e. by restricting CNOT gates to the edges of a 4 site square plaquette. The $d_C = 2$ layers are a sparser circuit construction and only take ~ 200ns to apply. As CNOTs within a single plaquette cannot be simultaneously applied, we decompose the $d_C = 4$ layer into 2 separate plaquette edge coverings. Thus the $d_C = 4$ circuit takes ~ 400 ns to apply, adding additional decoherence compared to $d_C = 2$. Additionally, since plaquettes may be covered in two ways as shown in Figure 2, alternating the two patterns results in a heterogeneous entangling layers structure for $d_C = 2$. For reference we also use the Chow-Liu tree-based design of [3] to define circuits with $d_C = 3$, though this entangling layer is not embeddable in a single square plaquette.
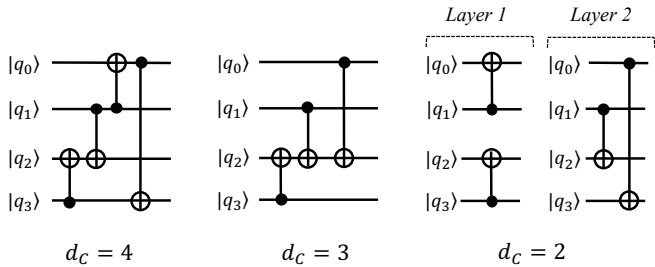

FIG. 2. The CNOT gate sets used to define individual entangling layers. The $d_C = 3$ entangling is the Chow-Liu tree-based design introduced in [3].

Many methods exist for training implicit generative models [15]. In this work the rotational parameters are optimized using Adam [16]. Overall we follow the training methods described in [3]: relying on the maximum mean discrepancy (MMD) [17] to define a loss function for circuit training and using the same unbiased estimator to evaluate the gradient. In this work we are modeling

a known target distribution: we know it is uniform, we can identify the binary states contained in the distribution, and we may sample classically from the distribution without error.

The target distribution $p(x)$ is fixed and defined by the BAS(2,2) dataset. For a given set of rotational parameters we execute a given QCBM circuit, draw N_{shots} samples and label this distribution $q(x)$. To compare $q(x)$ to $p(x)$ and quantify the overall QCBM performance we rely on the Kullback-Leiber (KL) divergence. The KL divergence compares the two sampled distributions $p(x), q(x)$ by computing the density ratio $p(x_i)/q(x_i)$ of individual states,

$$ \mathrm{D}(p|q) = \sum_i p(x_i) \log\left(\frac{p(x_i)}{q(x_i)} \right). \tag{1} $$

As $p(x_i)/q(x_i) \to 1$, $\mathrm{D}(p|q) \to 0$, but $\mathrm{D}(p|q)$ diverges if $p(x_i) \neq 0$ and $q(x_i) = 0$.

In addition, the performance metric known as the F_1 score [18] can be used. We modify the F_1 score to define an individual value assigned to each BAS(n,m) state and treat the dataset as a $2^m + 2^n - 2$ class system. This metric is analogous to measuring the fidelity of each state and we use it to gain insight into how well each circuit ansatz can learn the states of the BAS(n,m) system. The metric is complementary to $\mathrm{D}(p|q)$, giving insight into which eigenstates of the distribution are responsible for high KL values. Further details are given in Appendix A.

We note that the number of samples drawn from a circuit during training can be different from the number of samples taken when evaluating performance metrics. When evaluating the KL divergence we keep the number of shots fixed at $N_{shots} = 2048$.

III. RESULTS

We first use numerical simulation to train each QCBM in order to estimate how well the target distribution can be learned in the absence of noise. Circuits were constructed using the entangling layers shown in Figure 2 and trained using the QASM simulator available in IBM Qiskit-Terra. We limit the number of entangling layers to $L = 2$, for a total of 6 circuits. Each circuit is trained for 100 steps of Adam with learning rate $\alpha = 0.2$ and decay rates ($\beta_1 = 0.9, \beta_2 = 0.999$). The MMD loss function is calculated using Gaussian kernels with $\sigma = 0.1$. Figure 3 shows the overall performance of the 3 circuit ansatz with noiseless qubits for $L = 1, 2$ when $N_{shots} = 1024$ shots are drawn during training. For each set of rotational parameters, we evaluate the KL divergence of a given circuit 10 times at every training step with $N_{shots} = 2048$ and report the arithmetic mean value of $\mathrm{D}(p|q)$.

A. QCBM training with noiseless qubits

For each value of $\{d_C, L, N_{shots}\}$ a circuit was trained from a random intialization for $\{\theta^{(t=0)}\}$. Tables I and II show that for most circuits the shot size used during training has a modest effect on performance for the same circuit (fixed d_C, L), however different shot sizes will lead to different trajectories through $\{\theta\}$-space during training. In particular, the large discrepancy for $d_C = 3, L = 2$ between $N_{shots} = 512$ and $N_{shots} = 2048$ is most likely due to $\{\theta\}$ getting trapped in a sub-optimal minimum. For context, we also trained a non-entangling $L = 0, d_C = 0$ circuit. With $N_{shots} = 1024$ this circuit reached a minimum value of $D(p|q) = 1.0(1)$.

TABLE I. $\min(\langle D(p|q)\rangle)$ for $L = 1$ circuits simulated on noiseless qubits. Mean calculated over 10 independent metric evaluations.

L	N_{shots}	$d_C = 2$	$d_C = 3$	$d_C = 4$
1	512	0.95 ± 0.05	0.33 ± 0.02	0.24 ± 0.01
1	1024	0.93 ± 0.03	0.34 ± 0.01	0.23 ± 0.01
1	2048	0.93 ± 0.03	0.33 ± 0.01	0.23 ± 0.01

TABLE II. $\min(\langle D(p|q)\rangle)$ for $L = 2$ circuits simulated on noiseless qubits. Mean calculated over 10 independent metric evaluations.

