Quantum-assisted Helmholtz machines: A quantum-classical deep learning framework for industrial datasets in near-term devices

Marcello Benedetti,[1,2,3] John Realpe-Gómez,[1,4,5] and Alejandro Perdomo-Ortiz[1,2,3,*]

[1] *Quantum Artificial Intelligence Lab., NASA Ames Research Center, Moffett Field, CA 94035, USA*
[2] *USRA Research Institute for Advanced Computer Science (RIACS), Mountain View CA 94043, USA*
[3] *Department of Computer Science, University College London, WC1E 6BT London, UK*
[4] *SGT Inc., Greenbelt, MD 20770, USA*
[5] *Instituto de Matemáticas Aplicadas, Universidad de Cartagena, Bolívar 130001, Colombia*

Machine learning has been presented as one of the key applications for near-term quantum technologies, given its high commercial value and wide range of applicability. In this work, we introduce the *quantum-assisted Helmholtz machine:* a hybrid quantum-classical framework with the potential of tackling high-dimensional real-world machine learning datasets on continuous variables. Instead of using quantum computers to only assist deep learning, as previous approaches have suggested, we use deep learning to extract a low-dimensional binary representation of data, suitable for relatively small quantum processors which can assist the training of an unsupervised generative model. To demonstrate this concept on a real-world dataset, we used 1644 quantum bits of a noisy non-fault-tolerant quantum device, the D-Wave 2000Q, to assist the training of a sub-sampled version of the MNIST handwritten digit dataset with 16×16 continuous valued pixels. Although we illustrate this concept on a quantum annealer, adaptations to other quantum platforms, such as ion-trap technologies or superconducting gate-model architectures, could be explored within this flexible framework.

I. INTRODUCTION

There as been much interest in quantum algorithms for enhancing deep learning and other machine learning (ML) algorithms [1–27]. In this article, instead, we argue that deep learning and quantum devices can help each other to achieve hard tasks such as generative modeling. The resulting quantum-assisted ML (QAML) approach is much more suitable for implementation in near-term quantum hardware and can be used in real applications as well. Indeed, previous work has shown experimental evidence of the ability of quantum annealers to perform useful and realistic ML tasks, such as implementing generative models of small binarized datasets [13, 14, 16, 17, 22]. A natural extension is to develop techniques to handle large datasets—where variables could be discrete, continuous, or more general objects—and to include latent variables to increase the modelling capacity of the quantum-assisted architectures. Clearly, this would open up the possibility to use QAML in real-world domains and benchmark it against extensively studied classical approaches. This extension is the focus of this work.

The interest in generative models stem from their generality. Deep generative models with many layers of hidden stochastic variables have the ability to learn multimodal distributions over high-dimensional datasets [28]. Each additional layer provides an increasingly abstract representation of the data and improves the generalization capability of the model [29]. Furthermore, generative models apply to unlabeled data, which accounts for most of the public data in the Internet and the private data within a company. The price to pay for using generative models is the intractability of inference, training, and model selection. Generative models are trained in an unsupervised fashion, often relying on variational approximations and computationally costly Markov Chain Monte Carlo (MCMC) sampling. This is where we think quantum computation can have a significant impact. First, under the hypothesis that quantum computers allows for more efficient sampling, we can run the expensive subroutine on quantum hardware. Second, by exploiting the non-trivial graph topologies in quantum hardware, we can implement complex network that are usually avoided in favor of restricted ones (e.g. bipartite graphs are favored in classical neural networks for convenience).

Quantum information does not have to be encoded into binary observables (qubits), it could also be encoded into continuous observables [30]. Some researchers have followed the latter direction [31, 32]. However, most available quantum computers do work with qubits, nicely resembling the world of classical computation. Datasets commonly found in industrial applications have a large number of variables that are not binary. For instance, datasets of images with millions of pixels which can be in gray scale, with 256 intensities per pixel, or in color, represented by 3-dimensional vectors. We refer to this kind of datasets as *complex ML datasets*. A naive binarization of the data will quickly consume the qubits of any device with 100-1000 qubits. Several QAML algorithms [4, 7, 11] rely on amplitude encoding instead, a technique where continuous data vectors are stored into the amplitudes of a quantum state. This provides an exponentially efficient representation upon which one could perform linear algebra operations. Unfortunately, it is not clear how to prepare arbitrary states of this kind in

* Correspondence: alejandro.perdomoortiz@nasa.gov

near-term quantum systems. Even reading out all the amplitudes of an output vector might kill or significantly hamper any speedup claims [15].

Here we suggest using a quantum device to model an abstract representation of the data, that is, the deepest layers of a deep learning architecture. The number of hidden variables in the deepest layers of a network can indeed be much smaller than the number of visible variables, which is ideal for near-term implementation on early quantum technologies; either quantum annealers or gate-based quantum computers. Such a low-dimensional compact representation is often stochastic and binary in generative modelling [33]. We expect quantum devices to have a higher impact at this abstract representation, where the classically-tractable information has been already trimmed by the classical deep learning architecture. The lower layers of the network are classical components that effectively transforms samples from the quantum device to samples with the same structure of those in the dataset, and vice-versa. Hence, visible variables could be continuous variables, discrete variables, or other objects, effectively solving the encoding problem. (In Appendix A we argue why a direct implementation of stochastic continuous variables in hardware would be challenging even for the most trivial cases.) Moreover, because the quantum device works on a lower dimensional binary representation of the data, we are also able to handle datasets whose dimensionality is much larger than it would be possible with state-of-the-art hardware.

The structure of the article is as follows: In Sec. II we describe some of the architectures that can be handled within the approach we introduce in this work. In Sec. III we formally define the Quantum-Assisted Helmholtz Machine (QAHM) and derive the corresponding quantum assisted wake-sleep learning algorithm. In Sec. IV we describe some experimental results on the quantum-assisted generation of gray-scale handwritten digits of the MNIST dataset. In Sec. V we present the conclusions of this work and suggest future work.

II. QUANTUM-ASSISTED ARCHITECTURES

A deep generative model is based on a probability distribution $P(\mathbf{v}) = \sum_{\mathbf{u}} P(\mathbf{v}|\mathbf{u})P(\mathbf{u})$, where $\mathbf{v} = \{v_1, \ldots, v_N\}$ are visible units encoding the data and $\mathbf{u} = \{u_1, \ldots, u_M\}$ are unobservable or hidden units that serve to capture further non-trivial correlations. To perform inference and learning on this model we have to sample from the posterior distribution $P(\mathbf{u}|\mathbf{v})$, which is generally intractable, specially in models with several layers of hidden units. In Fig. 1, we show some of the deep architectures that could work in synergy with quantum devices. Generative models are often represented as graphs of stochastic nodes where edges may be directed, undirected, or both. Here, we color in blue the nodes that can be implemented on a quantum device and we use an edge marked at both ends to indicate a quantum interaction.

