Cambridge Quantum

Quantum-Assisted Helmholtz Machines — A Quantum-Classical Deep Learning Framework for Industrial Datasets in Near-Term Devices

by Marcello Benedetti, John Realpe-Gómez, Alejandro Perdomo-Ortiz

Quantum Artificial Intelligence Lab
NASA Ames Research Center, USA
USRA Research Institute for Advanced Computer Science, USA
Department of Computer Science
University College London, United Kingdom
Cambridge Quantum
MARCELLO BENEDETTI
marcello.benedetti@cambridgequantum.com

Quantum Artificial Intelligence Lab
NASA Ames Research Center, USA
SGT Inc., USA
Instituto de Matemáticas Aplicadas, Colombia
JOHN REALPE-GÓMEZ

Quantum Artificial Intelligence Lab
NASA Ames Research Center, USA
USRA Research Institute for Advanced Computer Science, USA
Department of Computer Science
University College London, United Kingdom
Cambridge Quantum
Qubitera, LLC., United Kingdom
ALEJANDRO PERDOMO-ORTIZ
20alejandro.perdomoortiz@nasa.gov

Cambridge Quantum
Terrington House
13-15 Hills Road
Cambridge CB2 1NL
United Kingdom

Published by Cambridge Quantum

31 August 2017

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

Marcello Benedetti,[1,2,3,4] John Realpe-Gómez,[1,5,6] and Alejandro Perdomo-Ortiz[1,2,3,4,7,*]

[1] *Quantum Artificial Intelligence Lab., NASA Ames Research Center, Moffett Field, CA 94035, USA*
[2] *USRA Research Institute for Advanced Computer Science, Mountain View, CA 94043, USA*
[3] *Department of Computer Science, University College London, WC1E 6BT London, UK*
[4] *Cambridge Quantum Computing Limited, CB2 1UB Cambridge, UK*
[5] *SGT Inc., Greenbelt, MD 20770, USA*
[6] *Instituto de Matemáticas Aplicadas, Universidad de Cartagena, Bolívar 130001, Colombia*
[7] *Qubitera, LLC., Mountain View, CA 94041, USA*

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 only to assist deep learning, as previous approaches have suggested, we use deep learning to extract a low-dimensional binary representation of data, suitable for processing on relatively small quantum computers. Then, the quantum hardware and deep learning architecture work together to train an unsupervised generative model. We demonstrate this concept using 1644 quantum bits of a D-Wave 2000Q quantum device to model 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 has been much interest in quantum algorithms for enhancing deep learning and other machine learning (ML) algorithms [1–33]. 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 [14–18]. 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 modeling 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 stems from their generality. Deep generative models with many layers of hidden stochastic variables have the ability to learn multimodal distributions over high-dimensional datasets [34]. Each additional layer provides an increasingly abstract representation of the data and improves the generalization capability of the model [35]. Furthermore, generative models apply to unlabeled data, which accounts for most of the public data in the Internet and most of the private data in a company. Often, the price to pay for using a generative model is the intractability of inference, training, and model selection. Generative models are trained in an unsupervised fashion, relying on variational approximations and computationally expensive Markov Chain Monte Carlo (MCMC) sampling. This is where we think quantum computation can have a significant impact. Under the hypothesis that quantum computers allow more efficient sampling, we can run the expensive subroutine on quantum hardware. This would also enable us to exploit the non-trivial graph topologies in quantum hardware to implement complex networks, 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 [36]. Some researchers have followed the latter direction [37, 38]. However, most available quantum computers do work with qubits, nicely resembling the world of classical computation. Yet, 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 is stored in the amplitudes of a quantum state. This provides an exponentially efficient representation upon which one could perform

* Correspondence: alejandro.perdomoortiz@nasa.gov

linear algebra operations. Unfortunately, it is not clear how to prepare arbitrary states of this kind in near-term quantum computers. Reading out all the amplitudes of an output vector, if required by the application, might kill or significantly hamper any speedup [13].

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 implementations on near-term quantum technologies, either quantum annealers or gate-based quantum computers. Such a low-dimensional compact representation is often stochastic and binary, in generative modeling [39]. We expect quantum devices to have a higher impact at processing 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 data points, 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.) Finally, because the quantum device works on a low-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 deep learning architectures that can be used in our framework. 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 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 variables encoding the data and $\mathbf{u} = \{u_1, \ldots, u_M\}$ are unobserved or hidden variables that serve to capture non-trivial correlations by encoding high-level features. To perform inference and learning on this model, we have to sample from the posterior distribution $P(\mathbf{u}|\mathbf{v})$, which is intractable in general. A standard approach to this problem consists of introducing a distribution $Q(\mathbf{u}|\mathbf{v})$ to approximate the true posterior. When choosing the family of such distribution, one should consider functional forms that are both expressive and tractable. The learning algorithm is then in charge of adjusting $P(\mathbf{v})$ to model the data, and adjusting $Q(\mathbf{u}|\mathbf{v})$ to approximate $P(\mathbf{u}|\mathbf{v})$.