L	N_{shots}	$d_C = 2$	$d_C = 3$	$d_C = 4$
2	512	0.013 ± 0.004	0.06 ± 0.01	0.02 ± 0.01
2	1024	0.088 ± 0.008	0.01 ± 0.01	0.02 ± 0.01
2	2048	0.011 ± 0.003	0.13 ± 0.01	0.01 ± 0.01

In general, Tables I and II show that increasing the complexity of a circuit by increasing the number of rotational parameters will improve performance. For example, the $d_C = 2, L = 2$ (28 rotational parameters) and $d_C = 4, L = 1$ (16 rotational parameters) circuits contain the same set of CNOT gates, however the better performance is measured with the $d_C = 2, L = 2$ circuit.
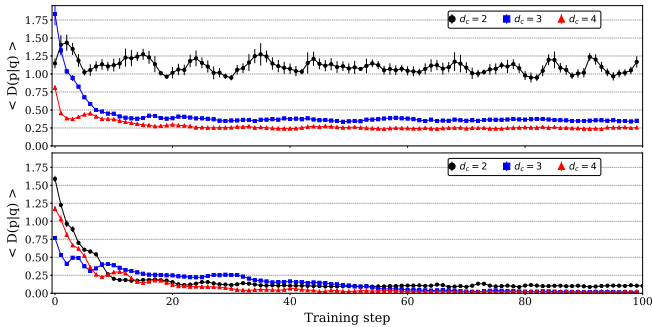

FIG. 3. KL divergence as a function of training step using $N_{shots} = 1024$ during training. Top (bottom) panel corresponds to $L = 1(2)$ entangling layers.

In Figure 3, training reduces the value of $\langle D(p|q)\rangle$ for the $d_C = 3, 4, L = 1$ circuits, while $\langle D(p|q)\rangle$ of the $d_C =$

$2, L = 1$ circuit fluctuates about a quasi-steady mean value ~ 1.1. With qubits being entangled pairwise, this ansatz generates a state manifold of the tensor product of two Bell states, up to local rotations. This tensor product structure lacks the complexity to fully learn and describe all of the BAS(2,2) states. The F_1 score supports this claim. In Appendix B we provide additional results for training with smaller learning rates.

In Figure 4, the individual F_1 score for each BAS(2,2) state is plotted as a function of training step. For the $(d_C = 2, L = 1)$ circuit, it is clear that the QCBM never learns the states $|1010\rangle$ or $|0101\rangle$.
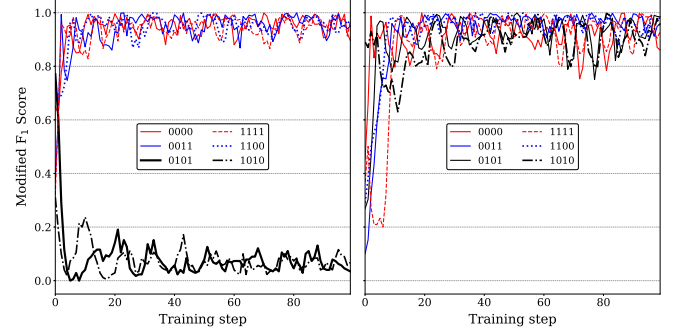

FIG. 4. The F_1 score for each of the 6 BAS(2,2) states evaluated with $N_{shots} = 2048$ at each training step for circuits trained with $N_{shots} = 1024$: (Left) $(d_C = 2, L = 1)$ circuit, (Right) $(d_C = 2, L = 2)$ circuit.

We deploy the circuits with trained noiseless parameters on the Tokyo chip to evaluate circuit performance in the presence of noise. While we leave more detailed discussion about circuit optimization in the presence of noise to Section IV, we show several examples here of how the behavior of $\langle D(p|q)\rangle$ is affected by the addition of noise. Many circuits show a general offset for $\langle D(p|q)\rangle$, but the behavior on noisy qubits can be substantially different from simulation. When the QCBM is actively learning (< 30 training steps) parameter updates which result in large fluctuations on noisless qubits (see Figure 5) will only result in small changes in $\langle D(p|q)\rangle$ on noisy qubits. When the QCBM training has converged (> 60 training steps) $\langle D(p|q)\rangle$ reaches a quasi-stationary value for most circuits (c.f. Figure 3). When deployed on hardware, noise can degrade the efficacy of training on noiseless qubits (see Figure 7, top). In contrast, the $d_C = 2, L = 2$, or $d_C = 3, L = 1$ circuits reach a quasi-stationary value of $\langle D(p|q)\rangle$ (see Figures 5 and 6) that is lower than the starting value.

TABLE III. $\min(\langle D(p|q)\rangle)$ circuits evaluated on IBM Tokyo. Mean calculated over 10 independent metric evaluations.

