

PAPER • OPEN ACCESS

A path towards quantum advantage in training deep generative models with quantum annealers

To cite this article: Walter Winci *et al* 2020 *Mach. Learn.: Sci. Technol.* 1 045028

View the [article online](#) for updates and enhancements.



PAPER

OPEN ACCESS


RECEIVED
4 March 2020REVISED
12 June 2020ACCEPTED FOR PUBLICATION
1 July 2020PUBLISHED
29 October 2020

Original Content from
this work may be used
under the terms of the
[Creative Commons
Attribution 4.0 licence](#).

Any further distribution
of this work must
maintain attribution to
the author(s) and the title
of the work, journal
citation and DOI.



A path towards quantum advantage in training deep generative models with quantum annealers

Walter Vinci^{1,2,7}, Lorenzo Buffoni^{3,4,5} , Hossein Sadeghi³, Amir Khoshaman³, Evgeny Andriyash³
and Mohammad H Amin^{3,6}

¹ Quantum Artificial Intelligence Lab. (QuAIL), Exploration Technology Directorate, NASA Ames Research Center, Moffett Field, CA 94035, United States of America

² Stinger Ghaffarian Technologies Inc., Greenbelt, MD 20770, United States of America

³ D-Wave Systems Inc., 3033 Beta Avenue, Burnaby, BC Canada V5G 4M9

⁴ Department of Physics and Astronomy, University of Florence, via G. Sansone 1, I-50019 Sesto Fiorentino, Italy

⁵ Department of Information Engineering, University of Florence, via S. Marta 3, I-50139 Sesto Fiorentino, Italy

⁶ Department of Physics, Simon Fraser University, Burnaby, BC Canada V5A 1S6

⁷ Author to whom any correspondence should be addressed

E-mail: walter.vinci@nasa.gov

Keywords: quantum annealing, generative models, variational autoencoders, sampling, quantum advantage, deep learning

Abstract

The development of quantum-classical hybrid (QCH) algorithms is critical to achieve state-of-the-art computational models. A QCH variational autoencoder (QVAE) was introduced in reference [1] by some of the authors of this paper. QVAE consists of a classical auto-encoding structure realized by traditional deep neural networks to perform inference to and generation from, a discrete latent space. The latent generative process is formalized as thermal sampling from a quantum Boltzmann machine (QBM). This setup allows quantum-assisted training of deep generative models by physically simulating the generative process with quantum annealers. In this paper, we have successfully employed D-Wave quantum annealers as Boltzmann samplers to perform quantum-assisted, end-to-end training of QVAE. The hybrid structure of QVAE allows us to deploy current-generation quantum annealers in QCH generative models to achieve competitive performance on datasets such as MNIST. The results presented in this paper suggest that commercially available quantum annealers can be deployed, in conjunction with well-crafted classical deep neural networks, to achieve competitive results in unsupervised and semisupervised tasks on large-scale datasets. We also provide evidence that our setup is able to exploit large latent-space QBMs, which develop slowly mixing modes. This expressive latent space results in slow and inefficient classical sampling and paves the way to achieve quantum advantage with quantum annealing in realistic sampling applications.

1. Introduction

Deep learning [2–6], in which a labeled dataset is used to train a statistical model to solve tasks such as clustering and classification, is currently revolutionizing the field of supervised learning. Deep neural networks (NN) are now commonly used in many scientific and industrial applications. Unlike supervised learning [7], unsupervised learning is a much harder, and still largely unsolved, problem. And yet, it has the appealing potential to learn the hidden statistical correlations of large unlabeled datasets [5, 8, 9], which constitute the vast majority of data available today.

Training and deployment of large-scale machine learning models, especially for unsupervised learning, faces computational challenges [10] that are only partially met by the development of special purpose classical computing units such as GPUs. This has led to an interest in applying quantum computing to machine learning tasks [11–15] and to the development of several quantum algorithms [16–19] with the potential to accelerate training. Most quantum machine learning (QML) algorithms need fault-tolerant quantum computation [20–22], which requires the large-scale integration of millions of qubits and is still

not available today. It is however believed that QML will provide the first breakthrough algorithms to be implemented on commercially available quantum annealers [23, 24] and gate-model devices [25, 26]. For example, small gate-model devices and quantum annealers have been used to perform quantum heuristic optimization [26–31] to solve clustering [32] and classification problems [33–36].