Fig. 1 (a) shows an instance of a Helmholtz machine [34–36], which consists of two networks: a *recognition network* to do approximate inference on hidden units using information extracted from real data, and a *generation network* to generate artificial data. The recognition network implements a distribution $Q(\mathbf{u}|\mathbf{v})$ that approximates the intractable distribution $P(\mathbf{u}|\mathbf{v})$, and is used to do bottom-up sampling starting from any visible vector $\mathbf{v}$; this network may be entirely classical or quantum-assisted as discussed in Sec. III. The generator network, instead, can be used to perform top-down sampling, starting at the deepest hidden layer (e.g. $\mathbf{u}^{(2)}$ in Fig. 1) of a quantum model and propagating the samples down to reach the visible layer. If the inference and generation networks shared the same quantum layer, we obtain the quantum-assisted version of a deep belief network [29, 33](QADBN; see Fig. 1 (b)). Deep belief networks usually implement a bipartite undirected graph in the deepest layer, but here we schematically show a more general structure with lateral connections that could be implemented in quantum hardware. Finally, if the recognition network is the exact inverse of the generation network, we obtain a quantum-assisted Boltzmann machine [37, 38] (QABM; see Fig. 1 (c)).

Nevertheless, these alternatives to QAHM pose additional challenges for the implementation in near-term devices, as discussed in detail in Ref. [14]. For this reason, we choose to work in the more flexible framework of the QAHM to be discussed next.

III. MODEL DEFINITION AND LEARNING ALGORITHM

Consider a data set $\mathcal{S} = \{\mathbf{v}^1, \ldots, \mathbf{v}^d\}$ with empirical distribution $Q_{\mathcal{S}}(\mathbf{v})$, to be modeled with a generative model $P(\mathbf{v}) = \sum_{\mathbf{u}} P(\mathbf{u}, \mathbf{v})$, where $P(\mathbf{u}, \mathbf{v}) = P(\mathbf{v}|\mathbf{u})P_{QC}(\mathbf{u})$. Here $\mathbf{v} = (v_1, \ldots, v_N)$ are the visible units that represent the data, while $\mathbf{u} = (u_1, \ldots, u_M)$ are unobserved or hidden units that serve to capture non-trivial correlations by encoding high-level features. The distribution $P_{QC}(\mathbf{u}) = \langle \mathbf{u}|\hat{\rho}|\mathbf{u}\rangle$ describes samples obtained from a quantum device which could be described, for example, by the diagonal elements of a quantum Gibbs distribution $\hat{\rho} = e^{-\beta\mathcal{H}}/\mathcal{Z}$. Here $\mathcal{H}$ is the corresponding Hamiltonian implemented in quantum hardware and $\mathcal{Z}$ is the partition function. For instance, in the case of the D-Wave-2000Q (DW2000Q) we have

$$\mathcal{H} = \sum_{i,j} J_{ij}\hat{Z}_i\hat{Z}_j + \sum_i h_i\hat{Z}_i + \Gamma\sum_i \hat{X}_i, \qquad (1)$$

where $\hat{X}_i$ and $\hat{Z}_i$ denote Pauli matrices in the x and z direction, respectively.

The conditional distribution $P(\mathbf{v}|\mathbf{u})$ stochastically translates samples from the quantum computer into sam-

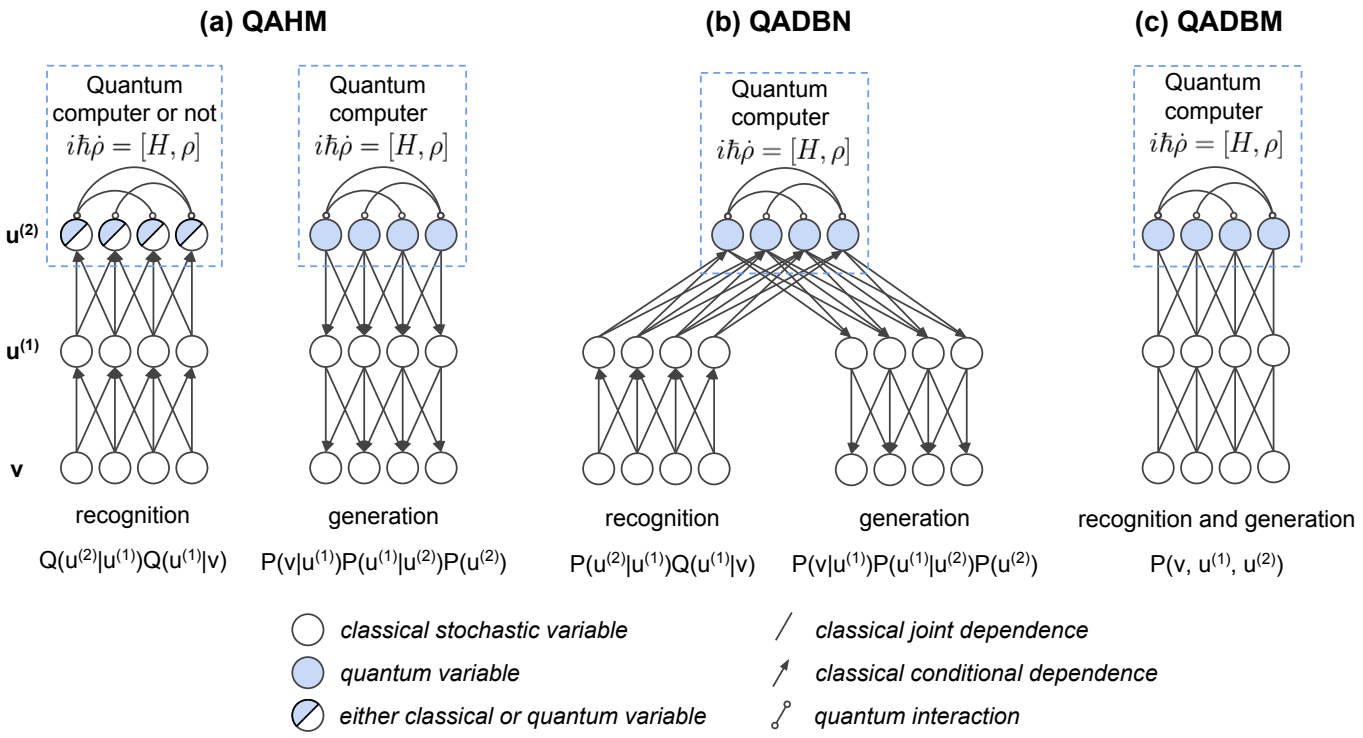

FIG. 1. Architectures for quantum-assisted machine learning (QAML). (a) Quantum-Assisted Helmholtz Machine (QAHM); (b) Quantum-Assisted Deep Belief Network (QADBN); (c) Quantum-Assisted Deep Boltzmann Machine (QADBM). We refer the reader to Sec. II for a brief description of the proposals pictured here.

ples on the domain of the data. That is, $\mathbf{v}$ could be a visible vector of continuous variables, binary variables, or other type of objects. This is a significant advantage over other quantum-assisted approaches where the visible units are directly represented with the available qubits.