We now consider some deep architectures that could work in synergy with quantum devices. In Fig. 1, generative models are represented as graphs of stochastic nodes where edges may be directed and undirected. We use the blue color for 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 [40–42], which consists of two networks: a *recognition network* to do approximate inference on hidden variables using information extracted from real data, and a *generator network* to generate artificial data. The recognition network implements the distribution $Q(\mathbf{u}|\mathbf{v})$ and is used to perform 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, implements the distribution $P(\mathbf{u}, \mathbf{v})$ and is used to perform top-down sampling starting from the deepest hidden layer (e.g. $\mathbf{u}^2$ in Fig. 1). The deepest hidden layer is modeled by quantum variables and quantum interactions. If the recognition and generator networks share the same quantum layer, we obtain the quantum-assisted version of a deep belief network [35, 39] (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 generator network, we obtain a quantum-assisted deep Boltzmann machine [43, 44] (QADBM; see Fig. 1 (c)).

All three quantum-assisted architectures can be readily implemented and tested on available quantum computers. However, there is a practical caveat related to the fact that deep learning architectures require large datasets. For each data point, we need to perform recognition, and that requires both QADBN and QADBM to sample from a quantum device. This amount of work would be daunting for near-term quantum computers in the case of modern datasets. The more flexible framework of QAHM opens up the possibility of using a classical recognition network, sidestepping such limitation. We now discuss the details of the QAHM.

III. MODEL DEFINITION AND LEARNING ALGORITHM

Consider a dataset $\mathcal{S} = \{\mathbf{v}^1, \ldots, \mathbf{v}^d\}$ with empirical distribution $Q_{\mathcal{S}}(\mathbf{v})$. We seek a generative model $P(\mathbf{v}) = \sum_{\mathbf{u}} P(\mathbf{u}, \mathbf{v})$, where $P(\mathbf{u}, \mathbf{v}) = P(\mathbf{v}|\mathbf{u})P_{QC}(\mathbf{u})$.

The prior distribution $P_{QC}(\mathbf{u}) = \langle \mathbf{u}|\rho|\mathbf{u}\rangle$ describes samples obtained from a quantum device. For example, it could correspond to the diagonal elements of a quantum Gibbs distribution $\rho = e^{-\beta\mathcal{H}}/\mathcal{Z}$, where $\mathcal{H}$ is the Hamil-