Current-generation quantum annealers physically simulate a transverse-field Ising model and operate in interaction with in a thermal environment [37]. Perhaps the most natural use of quantum annealers is thus as samplers from the Boltzmann distribution of an Ising system [38–40]. This observation inspired the use of quantum annealers to perform quantum-assisted training of classical Boltzmann machines (CBMs)⁸ [41–47], powerful and versatile probabilistic models that can be trained in either a supervised or unsupervised fashion. Training of CBMs requires computationally-expensive techniques, such as persistent contrastive divergence (PCD) [48] and population annealing (PA) [49], to accurately sample from the correct thermal distribution. Quantum annealers with advanced annealing control schedules for the transverse field allow sampling from both CBM and quantum Boltzmann machine (QBM) [50, 51]. QBMs [52] might be able to better model input datasets thanks to the more complex correlations realized by quantum states. Training QBM with classical algorithms is however prohibitively expensive since it requires the use of Quantum Monte Carlo techniques.

1.1. Quantum-Classical hybrid models

Early numerical studies showed that Boltzmann machines (BMs) with the quasi two-dimensional connectivities typical of current-generation quantum annealers perform poorly [53], even on simple standard datasets such as MNIST [54]. A possible approach to circumvent the limitation above is the following quantum-classical hybrid (QCH) paradigm: 1) employ classical feed-forward NN to encode the data into a compressed representation that is more easily processed by a quantum device and 2) jointly train the NN and the quantum device. This approach was pioneered in reference [55], where the authors introduced a quantum-assisted Helmholtz machine (QAHM), a generative model with latent variables. A QAHM has an *inference* (encoder), a *generation* (decoder) network and a generative process that is represented by BM and can thus be simulated by a quantum annealer. QAHMs do not have a well-defined loss function [9], which means that training does not scale well, even to standard machine-learning datasets such as MNIST. This approach was taken a step further in reference [1] by using variational autoencoders (VAE), a class of generative models with latent variables that provide an efficient inference mechanism [56, 57]. VAEs can be trained by minimizing the evidence lower bound (ELBO), a variational lower bound to the exact log-likelihood. The ELBO is a well-defined loss function with fully propagating gradients that can be efficiently optimized via backpropagation. This allows us to achieve competitive and scalable performance on large-scale datasets. Upon discretization of the latent space [58, 59], the generative process can be then realized by CBM or QBM. A QCH variational autoencoder (QVAE) is a VAE with an integrated QBM. Crucially, QVAEs can also be trained by optimizing a lower bound to the exact log-likelihood [1].

Our hybrid approach exploits a large amount of classical computational resources (backpropagation through deep NN performed on GPUs). This is advantageous since it allows us to achieve competitive results on relatively large-scale datasets such as MNIST. However, validating the training with quantum annealers is a major challenge: VAEs can train well even with unstructured priors (e.g., a product of Gaussian or Bernoulli distributions). One thus needs to carefully validate and demonstrate any performance improvement due to the use of structured samples coming from well-trained BMs. To this end, existence of a well-defined loss function, which allows for a quantitative comparison between different models, is critical. In order to single out the performance improvement due to computations that can be off-loaded to a quantum annealer (sampling from the BM), we always focus on comparing the performance of a model trained with a nontrivially connected BM (for example with full or Chimera connectivity **D**) to a baseline in which we employ a BM with no connectivity; that is, a set of independent Bernoulli variables. After a certain number of gradient updates, training of such models will have used exactly the same amount of classical resources that cannot be off-loaded to the quantum annealer, with the only difference being the computational effort needed to correctly sample from the latent-space BM.

A complete description of our model will be given in section 2, more theoretical details about hybrid quantum-classical VAEs can be found in [1].

1.2. Toward quantum advantage

We successfully train a convolutional QVAE with 288 latent units and Chimera-structured QBM (a 6-by-6 Chimera graph; see appendix **D**) using only D-Wave 2000Q quantum annealers to draw the samples required

⁸In the following, we will use the acronyms QBM and CBM to explicitly refer to quantum and classical Boltzmann machines (BM), while we will simply use BM to refer to either QBM or CBM when the distinction is not necessary.

to evaluate the gradients for the QBM parameters. We then use several techniques to validate the trained models. In particular, to perform quantitative evaluation we use auxiliary CBMs trained on the annealer samples to quantitatively estimate the test log-likelihood. Using these auxiliary CBMs is necessary in practice since we can easily evaluate the log-likelihood of reasonably large CBMs, while it would not be feasible to evaluate the log-likelihood of large QBMs. This approach is consistent because we empirically find that the quantum annealer samples from a QBM with a very small effective transverse field and it is thus close to simulating a CBM. With this approach, we show that models trained on Chimera connectivity outperform their trivial ('Bernoulli') baseline and can achieve competitive performance on the MNIST dataset.