Ideally, an unsupervised learning algorithm would maximize the average log-likelihood of the data

$$\mathcal{L} = \sum_{\mathbf{v}} Q_{\mathcal{S}}(\mathbf{v}) \ln P(\mathbf{v}). \tag{2}$$

The training of a Helmholtz machine is based on the inequality

$$\ln P(\mathbf{v}) \geq \sum_{\mathbf{u}} Q(\mathbf{u}|\mathbf{v}) \ln \frac{P(\mathbf{u}, \mathbf{v})}{Q(\mathbf{u}|\mathbf{v})}, \tag{3}$$

where $Q(\mathbf{u}|\mathbf{v})$ is an auxiliary recognition network that approximates the true intractable posterior $P(\mathbf{u}|\mathbf{v})$. Our hybrid architecture uses a classical neural network for Q, sidestepping the need to run a quantum device for each data point and for each iteration of learning; a bottleneck that is intrinsic in all the proposals we know up to date that treat quantum annealers as Boltzmann machines on the hidden layers of a neural network (e.g. see Ref. [39] for one recent such proposals).

From now on, we focus on the case of quantum Gibbs distributions. The term $\ln\langle\mathbf{u}|\hat{\rho}|\mathbf{u}\rangle$ arising from $\ln P(\mathbf{u}, \mathbf{v})$ in Eq. (3) is intractable due to the projection of the Gibbs distribution on the states $|\mathbf{u}\rangle$. A bound for this term was derived in Ref. [18] based on the Golden-Thompson inequality, which requires taking the logarithm of the singular matrix $|\mathbf{u}\rangle\langle\mathbf{u}|$. Here we derive the same result in a slightly different way. We expand $\hat{\rho}$ in terms of the eigenvectors $|n\rangle$ and eigenvalues E_n of the Hamiltonian and make use of the log-sum inequality [40]

$$\sum_i a_i \ln \frac{\sum_j b_j}{\sum_k a_k} \geq \sum_i a_i \ln \frac{b_i}{a_i}, \tag{4}$$

valid for arbitrary non-negative numbers a_i and b_i. With $a_i = |\langle n|\mathbf{u}\rangle|^2$ and $b_i = a_i e^{-\beta E_i}/\mathcal{Z}$, we obtain

$$\ln\langle\mathbf{u}|\rho|\mathbf{u}\rangle \geq \langle\mathbf{u}|\ln\rho|\mathbf{u}\rangle. \tag{5}$$

Combining Eqs. (3) and (5), we get a tractable lower bound for $\ln P(\mathbf{v})$. Instead of maximizing $\mathcal{L}$ in Eq. (2), we can maximize the data average of the resulting lower bound, i.e. the function

$$\mathcal{G}(\theta_G, \theta_{QC}) = \sum_{\mathbf{u},\mathbf{v}} Q_{\mathcal{S}}(\mathbf{v}) Q(\mathbf{u}|\mathbf{v}) \left[\ln P(\mathbf{v}|\mathbf{u}) + \langle\mathbf{u}|\ln\hat{\rho}|\mathbf{u}\rangle\right], \tag{6}$$

where θ_G and θ_{QC} denote the parameters determining the generator network $P(\mathbf{v}|\mathbf{u})$ and the quantum state $\hat{\rho}$, respectively. In Eq. (6) we have neglected terms that do *not* depend on either θ_G or θ_{QC}, as they vanish when computing the gradient of $\mathcal{G}$.

For a successful inference, the recognition network $Q(\mathbf{u}|\mathbf{v})$ has to track closely the true posterior during learning. It is easy to see that the bound in Eq. 3 is tight for $Q(\mathbf{u}|\mathbf{v}) = P(\mathbf{u}|\mathbf{v})$. Unfortunately, the optimization of the lower bound in Eq. 3 with respect to the parameters of the recognition network $Q(\mathbf{u}|\mathbf{v})$ is often intractable. The original wake-sleep algorithm proposed in Ref. [34] attempts to bring $Q(\mathbf{u}|\mathbf{v})$ closer to the true posterior $P(\mathbf{u}|\mathbf{v})$ by minimizing a more tractable notion of distance. Such distance is the Kullback-Leibler divergence

$$D\left[P(\mathbf{u}|\mathbf{v})||Q(\mathbf{u}|\mathbf{v})\right] = \sum_{\mathbf{u}} P(\mathbf{u}|\mathbf{v}) \ln \frac{P(\mathbf{u}|\mathbf{v})}{Q(\mathbf{u}|\mathbf{v})}, \quad (7)$$

or rather the average over the marginal $P(\mathbf{v})$ to take into account the relevance of each configuration $\mathbf{v}$. In other words, we have to maximize the function

$$\mathcal{R}(\theta_R) = \sum_{\mathbf{u},\mathbf{v}} P(\mathbf{u},\mathbf{v}) \ln Q(\mathbf{u}|\mathbf{v}), \quad (8)$$

where θ_R denotes, collectively, the parameters of the distribution $Q(\mathbf{u}|\mathbf{v})$. In Eq. (8) we neglected terms that do not depend on θ_R, as they vanish when computing the gradient of $\mathcal{R}$. The gradient ascent equations have the structure $\theta^{(t+1)} = \theta^{(t)} + \eta \nabla_\theta \mathcal{F}$, where θ stands for the parameters being updated, $\mathcal{F}$ stands for either $\mathcal{G}$ or $\mathcal{R}$, accordingly, and η is the learning rate.

Since $\ln \hat{\rho} = -\beta \mathcal{H} - \ln \mathcal{Z}$, for the parameters of the quantum distribution $\theta_{QC} = (J_{ij}, h_i)$ we have

$$-\frac{1}{\beta}\frac{\partial \mathcal{G}}{\partial J_{ij}} = \langle u_i u_j \rangle_Q - \langle u_i u_j \rangle_\rho, \quad (9)$$

$$-\frac{1}{\beta}\frac{\partial \mathcal{G}}{\partial h_i} = \langle u_i \rangle_Q - \langle u_i \rangle_\rho, \quad (10)$$

where $\langle\,\rangle_Q$ and $\langle\,\rangle_\rho$ denote expectation values with respect to $Q(\mathbf{u}|\mathbf{v})Q_{\mathcal{S}}(\mathbf{v})$ and $P_{QC}(\mathbf{u}) = \langle \mathbf{u}|\hat{\rho}|\mathbf{u}\rangle$, respectively. Here we have used the property $\hat{Z}_i|u_i\rangle = u_i|u_i\rangle$.