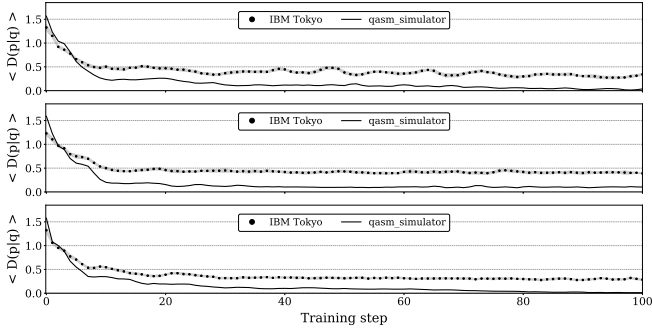| L	N_{shots}	$d_C = 2$	$d_C = 3$	$d_C = 4$
1	512	0.91 ± 0.01	0.64 ± 0.01	0.59 ± 0.02
1	1024	0.81 ± 0.02	0.60 ± 0.02	0.54 ± 0.01
1	2048	0.86 ± 0.01	0.57 ± 0.02	0.58 ± 0.01

FIG. 5. Comparison of $\langle \mathrm{D}(p|q)\rangle$ for 10 circuit evaluations of the $d_C = 2, L = 2$ circuit ansatz deployed on noiseless qubits (black, solid) and noisy qubits (black, circles). (Top) Trained with $N_{shots} = 512$, (Middle) $N_{shots} = 1024$, and (Bottom) $N_{shots} = 2048$. The standard deviation of $\langle \mathrm{D}(p|q)\rangle$ is shown by the grey shaded regions.

TABLE IV. $\min(\langle \mathrm{D}(p|q)\rangle)$ circuits evaluated on IBM Tokyo.

L	N_{shots}	$d_C = 2$	$d_C = 3$	$d_C = 4$
2	512	0.27 ± 0.02	0.48 ± 0.02	0.64 ± 0.02
2	1024	0.39 ± 0.01	0.39 ± 0.01	0.53 ± 0.01
2	2048	0.28 ± 0.02	0.59 ± 0.01	0.52 ± 0.01

In Tables III and IV we report the best metric values for each d_C, L and N_{shots} value. The smallest KL value was found with the $d_C = 2, L = 2$ circuit. When deployed on hardware, increasing the number of rotational parameters improves performance for the $d_C = 2, 3$ circuits, but not for $d_C = 4$ circuits.

B. QCBM training with noisy qubits

The experiments described in Section III explored how closely the value $\mathrm{D}(p|q)$ would follow the noiseless learning when measured with noisy qubits. In this section,
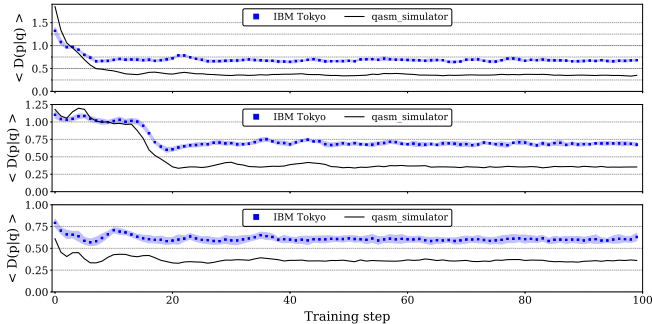

FIG. 6. Comparison of $\langle \mathrm{D}(p|q)\rangle$ for 10 circuit evaluations of the $d_C = 3, L = 1$ circuit ansatz deployed on noiseless qubits (black) and noisy qubits (blue, squares). (Top) Trained with $N_{shots} = 512$, (Middle) $N_{shots} = 1024$, and (Bottom) $N_{shots} = 2048$. The standard deviation of $\langle \mathrm{D}(p|q)\rangle$ is shown by the blue shaded regions.
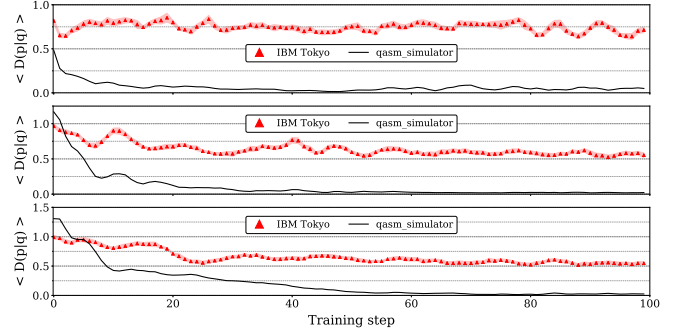

FIG. 7. Comparison of $\langle \mathrm{D}(p|q)\rangle$ for 10 circuit evaluations of the $d_C = 4, L = 2$ circuit ansatz deployed on noiseless qubits (black) and noisy qubits (red, triangles). (Top) Trained with $N_{shots} = 512$, (Middle) $N_{shots} = 1024$, and (Bottom) $N_{shots} = 2048$. The standard deviation of $\langle \mathrm{D}(p|q)\rangle$ is shown by the red shaded regions.

we investigate how well QCBM circuits can be trained with a finite number of steps utilizing noisy qubits. The experiments in this section allow us to explore hardware training within the rotational parameter space.

The goal of these tests is to determine if training a circuit ansatz with noisy qubits can improve the KL metric. In Table IV, the $(d_C = 2, L = 2)$ circuit reached a minimum value of $0.27(1)$ using theta values trained only with noiseless qubits. In this section the circuit initialization is chosen at equally spaced intervals from the first 60 training steps of each of the curves shown in Figure 5. This initializes the circuit with: completely random set of parameters $(S = 0)$, parameters that have undergone some optimization with Adam $(S = 10, 20, 30)$, or parameters that have mostly converged to a localized set of values $(S = 40, 50, 60)$. We only train the $(d_C = 2, L = 2)$ circuit, which was able to reach the lowest value of $\langle \mathrm{D}(p|q)\rangle$ with pre-trained parameters (see Table IV).

As in Section III A, the training is done with 3 shot sizes $N_{shots} = (512, 1024, 2048)$ but we evaluate KL metric with $N_{shots} = 2048$. The arithmetic mean value of $\mathrm{D}(p|q)$ is calculated from 10 circuit evaluations at every training step. We report the following values: the initial mean value $\langle \ldots \rangle_i$, the final value after training $\langle \ldots \rangle_f$ and the minimum KL value observed over training.