The next question is whether our approach offers a path toward obtaining quantum advantage with quantum annealers in machine learning applications. To achieve such a goal, we need to exploit large latent-space BMs that develop complex multimodal probability distributions (i.e., a complex energy landscape) from which sampling is classically inefficient. We give evidence that this is indeed possible within our setup. For example, we demonstrate that the BMs placed in the latent space develop nontrivial modes that are likely to cause Monte Carlo algorithms to have long mixing times. Moreover, we show that training on more complex datasets likely takes advantage of larger BMs to improve performance. In addition, we discuss the role of connectivity, emphasizing its importance even in this hybrid approach and the necessity to develop device-specific classical NN to better exploit physical connectivities such as the Chimera graph.

The structure of the paper is as follows. In section 2 we review VAE and the implementations of discrete latent variables, a necessary step to implement BMs and QBMs in their latent space. Section 3 motivates the use of quantum annealers as samplers to train quantum and classical BMs. In section 4 we report our experiments in training VAEs with D-Wave 2000Q systems. In section 5 we discuss a possible path toward quantum advantage in our setup. Finally, we present our conclusions in section 6.

2. Variational autoencoders

In this section, we will briefly introduce VAEs and describe their extension to discrete latent variables; a necessary step to hybridize with quantum priors and to perform quantum-assisted training.

In generative modeling, the goal is to train a probabilistic model such that the model distribution $p_{\theta}(\mathbf{X})$ (where θ are the parameters of the model) is as close as possible to the data distribution, $p_{\text{data}}(\mathbf{X})$, which is unknown but assumed to exist. The ensemble $\mathbf{X} = \{\mathbf{x}^d\}_{d=1}^N$ represents the training set; that is, N independent and identically distributed samples coming from $p_{\text{data}}(\mathbf{X})$. Maximum likelihood estimation (MLE) is arguably the most mathematically justified method for training probabilistic models (although needs not always to be the preferred, see reference [60] for a discussion). With MLE training the optimal model parameters are obtained by maximizing the log-likelihood $L(\mathbf{X}, \theta)$ of the dataset with respect to the model:

$$L(\mathbf{X}, \theta) = \sum_{\mathbf{x} \in \mathbf{X}} p_{\text{data}}(\mathbf{x}) \log p_{\theta}(\mathbf{x}) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log p_{\theta}(\mathbf{x})], \quad (1)$$

where $\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\dots]$ is the expectation over $p_{\text{data}}(\mathbf{x})$. Similarly to generative adversarial networks [61], VAEs [56] are 'directed' probabilistic models with latent variables (see figure 1): the model distribution, defined as the joint between the visible units \mathbf{x} and latent units ζ , is explicitly parameterized as the product of the 'prior' $p_{\theta}(\zeta)$ and 'marginal' $p_{\theta}(\mathbf{x}|\zeta)$ distributions, $p_{\theta}(\mathbf{x}, \zeta) = p_{\theta}(\mathbf{x}|\zeta)p_{\theta}(\zeta)$. The model prediction for the data is then obtained by marginalizing over the latent units:

$$p_{\theta}(\mathbf{x}) = \int p_{\theta}(\mathbf{x}|\zeta)p_{\theta}(\zeta)d\zeta. \quad (2)$$

Generative models with latent variables can potentially learn and encode useful representations of the data in the latent space. This is an important property that can be exploited in many practical applications [62–65] to improve other tasks such as supervised and semi-supervised learning [66]. The drawback is 'intractable inference' due to the appearance of integrals such as the one in equation (2). Essentially, VAEs remove the necessity to evaluate such integrals by introducing a variational approximation $q_{\phi}(\zeta|\mathbf{x})$ to the true posterior $p_{\theta}(\zeta|\mathbf{x})$. A so-called 'reparameterization trick' is also introduced to obtain an efficient and low-variance estimate of the gradients needed for training. We will briefly review these two important elements in the next two sections.