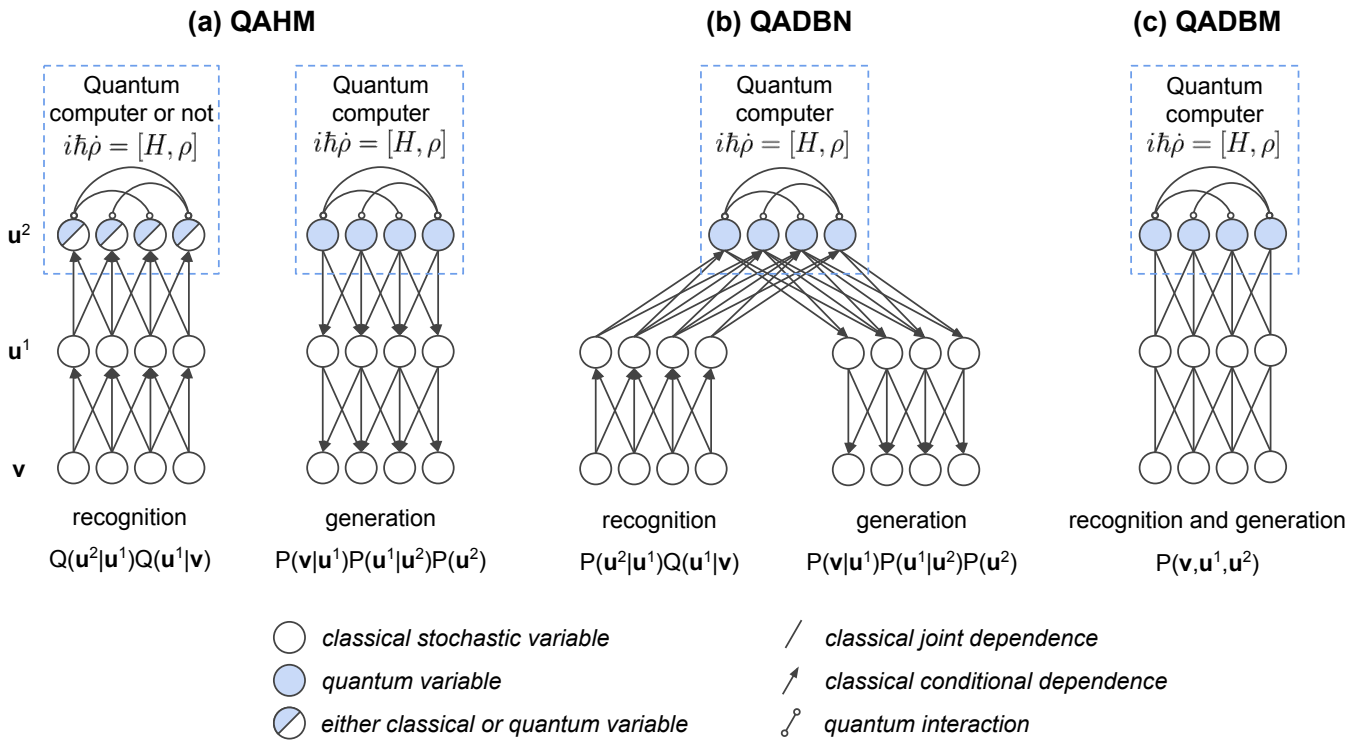

FIG. 1. Architectures for quantum-assisted machine learning (QAML). (a) quantum-assisted Helmholtz machine (QAHM); (b) quantum-assisted deep belief network (QADBN); (c) quantum-assisted deep Boltzmann machine (QADBM). We refer the reader to Sec. II for a brief description of the proposals pictured here.

tonian implemented in quantum hardware and $\mathcal{Z}$ is the partition function. For instance, in the case of quantum annealing hardware, we could have

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

where $\hat{Z}_i$ and $\hat{X}_i$ denote Pauli matrices in the z and x direction, respectively, while J_{ij}, h_i, and Γ are controllable parameters.

The conditional distribution $P(\mathbf{v}|\mathbf{u})$ stochastically translates samples from the quantum computer into samples on the domain of the data. That is, $\mathbf{v}$ could be a vector of continuous variables, binary variables, or other objects. This is a significant advantage over other quantum-assisted approaches where the visible variables are directly represented by qubits.

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

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

However, the training of a Helmholtz machine is based on the lower bound

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

where $Q(\mathbf{u}|\mathbf{v})$ is an auxiliary recognition network that approximates the intractable true posterior $P(\mathbf{u}|\mathbf{v})$. Indeed, the name of the model comes from the minimization of the non-equilibrium Helmholtz free energy which is contained in the equation above [41]. Our hybrid architecture uses a classical neural network for Q, sidestepping the need to sample from a quantum device for each data point and at each iteration of learning. This bottleneck 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. [45] for one recent such proposals).

From now on, we focus on the case of quantum Gibbs distributions. The term $\ln\langle\mathbf{u}|\rho|\mathbf{u}\rangle$ 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. [19] using the Golden-Thompson inequality. Instead, we use a simpler bound based on Jensen's inequality (see Appendix B for a derivation)

$$\ln\langle\mathbf{u}|\rho|\mathbf{u}\rangle \geq \langle\mathbf{u}|\ln\rho|\mathbf{u}.\rangle \tag{4}$$

Combining Eqs. (3) and (4), we get a tractable lower bound to maximize, i.e. the function

$$\mathcal{G}(\theta_G, \theta_{QC}) = \sum_{\mathbf{v},\mathbf{u}} Q_{\mathcal{S}}(\mathbf{v}) Q(\mathbf{u}|\mathbf{v}) \left[\ln P(\mathbf{v}|\mathbf{u}) + \langle\mathbf{u}|\ln\rho|\mathbf{u}\rangle\right], \tag{5}$$

where θ_G and θ_{QC} denote the parameters of generator network $P(\mathbf{v}|\mathbf{u})$ and quantum state ρ, respectively. In Eq. (5) we 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 closely track 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 maximization of the lower bound in Eq. (3) with respect to the parameters of the recognition network is often intractable. The wake-sleep algorithm [40] 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

$$D_{KL}\left[P(\mathbf{u}|\mathbf{v})||Q(\mathbf{u}|\mathbf{v})\right] = \sum_{\mathbf{u}} P(\mathbf{u}|\mathbf{v}) \ln \frac{P(\mathbf{u}|\mathbf{v})}{Q(\mathbf{u}|\mathbf{v})}, \quad (6)$$

averaged over the marginal $P(\mathbf{v})$ to take into account the relevance of each configuration $\mathbf{v}$. In other words, wake-sleep maximizes the function

$$\mathcal{R}(\theta_R) = \sum_{\mathbf{u},\mathbf{v}} P(\mathbf{u},\mathbf{v}) \ln Q(\mathbf{u}|\mathbf{v}), \quad (7)$$

where θ_R denotes, collectively, the parameters of the recognition network $Q(\mathbf{u}|\mathbf{v})$. In Eq. (7) 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 structure $\theta^{(t+1)} = \theta^{(t)} + \eta \nabla_\theta \mathcal{F}$, where θ stands for the parameters being updated, η is the learning rate, and $\mathcal{F}$ stands for either $\mathcal{G}$ or $\mathcal{R}$, accordingly. Since $\ln \rho = -\beta \mathcal{H} - \ln \mathcal{Z}$, for the parameters of the quantum distribution $\theta_{QC} = (J_{ij}, h_i)$ we have

$$-\frac{1}{\beta}\frac{\partial \mathcal{G}}{\partial J_{ij}} = \langle u_i u_j \rangle_Q - \langle u_i u_j \rangle_\rho, \quad (8)$$