TABLE V. Trained on IBM Tokyo ($N_{shots} = 512$). Mean value calculated over 10 independent metric evaluations.

S	$\langle \mathrm{D}(p|q)\rangle_i$	$\langle \mathrm{D}(p|q)\rangle_f$	$\min \langle \mathrm{D}(p|q)\rangle$
0	1.29 ± 0.05	0.39 ± 0.02	0.39 ± 0.02
10	0.48 ± 0.02	0.35 ± 0.02	0.35 ± 0.02
20	0.46 ± 0.02	0.34 ± 0.02	0.34 ± 0.02
30	0.32 ± 0.02	0.34 ± 0.01	0.32 ± 0.02
40	0.41 ± 0.02	0.30 ± 0.02	0.29 ± 0.02
50	0.36 ± 0.02	0.30 ± 0.03	0.30 ± 0.03
60	0.36 ± 0.02	0.31 ± 0.02	0.30 ± 0.01
70	0.32 ± 0.02	0.30 ± 0.01	0.29 ± 0.01
80	0.29 ± 0.02	0.32 ± 0.03	0.29 ± 0.02

TABLE VI. Trained on IBM Tokyo ($N_{shots} = 1024$). Mean value calculated over 10 independent metric evaluations.

S	$\langle \mathrm{D}(p\|q)\rangle_i$	$\langle \mathrm{D}(p\|q)\rangle_f$	min $\langle \mathrm{D}(p\|q)\rangle$
0	1.28 ± 0.05	0.44 ± 0.02	0.43 ± 0.02
10	0.48 ± 0.03	0.44 ± 0.02	0.43 ± 0.01
20	0.49 ± 0.02	0.46 ± 0.02	0.43 ± 0.02
30	0.42 ± 0.02	0.44 ± 0.02	0.42 ± 0.02
40	0.45 ± 0.02	0.50 ± 0.02	0.45 ± 0.02
50	0.44 ± 0.01	0.47 ± 0.02	0.44 ± 0.01
60	0.41 ± 0.04	0.44 ± 0.03	0.41 ± 0.04
70	0.47 ± 0.02	0.46 ± 0.03	0.45 ± 0.03
80	0.45 ± 0.03	0.46 ± 0.02	0.42 ± 0.01

For completely random initial parameters ($S = 0, 10$), training with noisy qubits was able to reduce $\langle \mathrm{D}(p|q)\rangle$. However, training that began at later points tend to return higher values of $\langle \mathrm{D}(p|q)\rangle$ or show minimal improvement of $\langle \mathrm{D}(p|q)\rangle$ after 10 training steps. We discuss the effects of noise and shot size on circuit training in Section IV.

TABLE VII. Trained on IBM Tokyo ($N_{shots} = 2048$). Mean value calculated over 10 independent metric evaluations.

S	$\langle \mathrm{D}(p\|q)\rangle_i$	$\langle \mathrm{D}(p\|q)\rangle_f$	min $\langle \mathrm{D}(p\|q)\rangle$
0	1.30 ± 0.06	0.34 ± 0.02	0.34 ± 0.02
10	0.58 ± 0.01	0.37 ± 0.02	0.37 ± 0.02
20	0.37 ± 0.02	0.35 ± 0.02	0.30 ± 0.02
30	0.30 ± 0.02	0.36 ± 0.02	0.30 ± 0.02
40	0.33 ± 0.01	0.35 ± 0.01	0.33 ± 0.01
50	0.38 ± 0.02	0.43 ± 0.03	0.38 ± 0.02
60	0.29 ± 0.02	0.34 ± 0.02	0.29 ± 0.02
70	0.30 ± 0.03	0.29 ± 0.02	0.29 ± 0.02
80	0.29 ± 0.02	0.30 ± 0.03	0.29 ± 0.02

IV. DISCUSSION

Effective classical machine learning relies on proper tuning of hyper-parameters and avoiding over-fitting. By limiting the number of training steps and rotational parameters our models try to fit, we believe that we have avoided circuit ansatz that are too complex for the dataset. The hyper-parameters of Adam were optimized using noiseless simulation and good rotational parameters were learned for the circuits in this paper, with the exception of the $d_C = 2, L = 1$ circuit which we will exclude from discussion in this section. In this section we will use the Kullback-Leiber divergence to discuss the qualitative changes in performance due to qubit noise and finite sampling.

A. Device noise

When simulated with noiseless qubits, increasing the number of rotational parameters improves the capabilities of the QCBM. The lowest $\langle \mathrm{D}(p|q)\rangle \sim 0.01$ values were found for $L = 2$, regardless of d_C value. This same convergence is not seen when circuits are deployed on noisy hardware. Current quantum devices have many sources of noise including: qubit decoherence, gate infidelity, and measurement errors. In this study we assume the training will be able to compensate for noise in the single qubit gates, and the limited circuit size will mitigate decoherence effects. In this initial study we have not included any readout error mitigation, and designed entangling layers to reduce the noise from 2 qubit CNOT gates. With the addition of noise the $d_C = 2, L = 2$ circuit returned the lowest value $\langle \mathrm{D}(p|q)\rangle = 0.27 \pm 0.02$ using values pre-trained via noiseless simulation. Comparable values are found when a circuit was trained on noisy qubits, the lowest value found after training was $\langle \mathrm{D}(p|q)\rangle = 0.29 \pm 0.01$ (see Tables V to VII). Understanding how training is affected by the loss function space is an active area of research for classical machine learning [19, 20]. We will use this concept to frame our discussion in this section using $\tau_U(\tau_{U'})$ for to the loss function space of a noiseless (noisy) circuit.