The generator and recognition networks can be written as deep learning architectures

$$P(\mathbf{v}|\mathbf{u}) = \sum_{\mathbf{u}^1,\ldots,\mathbf{u}^L} P_0(\mathbf{v}|\mathbf{u}^1) P_1(\mathbf{u}^1|\mathbf{u}^2) \cdots P_L(\mathbf{u}^L|\mathbf{u}), \quad (11)$$

$$Q(\mathbf{u}|\mathbf{v}) = \sum_{\mathbf{u}^1,\ldots,\mathbf{u}^L} Q_L(\mathbf{u}|\mathbf{u}^L) \cdots Q_1(\mathbf{u}^2|\mathbf{u}^1) Q_0(\mathbf{u}^1|\mathbf{v}), \quad (12)$$

in terms of L additional sets of hidden variables $\mathbf{u}^1, \ldots, \mathbf{u}^L$ that connect the variables $\mathbf{v} \equiv \mathbf{u}^0$ in the visible layer with those in the last hidden layer $\mathbf{u} \equiv \mathbf{u}^{L+1}$. More specifically, when using Bernoulli variables $u_i^\ell \in \{-1, +1\}$, we have

$$P_\ell(\mathbf{u}^\ell|\mathbf{u}^{\ell+1}) = \prod_i \sigma(u_i^\ell|\mathbf{u}^{\ell+1}; A^\ell, a^\ell) \quad (13)$$

$$Q_\ell(\mathbf{u}^\ell|\mathbf{u}^{\ell-1}) = \prod_i \sigma(u_i^\ell|\mathbf{u}^{\ell-1}; B^\ell, b^\ell), \quad (14)$$

where

$$\sigma(u_i|\mathbf{u}'; C, c) = \left[1 + e^{-2u_i\left(\sum_j C_{ij} u'_j + c_i\right)}\right]^{-1}. \quad (15)$$

The gradients for the generative network are

$$\frac{\partial \mathcal{G}}{\partial A_{ij}^\ell} = \langle u_i^\ell u_j^{\ell+1} \rangle_Q - \langle u_i^\ell \rangle_P \langle u_j^{\ell+1} \rangle_Q, \quad (16)$$

$$\frac{\partial \mathcal{G}}{\partial a_i^\ell} = \langle u_i^\ell \rangle_Q - \langle u_i^\ell \rangle_P, \quad (17)$$

and similarly for the recognition network

$$\frac{\partial \mathcal{R}}{\partial B_{ij}^\ell} = \langle u_i^\ell u_j^{\ell-1} \rangle_P - \langle u_i^\ell \rangle_Q \langle u_j^{\ell-1} \rangle_P, \quad (18)$$

$$\frac{\partial \mathcal{R}}{\partial b_i^\ell} = \langle u_i^\ell \rangle_P - \langle u_i^\ell \rangle_Q. \quad (19)$$

We now discuss some alternatives and improvements that can be found in the literature of deep generative models. A generalization of the wake-sleep algorithm, called reweighted wake-sleep algorithm, was introduced in Ref.[41]. The authors used Q as a candidate distribution for an importance sampler which provides an improved gradient estimator, as compared to those estimated here, by reducing its bias and variance. A different improvement was proposed in Ref. [42] in the context of deep Boltzmann machines. Samples from the recognition network Q were used as starting points for a set of mean-field equations, whose solutions allow to do approximated inference. Finally, there exists a contrastive version of the wake-sleep algorithm that was introduced in Ref. [29] to train deep belief networks with undirected edges between the two deepest layers. In this contrastive wake-sleep algorithm, samples from the generator network are obtained by seeding the deepest hidden layer with configurations obtained from the recognition network.

All the improved techniques discussed above require full knowledge of the parameters. This may not be available in noisy quantum annealers or quantum devices without error correction. Nevertheless, we now show how the vanilla wake-sleep algorithm can be used to train Helmholtz machines assisted by noisy quantum annealers. Disadvantages and potential solutions are discussed in Sec. V.

IV. EXPERIMENTS

We demonstrate the QAHM framework using a DW2000Q quantum annealer. The annealer implements a noisy version of the programmed Hamiltonian in Eq. (1), but non-trivial non-equilibrium effects may make samples deviate from the corresponding Gibbs distribution. This scenario requires some engineering of the

QAHM framework as well as additional actions besides those outlined in Sec. III. Following the work in Ref. [14], we use a gray-box model for the quantum annealer so that we can update its parameters without the need to estimate deviations from the Gibbs distribution. This work relies on the assumption that, despite the deviations, the estimated gradients have a positive projection on the correct direction. We would like to stress that the same algorithm can be carried out on other quantum annealer architectures [43, 44] or on noisy non-fault-tolerant gate-based quantum computers expected to appear in the near-term. Implementations in these architectures may require further, or fewer, engineering steps and will be the focus of future work.

Here, we implement a prior $P_{QC}(\mathbf{u})$ embedded in hardware with effective all-to-all connectivity over the hidden variables of the deepest layer, following the approach in Ref. [14]. That is, each logical variable of the last layer is mapped into a subgraph of qubits in hardware, such that additional physical interactions between qubits can effectively encode long-range interactions. The dynamics are described by a new Hamiltonian that is annealed into

$$\widetilde{\mathcal{H}}_P = -\frac{1}{2}\sum_{i,j=1}^{N}\sum_{k,l=1}^{Q_i,Q_j} J_{ij}^{(kl)}\hat{Z}_i^{(k)}\hat{Z}_j^{(l)} - \sum_{i=1}^{N}\sum_{k=1}^{Q_i} h_i^{(k)}\hat{Z}_i^{(k)}. \tag{20}$$

Here N is the number of hidden variables in the deepest layer, which equals the number of subgraphs realized in hardware, Q_i is the number of qubits in subgraph i, $\hat{Z}_i^{(k)}$ is the Pauli matrix in the z-direction for qubit k of subgraph i, $h_i^{(k)}$ is the local field for qubit k of subgraph i, and $J_{ij}^{(kl)}$ is the coupling between qubit k of subgraph i and qubit l of subgraph j.

Standard heuristic embedding techniques can be used to perform this expansion and obtain the desired effective connectivity. The gradients to update the control parameters of the quantum annealer are similar to those in Eqs. (9) and (10); further details can be found in Ref. [14].

Because of a varying unknown inverse temperature β, the learning rate at which parameters are updated varies too. This should not pose a problem as long as we schedule the learning rate to decrease, which is a general condition for convergence of stochastic approximation algorithms of Robbins-Monro type [45].