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

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 $P_{QC}(\mathbf{u}) = \langle \mathbf{u}|\rho|\mathbf{u}\rangle$, respectively. Here we have used the property $\hat{Z}_i|u_i\rangle = u_i|u_i\rangle$.

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

$$P(\mathbf{v}|\mathbf{u}) = \sum_{\mathbf{u}^1,\ldots,\mathbf{u}^L} P_0(\mathbf{v}|\mathbf{u}^1) P_1(\mathbf{u}^1|\mathbf{u}^2) \cdots P_L(\mathbf{u}^L|\mathbf{u}), \quad (10)$$

$$Q(\mathbf{u}|\mathbf{v}) = \sum_{\mathbf{u}^1,\ldots,\mathbf{u}^L} Q_L(\mathbf{u}|\mathbf{u}^L) \cdots Q_1(\mathbf{u}^2|\mathbf{u}^1) Q_0(\mathbf{u}^1|\mathbf{v}), \quad (11)$$

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 $\mathbf{u} \equiv \mathbf{u}^{L+1}$ in the last hidden layer. More specifically, when using Bernoulli variables $u_i^\ell \in \{-1, +1\}$, we have

$$P_\ell(\mathbf{u}^\ell|\mathbf{u}^{\ell+1}) = \prod_i \pi(u_i^\ell|\mathbf{u}^{\ell+1}; A^\ell, a^\ell), \quad (12)$$

$$Q_\ell(\mathbf{u}^\ell|\mathbf{u}^{\ell-1}) = \prod_i \pi(u_i^\ell|\mathbf{u}^{\ell-1}; B^\ell, b^\ell), \quad (13)$$

where

$$\pi(u_i|\mathbf{u}'; C, c) = \left[1 + e^{-2u_i\left(\sum_j C_{ij} u'_j + c_i\right)}\right]^{-1}. \quad (14)$$

The gradients for the generative network are

$$\frac{\partial \mathcal{G}}{\partial A_{ij}^\ell} = \langle u_i^\ell u_j^{\ell+1} \rangle_Q - \langle u_i^\ell \rangle_P \langle u_j^{\ell+1} \rangle_Q, \quad (15)$$

$$\frac{\partial \mathcal{G}}{\partial a_i^\ell} = \langle u_i^\ell \rangle_Q - \langle u_i^\ell \rangle_P, \quad (16)$$

and similarly for the recognition network

$$\frac{\partial \mathcal{R}}{\partial B_{ij}^\ell} = \langle u_i^\ell u_j^{\ell-1} \rangle_P - \langle u_i^\ell \rangle_Q \langle u_j^{\ell-1} \rangle_P, \quad (17)$$

$$\frac{\partial \mathcal{R}}{\partial b_i^\ell} = \langle u_i^\ell \rangle_P - \langle u_i^\ell \rangle_Q. \quad (18)$$

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, was introduced in Ref. [46]. The authors used Q as a proposal distribution for importance sampling of P, and obtained a better gradient estimator by reducing bias and variance. Another approach was introduced in Ref. [47] in the context of deep Boltzmann machines. Samples from Q were used as starting points for a set of mean-field equations; the mean-field solutions provided a closer approximation to the expectation values required for training. Finally, there exists a contrastive version of the wake-sleep algorithm that was introduced in Ref. [35] to train deep belief networks with undirected edges. In contrastive wake-sleep, samples from Q are used to seed a Gibbs sampler for the deepest layer of P, aiding thermalization.

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. Advantages, challenges and potential generalizations are discussed in Sec. V.

IV. EXPERIMENTS

We demonstrate the QAHM framework using a D-Wave 2000Q quantum annealer hosted by the NASA Ames Research Center. The annealer implements a noisy

version of the programmed Hamiltonian in Eq. (1) defined on a sparse graph of qubit interactions. In particular, the device is designed to exploits quantum tunneling to sample low energy states at transverse field $\Gamma \approx 0$. However, non-trivial non-equilibrium effects may make samples deviate from the corresponding classical Gibbs distribution. This scenario requires some engineering of the QAHM framework as well as additional actions besides those outlined in Sec. III. We would like to stress that the algorithm can be carried out on other quantum annealing architectures [48, 49], and on more general gate-based quantum computers. Implementations in these architectures may require further, or fewer, engineering steps, and could allow more general quantum distributions.