For a circuit with R rotational parameters, the loss function space τ is defined over the R dimensional set of all possible parameter values. We will compare the noiseless and noisy qubit performances to draw conclusions about how the addition of noise affects the space τ_U of a single circuit ansatz (c.f. Figures 5 to 7) and rely on several assumptions made without explicit models of these spaces. First, varying the value of d_C modifies the encoded degrees of entanglement. The local and global optimal parameters of circuits with different d_C, L will therefore be quite different. Also, for circuits with the same values of d_C, L noise will cause the spaces $(\tau_U, \tau_{U'})$ to differ.

In the absence of qubit noise the training has largely converged after ≈ 50 steps of training. With the weight decay implemented in Adam, this implies that the optimizer is taking small steps within a localized region of τ_U. Our first observation is trivial: just as the optima of τ_U are expected to be different for different d_C, L values; the minimum that Adam converges to in τ_U is not guaranteed to be a minimum in $\tau_{U'}$ and using Adam to optimize over τ_U instead may drive the system further from the ideal parameters for $\tau_{U'}$. However, small changes in parameters can lead to a good minimum within the space $\tau_{U'}$. Secondly, the stability of τ_U, does not necessarily predict the stability of $\tau_{U'}$. Small changes in parameters can lead to fluctuations in $\langle \mathrm{D}(p|q)\rangle$ or possible degradation on noisy qubits (c.f. Figure 7, $N_{shots} = 512$). On the other hand, the convergence in τ_U to an improved value can be seen in $\tau_{U'}$ (c.f. Figure 5, $N_{shots} = 1024$); the relative stability of the KL divergence implies that Adam is exploring a region of τ_U which is quasi-stable in $\tau_{U'}$.

Rotational parameters learned during training are dependent on hardware noise and variability. For all values of N_{shots}, training on hardware improved $\langle \mathrm{D}(p|q)\rangle$

when the circuit was initialized with a random set of parameters, or pre-trained parameters obtained from a low number of Adam steps ($S < 40$). On the other hand, continued training on hardware after the training in the simulator has already converged yields no improvement in $\langle D(p|q) \rangle$. The hardware-trained parameters overall yielded less of an improvement than trained parameters from the simulator due to the inherent noise of the quantum computer. Therefore, interleaving error mitigation steps with each training step is expected to improve performance of hardware-trained parameters, and this is the subject of a future study.

B. Sampling

In Section III we trained multiple circuits from random initial values using noiseless qubits. For each circuit, Adam trains a unique QCBM and defines a unique path in a $16(28)$-dimensional space for $L = 1(L = 2)$ circuits. Within 100 training steps the optimizer is able to find local minima, however it is not guaranteed to converge to the global optima (see Table II). The noise introduced by smaller N_{shots} values could improve exploration during training.

Sampling a circuit with a high number of shots can improve the KL metric evaluation by reducing the probability of erroneously populated states. However reducing the sampling error by increasing N_{shots} alone may not be sufficient to counteract the effects of noise on the overall performance of a given circuit.

V. CONCLUSIONS

As quantum devices become available there is a growing need for a cohesive set of benchmarks quantifying hardware performance. We have observed that while limited connectivity between qubits and noisy gates are not a significant obstacle to circuit learning, our results show that circuit ansatz design can affect generative modeling performance.

There are 6 possible CNOT gates that can be defined between pixels of the BAS(2,2) images, and the $d_C = 2, 4$ circuits show that the distribution can be modeled by placing CNOT gates between neighboring pairs of pixels. While larger image sizes require long range correlations, efficient encoding of larger datasets into hardware with fixed qubit connectivity remains an open question (see Appendix C). For the BAS(2,2) dataset, adding more CNOTs to a single qubit in each entangling layer led to minimal increases in performance on noisy qubits. When deployed on hardware, the $d_C = 2, L = 2$ circuit outperformed all other circuits.

Using a noise-robust stochastic optimizer allows us to train quantum circuits in the presence of noisy hardware. The provided metrics show the hardware's capability to reproduce desired probability distributions in the presence of both systematic and statistical noise. We also observe that measurement shot noise can minimally affect the training of a QCBM. However, classical effects such as the optimizer getting trapped in local minima are more significant.

Since the hardware is both noisy and has somewhat sparse connectivity, choosing entangling layers with sufficient sparseness to avoid excessive systematic error while still providing enough complexity to reproduce the distribution represents a trade-off that can be explored using the metrics as a guide. Evaluating the metric for a few entangling layer designs gives insight into which entanglement circuits are good at providing the complexity to represent certain distributions with low noise.

Further development of this benchmark will focus on improvements to the noise-resilience of circuit training which will lead to better estimates of the hardware's innate capabilities. Areas of development include: incorporating error mitigation [21] into circuit training to counteract the effects of measurement (readout) and gate errors, and exploring other classical optimizers to find the most robust methods for a given hardware device. The benchmark presented in this work is a useful measure of a quantum computer ability to reproduce a discrete probability distribution, and we demonstrated its utility by analyzing the performance of a superconducting quantum computer. While fully noise-robust circuit learning remains an open question, as a benchmark it shows promising avenues for future application and refinement.

Appendix A: Alternate performance metrics

In this section we use two alternate metrics to evaluate how well a circuit modeled the BAS(2,2) distribution, both are derived from the F_1 score. First we use the qBAS22 score derived in Ref. [4], then we introduce a state-based F_1 score.

An advantage to using the qBAS22 metric is that it remains finite even if a BAS state is absent from the sampled distribution. In [4] the terms of the F_1 are defined as follows: the precision is the number of counts returned in BAS states divided by the total number of counts, the recall is the number of BAS states returned divided by the total number of BAS states. The number of samples used to calculate the qBAS22 score (N_{qBAS}) is different from the number of shots (N_{shots}) used to sample from a circuit. For BAS(2,2) $N_{qBAS} = 15$, this was derived in the Appendix of [4].