The model is equipped with two deterministic functions that map samples back and forth between the two spaces (i.e. logical and qubit spaces). We use the following *replica* and *majority vote* mappings

$$z_i^{(k)} = f(\mathbf{u}, i) = u_i, \quad \text{for} \quad k = 1, \ldots, Q_i, \tag{21}$$

$$u_i = g(\mathbf{z}, i) = \text{sign}\left(\sum_{k=1}^{Q_i} z_i^{(k)}\right). \tag{22}$$

These mappings can be thought of as non-trainable edges in the recognition and generator networks respectively. To see why, consider a simple QAHM with one visible $\mathbf{v}$ and two hidden layers $\mathbf{u}^1$ and $\mathbf{u}^2$, like the one shown in Figure 2. In the recognition network, the hidden units $\mathbf{u}^2$ in the second layer are replicated into higher-dimensional vectors $\mathbf{z}$ (replicas are shown with the same color). The number of replicas Q_i for each unit i is dictated by a heuristic embedding that runs on the underlying quantum hardware. Notice that we can easily sample from the recognition network using a bottom-up pass that does not involve the quantum device. In the generator network instead, the quantum device is used to sample from a Gibbs-like distribution over the high-dimensional space of qubits. Samples are mapped back to the hidden units $\mathbf{u}^2$ using the majority vote over subgraphs (subgraphs are shown with the same color). Then, a top-down pass is used to sample the visible units $\mathbf{v}$. Notice that every directed and undirected edge in Figure 2 can be trained, except for the gray-colored directed edges corresponding to the fixed mappings in Eqs. (21) and (22). In future work we will consider extending the model by including a quantum device in the deepest layer of the recognition network. This will require to sample from the device conditionally on each datapoint. On the other hand, the QAHM designed this way can make use of *all* the qubits even those that are not part of the embedding; that is, qubits not belonging to any subgraph can be treated as additional hidden units and marginalized out.

Now, because we don't have complete knowledge of the parameters implemented by the annealer, we cannot use techniques such as importance sampling and mean-field equations that have been used to improve the wake-sleep algorithm and obtain state-of-the-art results (see Section III for a brief summary). We shall stress that this limitation is peculiar of our case-study and may not be present in other quantum hardware (e.g. error-corrected quantum computers). Improved and faster learning can also be obtained by initializing the approximate posterior $Q(\mathbf{u}|\mathbf{v})$ close to true posterior $P(\mathbf{u}|\mathbf{v})$ when $\mathbf{v}$ is sampled from the dataset. This initialization, also called *pre-training*, is often carried out by stacking layers of restricted Boltzmann machines and training them greedily with some fast approximate algorithm [29, 42]. In principle, we could use pre-training to initialize all the trainable directed edges of our model (see Figure 2). The procedure would trivially extend to the undirected edges in the generator network because the pre-trained recognition network would effectively provide a fully-observed dataset for computing the gradients in Eqs. (9) and (10). We decided not to carry out pre-training in our small scale experiments as it could initialize the model to an almost-optimal configuration, hence hiding any contribution of the quantum device. For the reasons outlined above, we acknowledge that our vanilla wake-sleep algorithm may be slow and sub-optimal (this is further dis-

Algorithm 1 Wake-sleep algorithm for quantum assisted Helmholtz machines on quantum annealers

use an heuristic to embed a fully-connected graph in hardware corresponding to the deepest layer of hidden units
define mappings $f(\mathbf{u}, i)$ and $g(\mathbf{z}, i)$ from hidden variables to qubits and back, accordingly
for number of training epochs **do**
  sample $(\mathbf{v}^d, \mathbf{u}^d, \mathbf{z}^d)$ where $(\mathbf{v}^d, \mathbf{u}^d) \sim Q(\mathbf{u}|\mathbf{v})Q_{\mathcal{S}}(\mathbf{v})$ and $z_i^d = f(\mathbf{u}^d, i)$
  sample $(\mathbf{v}^k, \mathbf{u}^k, \mathbf{z}^k)$ where $\mathbf{z}^k \sim \langle \mathbf{z}|\hat{\rho}|\mathbf{z}\rangle$, $u_i^k = g(\mathbf{z}^k, i)$ and $\mathbf{v}^k \sim P(\mathbf{v}|\mathbf{u}^k)$
  estimate $\nabla_\theta \mathcal{G}$ and $\nabla_\theta \mathcal{R}$ from samples
  update $\theta_{\mathcal{G}}^{(t+1)} = \theta_{\mathcal{G}}^{(t)} + \eta \nabla_\theta \mathcal{G}$
  update $\theta_{\mathcal{R}}^{(t+1)} = \theta_{\mathcal{R}}^{(t)} + \eta \nabla_\theta \mathcal{R}$
  decrease η
end for