Following the work in Ref. [15], 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 approach relies on the assumption that, despite the deviations, the estimated gradients have a positive projection in the direction of the true gradient. 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 [50].

Now, we would like to implement a fully connected prior distribution $P_{QC}(\mathbf{u})$ over hidden variables in the deepest layer. This connectivity in not available in hardware, so we map each variable to a subgraph of physical qubits. This way, the additional physical interactions between qubits can effectively encode long-range interactions. This expansion needs not be globally optimal, and can be found efficiently using heuristic techniques. The new dynamics are described by the programmed Hamiltonian

$$\widetilde{\mathcal{H}} = -\frac{1}{2}\sum_{i,j=1}^{N}\sum_{k,l=1}^{Q_i,Q_j} J_{ij}^{(kl)}\hat{Z}_i^{(k)}\hat{Z}_j^{(l)} - \sum_{i=1}^{N}\sum_{k=1}^{Q_i} h_i^{(k)}\hat{Z}_i^{(k)}. \quad (19)$$

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. Note that the couplings serve to model both the consistency within subgraphs, when $i = j$, and the correlation among subgraphs, when $i \neq j$. A factor of $1/2$ is required to avoid double counting. The gradients required to learn these parameters are similar to those in Eqs. (8) and (9), and can also be found in Ref. [15].

The model is also 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

$$z_i^{(k)} = f(\mathbf{u}, i) = u_i \quad (\text{for } k = 1, \ldots, Q_i), \quad (20)$$

$$u_i = g(\mathbf{z}, i) = \text{sign}\left(\sum_{k=1}^{Q_i} z_i^{(k)}\right). \quad (21)$$

These mappings can be thought of as non-trainable edges in the recognition and generator networks, respectively. To see why, consider a QAHM with one visible $\mathbf{v}$ and two hidden layers $\mathbf{u}^1$ and $\mathbf{u}^2$, like the one shown in Figs. 2 (a) and 2 (b). In the recognition network, the hidden variables $\mathbf{u}^2$ get replicated into higher-dimensional vectors $\mathbf{z}$ (replicas are shown with the same color). 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 $\mathbf{z}$ from a Gibbs-like distribution. Samples are mapped back to the hidden variables $\mathbf{u}^2$ using the majority vote over subgraphs (subgraphs are shown with the same color). Then, a top-down pass is used to sample the visible variables $\mathbf{v}$. Hence, every directed and undirected edge in Fig. 2 can be trained, except for the gray-colored directed edges corresponding to the fixed mappings in Eqs. (20) and (21). 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 data point.

Now, because we don't have complete knowledge of the parameters implemented by the annealer, we cannot use techniques such as importance sampling 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 [35, 47]. In principle, we could use pre-training to initialize all the trainable directed edges of our model (see Fig. 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. (8) and (9). We decided not to carry out pre-training in our small scale experiment 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 discussed in Section V). The wake-sleep algorithm for Helmholtz machines on quantum annealers is summarized in Algorithm 1.

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

use an heuristic to embed in hardware a fully connected graph corresponding to the deepest hidden layer
define mappings $f(\mathbf{u}, i)$ and $g(\mathbf{z}, i)$ from hidden variables to qubits and back
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 $\mathbf{z}^k \sim \langle \mathbf{z}|\rho|\mathbf{z}\rangle$, $u_i^k = g(\mathbf{z}^k, i)$ 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