We focus on circuits trained with $N_{shots} = 1024$ and calculate the $\langle$qBAS22$\rangle$ scores at each training step for the 6 circuits introduced in the main text. At each training step we sample from a given circuit 11 times each with $N_{shots} = 1024$, generating 11 independent distributions $\{q_i(x)\}_{i=1}^{11}$. For a single distribution ($q_i(x)$) we generate a set of sub-samples $\{s_j\}_{j=1}^{10,000}$ by drawing 10,000 samples of size $N_{qBAS} = 15$ (sampling done with replacement). The set $\{s_j\}$ is used to calculate $\mu(\text{qBAS22})_i \pm \sigma_i$, the qBAS22 score of $q_i(x)$. The reported $\langle$qBAS22$\rangle$ values are the weighted arithmetic mean of means taken over the 11 independent distributions: $\langle$qBAS22$\rangle = \frac{1}{11} \sum_{i=1}^{11} \frac{1}{\sigma_i^2} \mu(\text{qBAS22})_i$. The uncertainty is given by the variance of the mean.

We evaluate this metric for circuits trained with noiseless qubits and for circuits trained on hardware. In Fig. 8 we show the qBAS22 score for circuits that are trained with noiseless qubits and the metric is evaluated with noiseless qubits. The qBAS22 score for the $d_C = 2, L = 1$ circuit (which doesn't completely model the entire BAS(2,2) dataset) is the lowest performing circuit. Of the 6 circuits shown in Fig. 8 the $L = 2, d_C = 3, 4$ circuits have the highest qBAS22 scores (0.96 ± 0.04 and 0.95 ± 0.4, respectively).

However the device noise strongly affects the $d_c = 3, 4$ circuits. In Figure 9 we present $\langle$qBAS22$\rangle$ scores for circuits trained with noiseless qubits, but evaluate the metric on IBM Tokyo. We see that for $L = 1$ circuits the $d_C = 2$ circuit perform comparably to the $d_C = 3, 4$ circuits, even though this circuit is known to only fit 4 out of the 6 BAS(2,2) states. When the circuit size is increased to $L = 2$, the $d_c = 2, 3$ circuits have comparable performance after 100 steps of training (0.75 ± 0.04 and 0.75 ± 0.04, respectively), out-performing the $d_c = 4$ circuit (0.69 ± 0.04). Similar behavior is seen in the KL metric reported in the main text (c.f. Table IV).

In Table VIII we present the qBAS22 score evaluated on IBM Tokyo for the ($d_C = 2, L = 2$) circuit trained on hardware. As in Section III B the circuits are pre-trained using noiseless simulation for a fixed number of steps, then deployed on IBM Tokyo hardware to execute 10 steps of Adam training. The best performance of a circuit trained on hardware for 10 steps of Adam was $\langle$qBAS22$\rangle = 0.74 \pm 0.03$.

The qBAS22 metric and the KL metric give a measure of the global performance of a circuit but there is also a need for local metrics. We adapt a *modified* F_1 score
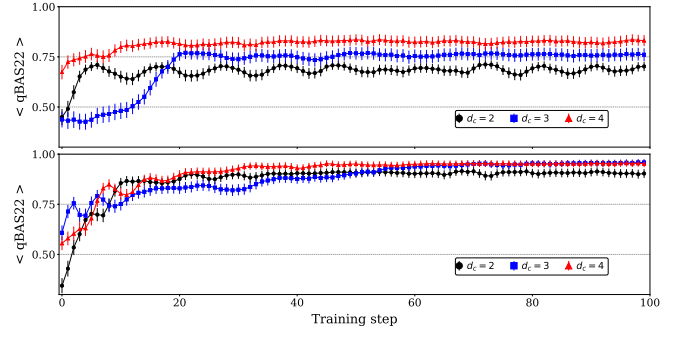

FIG. 8. The weighted mean taken over 11 independent distributions of $\langle$qBAS22$\rangle$ scores evaluated at each training step using $N_{qBAS} = 15$ and $10,000$ samples(error bars are the weighted variance). The circuit training and metric evaluation are both done on noiseless qubits: (Top) $L = 1$ circuits, (Bottom) $L = 2$ circuits.
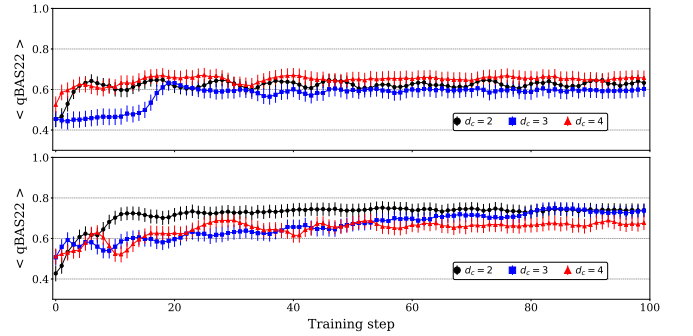

FIG. 9. The weighted mean taken over 11 independent distributions of $\langle$qBAS22$\rangle$ scores evaluated at each training step using $N_{qBAS} = 15$ and $10,000$ samples(error bars are the weighted variance). The circuit training is done on noiseless qubits and the metric is evaluated on IBM Tokyo: (Top) $L = 1$ circuits, (Bottom) $L = 2$ circuits.

[18], and apply it to the individual BAS(n,m) states to define a metric that measures how well a circuit learns each state and can be applied to uniform or non-uniform discrete distributions. However, it requires that the user specify the exact form of the target distribution. For benchmarking tasks where the performance is measured

TABLE VIII. $\langle$qBAS22$\rangle$ scores evaluated on IBM Tokyo at each training step for circuits trained on IBM Tokyo.