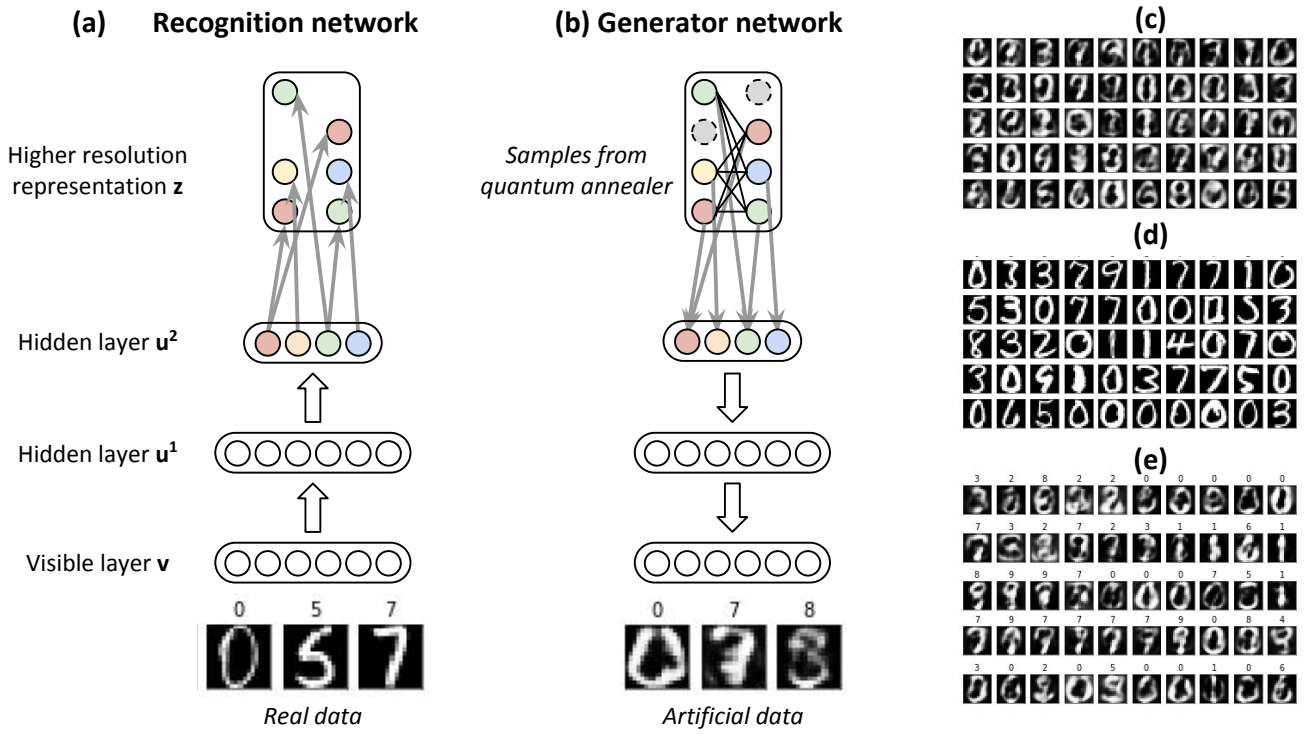

FIG. 2. Scheme for the experimental implementation of the QAHM on the DW2000Q quantum annealer. The recognition network (a) is entirely classical to avoid calls to the quantum device for each of the points in the dataset (7291 digits from this sub-sampled version of the MNIST handwritten dataset). In these experiments we used two sets of 120 and 60 hidden binary variables, $\mathbf{u}^1$ and $\mathbf{u}^2$, representing the first and second hidden layers respectively. We used 266 continuous units $\mathbf{v}$ to encode the gray-scale pixels in the 16×16 images, and 10 visible binary variables to encode the class of the digits. The variables $\mathbf{u}^2$ in the second hidden layer are effectively connected all-to-all through an embedding into 1644 qubits of the quantum annealer, representing variables $\mathbf{z}$ (see Ref. [14]). Although the quantum device is part of the generation network (b), each of the 60 variables of the recognition network is replicated according to the embedding to enable the necessary correspondence between the two networks. Details of the training algorithm is given in Sec. III. (c) Digits obtained from the generator network after training. (d) Images in the training set that is closest in Euclidean distance to the generated images described in (c). Note the artificial data (c) generated by the network is not merely a copy of the training set. (e) Some artificial samples along with their most probable class according to the model. Visually, the quantum-assisted model seems to correlate class and pixels most of the time.

cussed in Section V). The wake-sleep algorithm for gray-box Helmholtz machines is summarized in Algorithm 1.

We tested our ideas on a sub-sampled version of the standard handwritten digits dataset MNIST [46]. Our training set consists of 7291 images of 16×16 gray-scale pixels and a categorical variable indicating the corresponding digit. First, we normalized pixels to take real-values in $[-1, +1]$. Second, we used a one-hot encoding for the class (i.e. $c_i^d = -1$ for $i \neq j$, $c_j^d = +1$ where j indexes the class for image d) obtaining 10 binary variables. The visible layer was connected to a first hidden layer of 120 binary variables which in turn was connected to a

second hidden layer of 60 binary variables. We used D-Wave's heuristics [47] to embed a fully-connected graph of 60 variables in DW2000Q. This resulted in a graph of 1644 qubits where the largest subgraph had 43 spins. The maps in Eqs. (21) and (22) were set up accordingly. Figure 2 shows the final model composed of two networks and a quantum annealer implementing a prior over the hidden variables $\mathbf{u}^2$. It can be easily seen that the resulting model is an engineered version of the model in Fig. 1 (a). To implement continuous variables we use a deterministic layer with hyperbolic-tangent non-linear units. So, we have to normalize the continuous data in the interval $[-1, 1]$; alternatively, we could also use stochastic Gaussian units.

We run the vanilla wake-sleep algorithm for 500 epochs with a learning rate of 0.005 for all the gradient updates. Subsequently, we trained for another 500 epochs by linearly decreasing the learning rate down to 0.0005. At each training iteration we inferred hidden configurations from the recognition network for all the data points in the training set, and we sampled 1000 artificial points from the generator network. These two sets are used to compute the gradients for the two networks as in Algorithm 1. Quantum annealing hyperparameters such as annealing time, programming thermalization and readout thermalization were set to their corresponding default values. In particular, the annealing time, which determines the time per annealing sample, was set to its minimum of 5 μs at all times to obtain samples as fast as possible.

Figure 2(c) shows samples from the generator network after training, while Fig. 2(d) shows the image in the training set that is closest in Euclidean distance at the corresponding location. We can see that the artificial data generated by the network is not merely a copy of the training set, but they do present variations and novelty in some cases; this reflects generalization capabilities which is a desired feature in a unsupervised generative model. Although these are only preliminary results, and we cannot compete with state-of-the-art ML, the artificial data often resemble digits written by humans. Indeed, the problem of generating blurry artificial images is common to other approaches as well; only the recent development of Generative Adversarial Networks [48] led to much sharper artificial images. Finally, Fig. 2(e) shows some artificial samples along with their most probable class according to the model. Visually, the model seems to correlate class and pixels most of the time. The process can be easily generalized to perform classification, where test images are provided through the recognition network and the most likely class is inferred through the generator network.

V. CONCLUSIONS AND FUTURE WORK

Despite significant effort in Quantum-Assisted Machine Learning (QAML), there has been a disconnect between most algorithmic proposals, the needs of machine learning (ML) practitioners, and the capabilities of near-term quantum devices to demonstrate quantum enhancement in the near future. Inspired by the challenges and guidelines exposed in Ref. [49], we implemented a hybrid classical-quantum architecture for unsupervised learning. We demonstrated how currently available quantum devices can be used in real-world modeling applications on datasets with higher dimensionality than apparently possible, which are defined on variables which are not binary, e.g. modeling of gray-scale handwritten digits of 16×16 pixels. In our case study, we used a noisy quantum annealer to learn an implicit prior distribution for the latent variables of a deep generative model.

Here we summarize some of the advantages and challenges with the current implementation of the Quantum-Assisted Helmholtz Machines (QAHMs) and we propose some generalizations for future work.

Advantages of the QAHM framework:

- A classical recognition network is used to approximate inference. There is no need to sample from a quantum device for each data point at each learning iteration.

- The quantum device is employed in the deepest layers of a generator network. The lowest layers stochastically transform the information contained in the qubits into artificial data. The latter can be discrete, continuous, or of a more general type, e.g. 3D vectors encoding color infomation.

- The quantum device models an abstract representation whose dimensionality is expected to be much smaller than that of the raw-data. This enables the handling of datasets of relevant size, a significant step towards real-world applications.

Challenges and why our experiments are sub-optimal:

- The sleep phase of the wake-sleep algorithm optimizes the wrong cost function [34]. Solutions found in the literature [36, 38] require full knowledge of the model's parameters which is not available under the gray-box approach employed here.

- The recognition network has to be expressive enough to closely track the true posterior. As pointed out in the original work on Helmholtz machines [34], factorized distributions are not able to model complex posteriors because of non-trivial effects such as *explaining away*. Studies shown that better likelihoods are obtained when the recognition network is equipped with more complex hidden layers (e.g. autoregressive or NADE) [36]. However, we expect the problem to be much more dramatic when using quantum distributions in the generative network as done here. This may require the introduction of a quantum distribution in the recognition network as well, hence losing one of the ad-

vantages listed above. Strategies to overcome this will be discussed elsewhere [50].