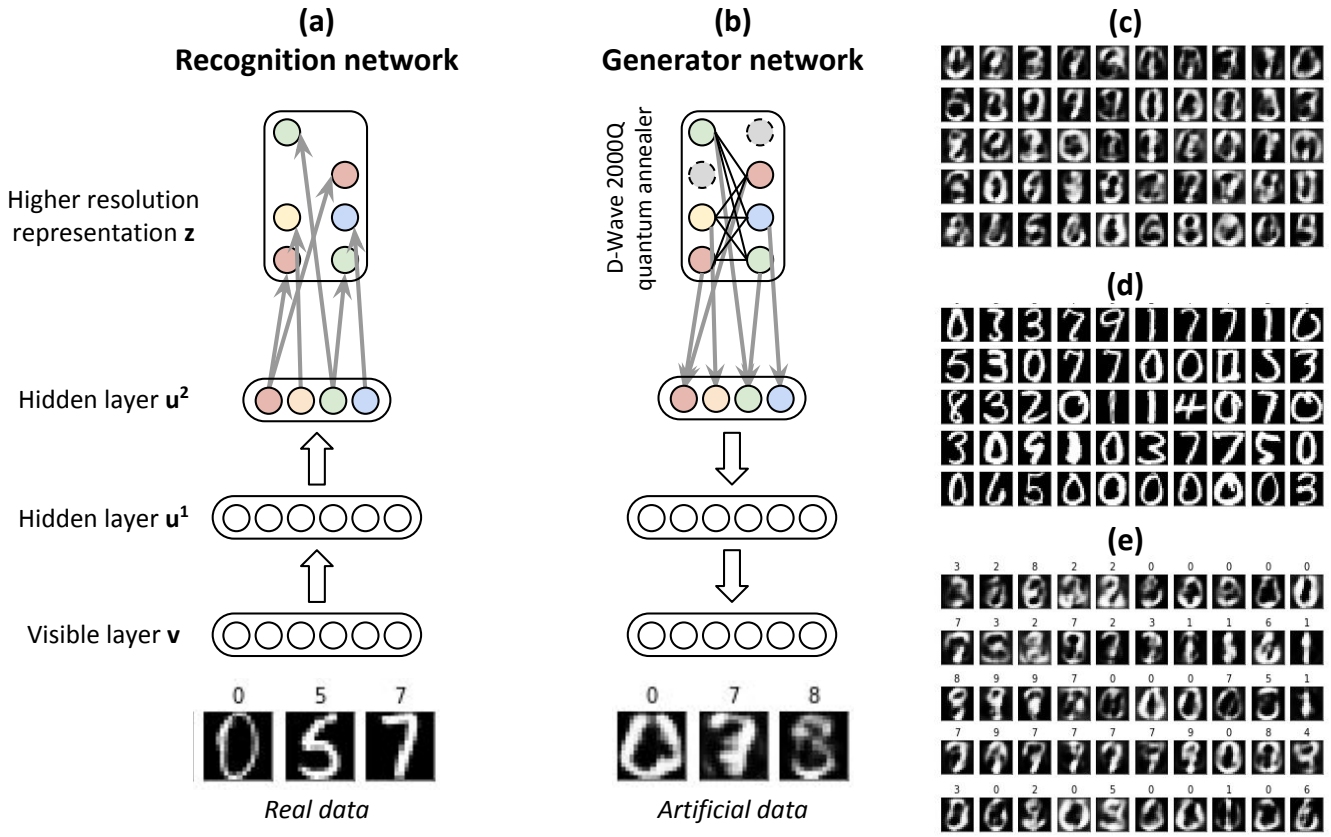

FIG. 2. Scheme for the experimental implementation of the QAHM on the D-Wave 2000Q quantum annealer for the sub-sampled MNIST dataset. The visible layer consists of 256 continuous variables $\mathbf{v}$ that encode the gray-scale pixels of 16×16 images, and 10 binary variables that encode the class. There are two hidden layers, $\mathbf{u}^1$ and $\mathbf{u}^2$, with 120 and 60 hidden binary variables, respectively. The variables $\mathbf{u}^2$ in the second hidden layer are effectively connected all-to-all through an embedding into 1644 qubits, $\mathbf{z}$, of the quantum annealer (see Ref. [15] for details). The recognition network (a) is entirely classical to avoid calling the quantum device for each point in the dataset (7291 images from the sub-sampled version of the MNIST handwritten dataset used here). The generator network (b) samples the deepest layer from the quantum annealer. The necessary correspondence between recognition and generator networks is enforced by two deterministic mappings, here represented by gray-colored edges. Panel (c) shows artificial images obtained from the generator network after training. Panel (d) shows the images in the training set that are closest in Euclidean distance to the artificial ones in panel (c). Note that artificial images are not merely copies of the training images. Panel (e) shows additional artificial images 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.

We tested our ideas on a sub-sampled version of the MNIST handwritten digits dataset [51]. Our training set consists of 7291 images of 16×16 gray-scale pixels, and a categorical variable indicating the corresponding digit. First, we rescaled 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 heuristics [52] to embed a fully connected graph of 60 variables in the D-Wave 2000Q. This resulted in a graph of 1644 qubits in total, where the largest subgraph had 43 qubits and the smallest subgraph had 18 qubits. The maps in Eqs. (20) and (21) were set up accordingly. Figure 2 shows the final model composed of two networks and a quantum annealer implementing a prior over the second hidden layer, $\mathbf{u}^2$. It can be easily seen that the final model is an engineered version of the model in Fig. 1 (a). To implement the continuous variables, $\mathbf{v}$, we used a deterministic layer of hyperbolic tangent non-linearities, which is compatible with our rescaling in the interval $[-1, 1]$. Alternatively, one can use stochastic Gaussian variables and a different, compatible, rescaling.

We ran the vanilla wake-sleep algorithm for 500 epochs with a learning rate of 0.005 for all the gradient updates. Subsequently, we trained for other 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 sampled 1000 artificial points from the generator network. These two sets are used to compute gradients as in Algorithm 1. Quantum annealing hyperparameters such as annealing time, programming thermalization and readout thermalization were set to their corresponding minimum values in order to obtain samples as fast as possible. Of particular importance, the annealing time determines how fast the quantum computing environment evolves towards the programmed Hamiltonian in Eq. (19). The use of the minimum annealing time is a well established practice due to extensive benchmarking by the combinatorial optimization community. We are not aware of similar systematic studies in the context of sampling, although we expect annealing time to have a significant impact on the form of the distribution. Because the gray-box model considered here does not require knowledge of the exact form of the distribution, we chose the minimum annealing time of $5\mu s$.