S	$\langle$qBAS22$\rangle_i$	$\langle$qBAS22$\rangle_f$	max $\langle$qBAS22$\rangle$
0	0.42 ± 0.04	0.74 ± 0.03	0.74 ± 0.03
10	0.71 ± 0.03	0.72 ± 0.03	0.74 ± 0.03
20	0.70 ± 0.03	0.71 ± 0.03	0.72 ± 0.03
30	0.74 ± 0.03	0.73 ± 0.03	0.74 ± 0.03
40	0.73 ± 0.03	0.69 ± 0.04	0.73 ± 0.03
50	0.73 ± 0.03	0.71 ± 0.03	0.73 ± 0.03
60	0.74 ± 0.03	0.72 ± 0.03	0.74 ± 0.03
70	0.72 ± 0.03	0.71 ± 0.03	0.72 ± 0.03
80	0.72 ± 0.03	0.71 ± 0.03	0.73 ± 0.03

with regards to a known distribution this is not a problem, but it may limit the usability of the F_1 score metric for future applications.

The modified F_1 score defines the precision and true positive rate of a model with respect to the uniform BAS(2,2) distribution ($p_i = 1/6$ if $|x_i\rangle$ is a BAS(2,2) state). Device noise (such as readout errors) leads to a number of incorrectly measured states, but in our initial approximation, for each state $|x_i\rangle$ of the BAS dataset we define the number of true positives as: $TP(x_i) = q(x_i)$ if $q(x_i) < p(x_i)$ and $TP(x_i) = p(x_i)$ if $q(x_i) > p(x_i)$. We define the number of false positives (FP) and false negatives (FN) using the difference $\Delta = |q(x_i) - p(x_i)|$. If $q(x_i) > p(x_i)$ then $FP(x_i) = \Delta$ and $FN(x_i) = 0$; if $q(x_i) < p(x_i)$ then $FN(x_i) = \Delta$ and $FP(x_i) = 0$. With these values we can define for each state x_i: the true positive rate

$$\text{TPR}(x_i) = \frac{\text{TP}(x_i)}{[\text{TP}(x_i) + \text{FN}(x_i)]}, \tag{A1}$$

and the precision

$$\text{P}(x_i) = \frac{\text{TP}(x_i)}{[\text{TP}(x_i) + \text{FP}(x_i)]}. \tag{A2}$$

The modified F_1 score is defined in terms of the precision and true positive rate,

$$F_1(x_i) = 2\left(\frac{\text{P}(x_i) \times \text{TPR}(x_i)}{\text{P}(x_i) + \text{TPR}(x_i)}\right). \tag{A3}$$

Appendix B: Alternate learning rates

In this section we highlight the specific case of the $(d_C = 2, L = 1)$ circuit. In Figure 3 the KL value oscillated around $D(p|q) \sim 1.1$ and in Sections III and IV we argue that this behavior is due to the circuit being overly simplistic and not from a too-large learning rate.

To prove this we randomly initialize the $(d_C = 2, L = 1)$ circuit, and train it with: $N_{shots} = 1024$, different learning rates $\alpha = \{0.05, 0.3\}$ and train for 200 steps of Adam. In Figures 10 and 11 we show the state-by-state F_1 score (evaluated with $N_{shots} = 2048$). For either learning rate ($\alpha = 0.05, 0.3$) the $(d_C = 2, L = 1)$ circuit fails to learn the states $|1010\rangle$ and $|0101\rangle$. In contrast, $(d_C = 2, L = 2)$ circuit is able to learn all 6 BAS states.

Appendix C: Connectivity, correlation locality, and hardware embedding

We define local or non-local connections with respect to the image pixels of the BAS(2,2) dataset. There are 6 possible pairs that can be formed from the four pixels of each image (4 local, 2 non-local). The nearest neighbor pairs of pixels $[(0,1),(0,2),(1,3),(2,3)]$ form the local
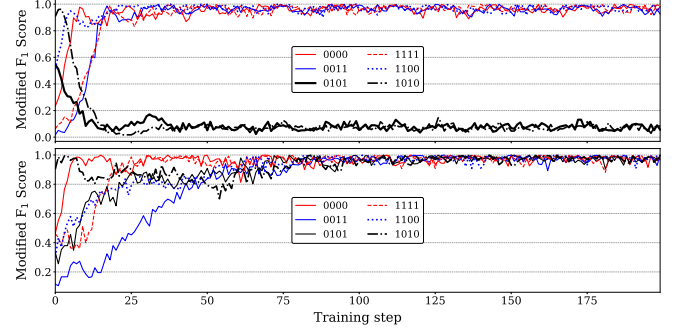

FIG. 10. F_1 score per BAS state of a circuit trained on a noiseless simulator with: 200 steps of Adam, $\alpha = 0.05$. (Top) $d_C = 2, L = 1$, (Bottom) $d_C = 2, L = 2$.
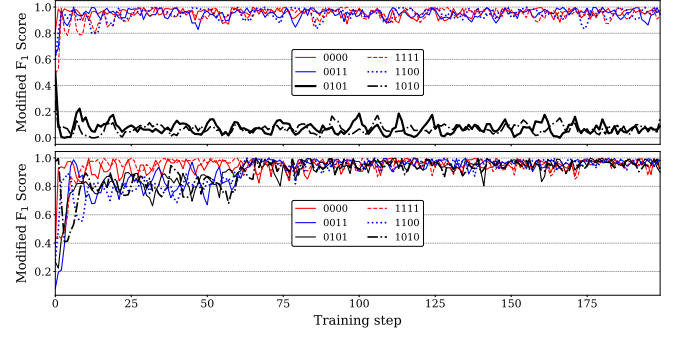

FIG. 11. F_1 score per BAS state of a circuit trained on a noiseless simulator with: 200 steps of Adam, $\alpha = 0.3$. (Top) $d_C = 2, L = 1$, (Bottom) $d_C = 2, L = 2$.

connections, while the remaining pairs $[(0,3),(1,2)]$ are non-local.