Some potential generalizations:

- The deterministic mappings in Eqs. (21) and (22), used here to translate information into and from quantum hardware, can be relaxed into trainable functions. The variables $\mathbf{z}$ in the recognition network and $\mathbf{u}^2$ in the generator network become stochastic Ising variables. Indeed, the expected value of an Ising variable u_i, conditioned on the configuration $\mathbf{u}'$ of the previous layer, is described by the hyperbolic tangent function $\mathbb{E}\left[u_i|\mathbf{u}'\right] = \tanh(c_i + \sum_j C_{ij}u'_j)$. When $C_{ij} \gg 1$ and $c_i = 0$, this function implements a majority vote of the variables in the previous layer. The replica function can be thought of a majority vote over a single spin in the previous layer. Hence, by allowing all parameters c_i and C_{ij} to be learned, we obtain a generalized version of the quantum-assisted wake-sleep algorithm introduced here. While this generalization requires fitting additional parameters, it has the potential to discover a better embedding than that found via heuristics.

- The general framework of the QAHM allows for using quantum devices in both, the recognition and the generator network [see Fig. 1(a)]. The motivation to use the quantum device only on the generator network is to bypass the issue of making calls to the quantum device for every point in the dataset, which will be required if we had the quantum device in the recognition network. It is an open question whether using the quantum device on the recognition network, as well, can significantly enhance the quality of the model learned; this question will be explored in future work. Extensions to more general quantum models will be discussed elsewhere [50].

Although the results of the current implementation on quantum annealers do not compete with state-of-the-art computer vision systems, we hope this flexible QAHM framework motivates researchers to develope novel hybrid quantum-classical ML approaches with the intention to use near-term quantum computers for intractable ML tasks such as unsupervised learning and sampling.

ACKNOWLEDGEMENTS

The work of A.P-O., J-R-G, and M.B. was supported in part by the AFRL Information Directorate under grant F4HBKC4162G001, the Office of the Director of National Intelligence (ODNI), the Intelligence Advanced Research Projects Activity (IARPA), via IAA 145483, and the U.S. Army TARDEC under the "Quantum-assisted Machine Learning for Mobility Studies" project. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of ODNI, IARPA, AFRL, U.S. Army TARDEC or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purpose notwithstanding any copyright annotation thereon. M.B. was partially supported by the UK Engineering and Physical Sciences Research Council (EPSRC) and by Cambridge Quantum Computing Limited (CQCL).

Appendix A: Approximating continuous stochastic variables in quantum annealers

Here we show how naive approaches to encoding continuous variables in quantum annealers are likely to fail. Consider the task of approximating a simple univariate Gaussian probability. If we were able to do that, we could control its mean μ, variance σ^2, and sample accordingly. While this is a trivial task in classical computers, it serves as an example to show the challenge of implementing continuous variables in quantum annealers. One way to approach the problem is to approximate the stochastic continuous variable x with the weighted sum of a large number of spins, i.e. $x = \sum_i w_i s_i$ where w_i are programmable weights in the annealer. Notice that n-ary expansions commonly used in classical computers are just special cases of this weighted sum where weights increase/decrease exponentially with the the precision (i.e. number of spins used for the encoding). This is not practical for state-of-the-art devices as it requires high-precision parameters that are not available because of noise, bias, and finite control precision. A more general weighted-sum encoding may introduce degeneracy, but this is not a problem in the machine learning setting considered as long as the probability distribution approximates the desired continuous probability distribution. Moreover, in the machine learning setting we could learn all the parameters, including the weights w_i.

Now consider approximating the Gaussian probability over x in the annealer by defining the energy function

$$\begin{aligned} E(\mathbf{s}) &= \frac{1}{2\sigma^2}\Big(\sum_i w_i s_i - \mu\Big)^2 \\ &= \frac{1}{2\sigma^2}\Big(\sum_{i\neq j} w_i w_j s_i s_j + \sum_i w_i^2 + \mu^2 - 2\mu\sum_i w_i s_i\Big) \\ &= \sum_{i\neq j} J_{ij} s_i s_j + \sum_i h_i s_i + C \end{aligned} \tag{A1}$$

where $J_{ij} = w_i w_j/2\sigma^2$ are couplings, $h_i = -\mu w_i/\sigma^2$ are local fields, and we collected the constant terms in C. The result is a fully-connected graph that must be natively implemented in hardware. That is, if we want N-bits of precision, we are required to have an N-clique in

the hardware interaction graph. To see why, assume one of the interactions is not available in hardware, that is $J_{ij} = 0$. From the definition of J_{ij} we see that either $w_i = 0$ or $w_j = 0$. Take $w_i = 0$ and notice that $J_{ik} = 0$ for each k, or in words, spin i is disconnected from the interaction graph and the variable is useless for the purpose of encoding a continuous variable. As an example, the chimera interaction graph used in in D-Wave hardware has a largest clique of size 2. Hence, the best naive encoding has 2 bits of precision and they are clearly not enough to approximate and have control over any desired Gaussian distribution.

While in this specific instance a simple solution is possible through the central-limit theorem, and more elaborated approaches may also be possible, this discussion suggests that the implementation of stochastic continuous variables in state-of-the-art quantum annealers is challenging in more general setups that go beyond the univariate Gaussian case.

[1] Harmut Neven, Vasil S Denchev, Marshall Drew-Brook, Jiayong Zhang, William G Macready, and Geordie Rose, "Binary classification using hardware implementation of quantum annealing," in *Demonstrations at NIPS-09, 24th Annual Conference on Neural Information Processing Systems* (2009) pp. 1–17.

[2] Zhengbing Bian, Fabian Chudak, William G Macready, and Geordie Rose, *The Ising model: teaching an old problem new tricks*, Tech. Rep. (D-Wave Systems, 2010).

[3] Misha Denil and Nando De Freitas, "Toward the implementation of a quantum RBM," NIPS Deep Learning and Unsupervised Feature Learning Workshop (2011).

[4] Nathan Wiebe, Daniel Braun, and Seth Lloyd, "Quantum algorithm for data fitting," Physical review letters **109**, 050505 (2012).

[5] KristenL. Pudenz and DanielA. Lidar, "Quantum adiabatic machine learning," Quantum Information Processing **12**, 2027–2070 (2013).

[6] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost, "Quantum algorithms for supervised and unsupervised machine learning," arXiv:1307.0411 (2013).

[7] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd, "Quantum support vector machine for big data classification," Phys. Rev. Lett. **113**, 130503 (2014).

[8] G. Wang, "Quantum Algorithm for Linear Regression," ArXiv e-prints (2014), arXiv:1402.0660 [quant-ph].

[9] Z. Zhao, J. K. Fitzsimons, and J. F. Fitzsimons, "Quantum assisted Gaussian process regression," ArXiv e-prints (2015), arXiv:1512.03929 [quant-ph].

[10] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost, "Quantum principal component analysis," Nature Physics **10**, 631–633 (2014).

[11] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione, "Prediction by linear regression on a quantum computer," Physical Review A **94**, 022342 (2016).

[12] Krysta M. Svore Nathan Wiebe, Ashish Kapoor, "Quantum deep learning," arXiv:1412.3489 (2015).

[13] Marcello Benedetti, John Realpe-Gómez, Rupak Biswas, and Alejandro Perdomo-Ortiz, "Estimation of effective temperatures in quantum annealers for sampling applications: A case study with possible applications in deep learning," Phys. Rev. A **94**, 022308 (2016).