Figure 2 (c) shows samples from the generator network after training. For each of those, Fig. 2 (d) shows the image in the training set that is closest in Euclidean distance. We can see that the artificial data generated by the model is *not* merely a copy of the training set. The generated data presents variations and, in some cases, novelty, reflecting the generalization capabilities of the model. Although these preliminary results cannot compete with state-of-the-art ML, the generated data often resemble digits written by humans. Indeed, the problem of generating blurry artificial images affects other approaches as well; only the recent development of generative adversarial networks [53] 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. Inspired by the challenges and guidelines exposed in Ref. [31], 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, and 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 machine (QAHM), and we propose some generalizations for future work.

Advantages of the QAHM framework:

- A classical recognition network is used to perform approximate inference. There is no need to sample from a quantum device for each data point and for each learning iteration.
- The quantum device is employed in the deepest layers of a generator network. The lowest layers stochastically transform the information from qubits to data vectors, and back. Data vectors can be discrete, continuous, or of a more general type.
- 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 [40]. Solutions found in the literature [42, 44] 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 [40], 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) [42]. 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 advantages listed above.

Some potential generalizations:

- The deterministic mappings in Eqs. (20) and (21), used here to translate information from and to quantum hardware, can be relaxed into trainable functions. In this case, variables $\mathbf{z}$ in the recognition network and $\mathbf{u}^2$ in the generator network become stochastic Bernoulli variables. Indeed, the expected value of a Bernoulli variable $u_i \in \{-1, +1\}$, 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 as a majority vote over a single qubit in the previous layer. Hence, by allowing all parameters c_i and C_{ij} to be learned, one obtains 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 better embeddings than those found via heuristics.

- The general QAHM framework allows to use quantum devices in both the recognition and the generator networks (see Fig. 1 (a)). The motivation for using the quantum device only in the generator network is to bypass the issue of making calls to the quantum device for every point in the dataset. It is an open question whether using the quantum device in the recognition network can significantly enhance the quality of the model.

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 will motivate researchers to develop novel hybrid quantum-classical approaches, with the intention to use near-term quantum computers for intractable 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 and 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 qubits, 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 or decrease exponentially with the precision (i.e. number of qubits used for the encoding). This would not be 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 here as long as the results approximate 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. We define an energy function encoding the eigenvalues of the Hamiltonian in Eq. (1)

with zero transverse field

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

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} above, 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, qubit i is disconnected from the interaction graph. Then, qubit i is useless for the purpose of approximating the desired continuous variable. As an example, the chimera interaction graph used in the D-Wave 2000Q 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 may be challenging in more general setups that go beyond the univariate Gaussian case.

Appendix B: Derivation of the bound for quantum Gibbs distributions

We require a tractable bound for $\ln\langle\mathbf{u}|\rho|\mathbf{u}\rangle$ in order to train the QAHM when a quantum Gibbs distributions is used in the generator network. First, write the density matrix in terms of eigenvectors $|i\rangle$ and eigenvalues E_i of the Hamiltonian

$$\rho = \sum_i \frac{e^{-E_i}}{\mathcal{Z}} |i\rangle\langle i|, \tag{B1}$$

where $\mathcal{Z} = \sum_i e^{-E_i}$ is the normalization constant. Then, plug this expansion into the intractable expression and use Jensen's inequality

$$\begin{aligned} \ln\langle\mathbf{u}|\rho|\mathbf{u}\rangle &= \ln\langle\mathbf{u}|\sum_i \frac{e^{-E_i}}{\mathcal{Z}}|i\rangle\langle i|\mathbf{u}\rangle \\ &= \ln\sum_i |\langle i|\mathbf{u}\rangle|^2 \frac{e^{-E_i}}{\mathcal{Z}} \\ &\geq \sum_i |\langle i|\mathbf{u}\rangle|^2 \ln\frac{e^{-E_i}}{\mathcal{Z}} \\ &= \langle\mathbf{u}|\sum_i \ln\frac{e^{-E_i}}{\mathcal{Z}}|i\rangle\langle i|\mathbf{u}\rangle \\ &= \langle\mathbf{u}|\ln\rho|\mathbf{u}\rangle, \end{aligned} \tag{B2}$$

where $|\langle i|\mathbf{u}\rangle|^2$ are probabilities and sum up to 1.

[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] Kristen L. Pudenz and Daniel A. 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] Guoming Wang, "Quantum algorithm for linear regression," Physical Review A **96**, 012335 (2017).