If the hardware supports all-to-all connectivity then all local and non-local connections can be mapped to CNOT gates and implemented in a single QCBM. With limited qubit connectivity, it is possible to embed to non-local connections into hardware but often at the cost of removing local connections. The $d_C = 2, 4$ layers construct QCBMs with 4 local connections and 0 non-local con-
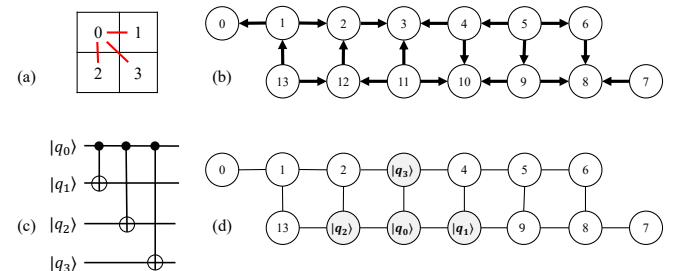

FIG. 12. (a) The pixels of a BAS(2,2) image with the edges of the Chow-Liu tree defined from the mutual information (red). (b) The connectivity graph of IBM's Melbourne chip [13]. (c) The $d_C = 3$ entangling layer defined using the Chow-Liu tree in (a). (d) The $d_C = 3$ layer embedded into IBM Melbourne.

nections, whereas the $d_C = 3$ layers construct QCBMs with 1 non-local and 2 local connections. In Figure 12 we show the construction and hardware embedding of a $d_C = 3$ entangling layer from the edges of a Chow-Liu tree rooted at pixel 0. Understanding the trade-offs between local or non-local connections will be necessary to construct QCBMs that can model larger images or more complicated distributions.

[1] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. Nature, 549(7671):195–202, sep 2017.

[2] Alejandro Perdomo-Ortiz, Marcello Benedetti, John Realpe-Gmez, and Rupak Biswas. Opportunities and challenges for quantum-assisted machine learning in near-term quantum computers. Quantum Science and Technology, 3(3):030502, 2018.

[3] Jin-Guo Liu and Lei Wang. Differentiable learning of quantum circuit Born machine. arXiv preprint arXiv:1804.04168, 2018.

[4] Marcello Benedetti, Delfina Garcia-Pintos, Yunseong Nam, and Alejandro Perdomo-Ortiz. A generative modeling approach for benchmarking and training shallow quantum circuits. arXiv preprint arXiv:1801.07686, 2018.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in neural information processing systems, pages 2672–2680, 2014.

[6] Seth Lloyd and Christian Weedbrook. Quantum generative adversarial learning. arXiv preprint arXiv:1804.09139, 2018.

[7] Pierre-Luc Dallaire-Demers and Nathan Killoran. Quantum generative adversarial networks. arXiv preprint arXiv:1804.08641, 2018.

[8] Marcello Benedetti, Edward Grant, Leonard Wossnig, and Simone Severini. Adversarial quantum circuit learning for pure state approximation. arXiv preprint arXiv:1806.00463, 2018.

[9] Ling Hu, Shu-Hao Wu, Weizhou Cai, Yuwei Ma, Xianghao Mu, Yuan Xu, Haiyan Wang, Yipu Song, Dong-Ling Deng, Chang-Ling Zou, et al. Quantum generative adversarial learning in a superconducting quantum circuit. arXiv preprint arXiv:1808.02893, 2018.

[10] Jinfeng Zeng, Yufeng Wu, Jin-Guo Liu, Lei Wang, and Jiangping Hu. Learning and inference on generative adversarial quantum circuits. arXiv preprint arXiv:1808.03425, 2018.

[11] Song Cheng, Jing Chen, and Lei Wang. Information perspective to probabilistic modeling: Boltzmann machines versus Born machines. Entropy, 20(8):583, 2018.

[12] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M Chow, and Jay M Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. Nature, 549(7671):242, 2017.

[13] 16 qubit backend: IBM Q team. IBM Q 16 Melbourne backend specification v1.1.0, 2018. Retrieved from https://ibm.biz/qiskit-melbourne.

[14] Kristel Michielsen, Madita Nocon, Dennis Willsch, Fengping Jin, Thomas Lippert, and Hans De Raedt. Benchmarking gate-based quantum computers. Computer Physics Communications, 220:44–55, 2017.

[15] Shakir Mohamed and Balaji Lakshminarayanan. Learning in implicit generative models. arXiv preprint arXiv:1610.03483, 2016.

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[17] Arthur Gretton, Karsten M Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J Smola. A kernel method for the two-sample-problem. In Advances in neural information processing systems, pages 513–520, 2007.

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.

[19] Hao Li, Zheng Xu, Gavin Taylor, and Tom Goldstein. Visualizing the loss landscape of neural nets. arXiv preprint arXiv:1712.09913, 2017.

[20] Pratik Chaudhari and Stefano Soatto. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In 2018 Information Theory and Applications Workshop (ITA), pages 1–10. IEEE, 2018.

[21] Abhinav Kandala, Kristan Temme, Antonio D. Crcoles, Antonio Mezzacapo, Jerry M. Chow, and Jay M. Gambetta. Error mitigation extends the computational reach of a noisy quantum processor. Nature, 567(7749):491, March 2019.