[14] Marcello Benedetti, John Realpe-Gómez, Rupak Biswas, and Alejandro Perdomo-Ortiz, "Quantum-assisted learning of graphical models with arbitrary pairwise connectivity," arXiv:1609.02542 (2016).

[15] Scott Aaronson, "Read the fine print," Nature Physics **11**, 291–293 (2015), commentary.

[16] Steven H. Adachi and Maxwell P. Henderson, "Application of quantum annealing to training of deep neural networks," arXiv:1510.06356 (2015).

[17] Nicholas Chancellor, Szilard Szoke, Walter Vinci, Gabriel Aeppli, and Paul A Warburton, "Maximum-entropy inference with a programmable annealer," Scientific reports **6** (2016).

[18] Mohammad H. Amin and Evgeny Andriyash and Jason Rolfe and Bohdan Kulchytskyy and Roger Melko, "Quantum Boltzmann Machine," arXiv:1601.02036 (2016).

[19] Maria Kieferova and Nathan Wiebe, "Tomography and generative data modeling via quantum boltzmann training," arXiv preprint arXiv:1612.05204 (2016).

[20] Iordanis Kerenidis and Anupam Prakash, "Quantum recommendation systems," arXiv preprint arXiv:1603.08675 (2016).

[21] Peter Wittek and Christian Gogolin, "Quantum enhanced inference in markov logic networks," Scientific Reports **7** (2017).

[22] Thomas E. Potok, Catherine Schuman, Steven R. Young, Robert M. Patton, Federico Spedalieri, Jeremy Liu, Ke-Thia Yao, Garrett Rose, and Gangotree Chakma, "A study of complex deep learning networks on high performance, neuromorphic, and quantum computers," arXiv:1703.05364 (2017).

[23] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione, "An introduction to quantum machine learning," Contemporary Physics **56**, 172–185 (2015).

[24] Jonathan Romero, Jonathan P. Olson, and Alan Aspuru-Guzik, "Quantum autoencoders for efficient compression of quantum data," arXiv:1612.02806 (2017).

[25] Jeremy Adcock, Euan Allen, Matthew Day, Stefan Frick, Janna Hinchliff, Mack Johnson, Sam Morley-Short, Sam Pallister, Alasdair Price, and Stasja Stanisic, "Advances in quantum machine learning," arXiv preprint arXiv:1512.02900 (2015).

[26] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd, "Quantum machine learning," arXiv preprint arXiv:1611.09347 (2016).

[27] C. Ciliberto, M. Herbster, A. Davide Ialongo, M. Pontil, A. Rocchetto, S. Severini, and L. Wossnig, "Quantum machine learning: a classical perspective," ArXiv e-prints (2017), arXiv:1707.08561 [quant-ph].

[28] Yoshua Bengio *et al.*, "Learning deep architectures for ai," Foundations and trends® in Machine Learning **2**, 1–127 (2009).

[29] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh,

"A fast learning algorithm for deep belief nets," Neural computation **18**, 1527–1554 (2006).
[30] Seth Lloyd and Samuel L Braunstein, "Quantum computation over continuous variables," Physical Review Letters **82**, 1784 (1999).
[31] Hoi-Kwan Lau, Raphael Pooser, George Siopsis, and Christian Weedbrook, "Quantum machine learning over infinite dimensions," Physical Review Letters **118**, 080501 (2017).
[32] S. Das, G. Siopsis, and C. Weedbrook, "Continuous-variable quantum Gaussian process regression and quantum singular value decomposition of non-sparse low rank matrices," ArXiv e-prints (2017), arXiv:1707.00360 [quant-ph].
[33] Ian Goodfellow Yoshua Bengio and Aaron Courville, "Deep learning," (2016), mIT Press.
[34] Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal, "The" wake-sleep" algorithm for unsupervised neural networks," Science **268**, 1158 (1995).
[35] Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel, "The helmholtz machine," Neural computation **7**, 889–904 (1995).
[36] Jorg Bornschein, Samira Shabanian, Asja Fischer, and Yoshua Bengio, "Bidirectional helmholtz machines," in *International Conference on Machine Learning* (2016) pp. 2511–2519.
[37] David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski, "A learning algorithm for boltzmann machines," Cognitive science **9**, 147–169 (1985).
[38] Ruslan Salakhutdinov and Geoffrey Hinton, "Deep boltzmann machines," in *Artificial Intelligence and Statistics* (2009) pp. 448–455.
[39] Thomas E Potok, Catherine D Schuman, Steven R Young, Robert M Patton, Federico Spedalieri, Jeremy Liu, Ke-Thia Yao, Garrett Rose, and Gangotree Chakma, "A study of complex deep learning networks on high performance, neuromorphic, and quantum computers," in *Proceedings of the Workshop on Machine Learning in High Performance Computing Environments* (IEEE Press, 2016) pp. 47–55.
[40] Imre Csiszár, Paul C Shields, *et al.*, "Information theory and statistics: A tutorial," Foundations and Trends® in Communications and Information Theory **1**, 417–528 (2004).
[41] Jörg Bornschein and Yoshua Bengio, "Reweighted wake-sleep," arXiv preprint arXiv:1406.2751 (2014).
[42] Ruslan Salakhutdinov and Hugo Larochelle, "Efficient learning of deep boltzmann machines," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (2010) pp. 693–700.
[43] Wolfgang Lechner, Philipp Hauke, and Peter Zoller, "A quantum annealing architecture with all-to-all connectivity from local interactions," Science advances **1**, e1500838 (2015).
[44] Alejandro Perdomo-Ortiz, Alexander Feldman, Asier Ozaeta, Sergei Isakov, Zheng Zhu, Bryan OGorman, Helmut Katzgraber, Alexander Diedrich, Hartmut Neven, Johan de Kleer, Brad Lackey, and Rupak Biswas, "On the readiness of quantum annealers as optimization solvers," In preparation (2017).
[45] Laurent Younes, "On the convergence of markovian stochastic algorithms with rapidly decreasing ergodicity rates," Stochastics: An International Journal of Probability and Stochastic Processes **65**, 177–228 (1999).
[46] "Sub-sampled mnist version," https://github.com/marybigday/stat665-1/tree/master/data (Accessed: August 2017).
[47] Jun Cai, William G Macready, and Aidan Roy, "A practical heuristic for finding graph minors," arXiv:1406.2741 (2014).
[48] 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* (2014) pp. 2672–2680.
[49] Alejandro Perdomo-Ortiz, Marcello Benedetti, John Realpe-Gómez, and Rupak Biswas, "Opportunities and challenges for quantum-assisted machine learning in near-term quantum computers," Preprint available at arXiv.org (2017).
[50] John Realpe-Gómez, Marcello Benedetti, and Alejandro Perdomo-Ortiz, "Quantum models for quantum-like datasets in near-term devices," In preparation (2017).