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

[14] 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).
[15] Marcello Benedetti, John Realpe-Gómez, Rupak Biswas, and Alejandro Perdomo-Ortiz, "Quantum-assisted learning of hardware-embedded probabilistic graphical models," Phys. Rev. X **7**, 041052 (2017).
[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] 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).
[19] Mohammad H. Amin and Evgeny Andriyash and Jason Rolfe and Bohdan Kulchytskyy and Roger Melko, "Quantum Boltzmann Machine," arXiv:1601.02036 (2016).
[20] Mária Kieferová and Nathan Wiebe, "Tomography and generative training with quantum boltzmann machines," Phys. Rev. A **96**, 062327 (2017).
[21] Iordanis Kerenidis and Anupam Prakash, "Quantum recommendation systems," arXiv preprint arXiv:1603.08675 (2016).
[22] Peter Wittek and Christian Gogolin, "Quantum enhanced inference in markov logic networks," Scientific Reports **7** (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," Quantum Sci. Technol. **2**, 045001 (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] Unai Alvarez-Rodriguez, Lucas Lamata, Pablo Escandell-Montero, José D Martín-Guerrero, and Enrique Solano, "Quantum machine learning without measurements," arXiv preprint arXiv:1612.05535 (2016).
[28] Lucas Lamata, "Basic protocols in quantum reinforcement learning with superconducting circuits," Scientific Reports **7** (2017).
[29] Maria Schuld, Mark Fingerhuth, and Francesco Petruccione, "Quantum machine learning with small-scale devices: Implementing a distance-based classifier with a quantum interference circuit," arXiv preprint arXiv:1703.10793 (2017).
[30] 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].
[31] Alejandro Perdomo-Ortiz, Marcello Benedetti, John Realpe-Gómez, and Rupak Biswas, "Opportunities and challenges for quantum-assisted machine learning in near-term quantum computers," arXiv:1708.09757 (2017).
[32] Marcello Benedetti, Delfina Garcia-Pintos, Yunseong Nam, and Alejandro Perdomo-Ortiz, "A generative modeling approach for benchmarking and training shallow quantum circuits," arXiv:1801.07686 (2018).
[33] Edward Farhi and Hartmut Neven, "Classification with quantum neural networks on near term processors," arXiv:1802.06002 (2018).
[34] Yoshua Bengio *et al.*, "Learning deep architectures for ai," Foundations and trend in Machine Learning **2**, 1–127 (2009).
[35] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh, "A fast learning algorithm for deep belief nets," Neural computation **18**, 1527–1554 (2006).
[36] Seth Lloyd and Samuel L Braunstein, "Quantum computation over continuous variables," Physical Review Letters **82**, 1784 (1999).
[37] Hoi-Kwan Lau, Raphael Pooser, George Siopsis, and Christian Weedbrook, "Quantum machine learning over infinite dimensions," Physical Review Letters **118**, 080501 (2017).
[38] 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].
[39] Ian Goodfellow Yoshua Bengio and Aaron Courville, "Deep learning," (2016), mIT Press.
[40] Geoffrey E. Hinton, Peter Dayan, Brendan J. Frey, and Radford M. Neal, "The wake-sleep algorithm for unsupervised neural networks," Science **268**, 1158 (1995).
[41] Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel, "The helmholtz machine," Neural computation **7**, 889–904 (1995).
[42] Jorg Bornschein, Samira Shabanian, Asja Fischer, and Yoshua Bengio, "Bidirectional helmholtz machines," in *International Conference on Machine Learning* (2016) pp. 2511–2519.
[43] David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski, "A learning algorithm for boltzmann machines," Cognitive science **9**, 147–169 (1985).
[44] Ruslan Salakhutdinov and Geoffrey Hinton, "Deep boltzmann machines," in *Artificial Intelligence and Statistics* (2009) pp. 448–455.
[45] 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.
[46] Jörg Bornschein and Yoshua Bengio, "Reweighted wake-sleep," arXiv preprint arXiv:1406.2751 (2014).
[47] 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.
[48] Wolfgang Lechner, Philipp Hauke, and Peter Zoller, "A quantum annealing architecture with all-to-all connectiv-

ity from local interactions," Science advances **1**, e1500838 (2015).

[49] Alejandro Perdomo-Ortiz, Alexander Feldman, Asier Ozaeta, Sergei V. Isakov, Zheng Zhu, Bryan O'Gorman, Helmut G. Katzgraber, Alexander Diedrich, Hartmut Neven, Johan de Kleer, Brad Lackey, and Rupak Biswas, "On the readiness of quantum optimization machines for industrial applications," arXiv:1708.09780 (2017).

[50] 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).

[51] "A sub-sampled version of the mnist dataset," https://github.com/marybigday/stat665-1/tree/master/data (Accessed: August 2017).

[52] Jun Cai, William G Macready, and Aidan Roy, "A practical heuristic for finding graph minors," arXiv:1406.2741 (2014).

[53] 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.