2.1. Variational inference

Training generative models with latent variables via MLE requires the evaluation of the intractable integral of equation (2) to calculate the posterior distribution $p_{\theta}(\zeta|\mathbf{x})$. VAEs circumvent this problem by introducing a tractable variational approximation $q_{\phi}(\zeta|\mathbf{x})$ to the true posterior [67], with variational parameters ϕ (see

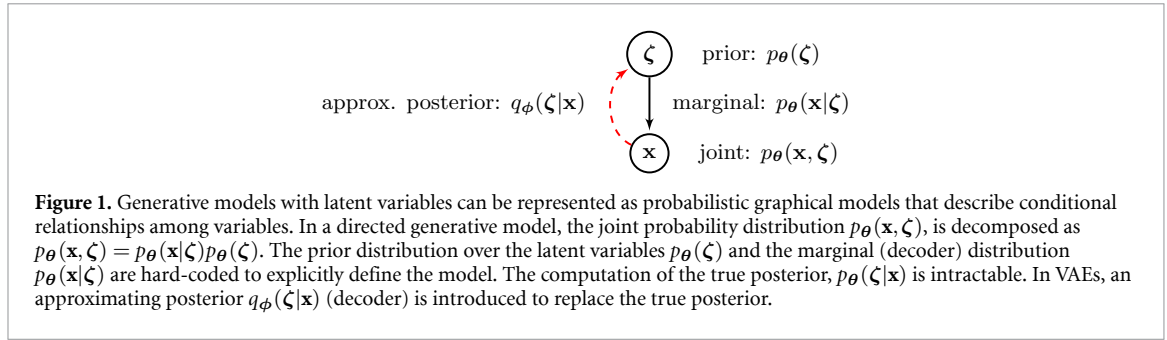


figure 1). VAEs are then trained by maximizing a variational lower bound $\mathcal{L}(\mathbf{x}, \theta, \phi)$ to the log-probabilities $\log p_{\theta}(\mathbf{x})$:

$$\begin{aligned} \mathcal{L}(\mathbf{x}, \theta, \phi) &= \log p_{\theta}(\mathbf{x}) - \mathbb{E}_{\zeta \sim q_{\phi}(\zeta|\mathbf{x})} \left[\log \frac{q_{\phi}(\zeta|\mathbf{x})}{p_{\theta}(\zeta|\mathbf{x})} \right] \equiv \\ &\equiv \log p_{\theta}(\mathbf{x}) - D_{KL}(q_{\phi}(\zeta|\mathbf{x}) || p_{\theta}(\zeta|\mathbf{x})), \end{aligned} \quad (3)$$

where $D_{KL}(q_{\phi}(\zeta|\mathbf{x}) || p_{\theta}(\zeta|\mathbf{x}))$ is the Kullback-Leibler divergence (KL divergence) between the true and approximating posteriors. Since KL divergences are always non-negative, we have

$$\mathcal{L}(\mathbf{x}, \theta, \phi) \leq \log p_{\theta}(\mathbf{x}), \quad (4)$$

which immediately gives:

$$\begin{aligned} \mathcal{L}(\mathbf{X}, \theta, \phi) &\equiv \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\mathcal{L}(\mathbf{x}, \theta, \phi)] \\ &\leq \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log p_{\theta}(\mathbf{x})] \equiv L(\mathbf{X}, \theta), \end{aligned} \quad (5)$$

where $\mathcal{L}(\mathbf{X}, \theta, \phi)$ is called the ELBO. The ELBO can be written in terms of tractable quantities:

$$\begin{aligned} \mathcal{L}(\mathbf{X}, \theta, \phi) &= \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \left[\mathbb{E}_{\zeta \sim q_{\phi}(\zeta|\mathbf{x})} [\log p_{\theta}(\mathbf{x}|\zeta)] + \right. \\ &\quad \left. - D_{KL}(q_{\phi}(\zeta|\mathbf{x}) || p_{\theta}(\zeta)) \right]. \end{aligned} \quad (6)$$

The marginal $p_{\theta}(\mathbf{x}|\zeta)$ and approximating posterior $q_{\phi}(\zeta|\mathbf{x})$, also called ‘decoder’ and ‘encoder’, respectively. In our set-up, both encoder and decoder are a product of Bernoulli distributions whose parameters (probabilities, or logits) are modeled by the convolutional NN described in section A.

2.2. The reparameterization trick

To train VAEs, we need to calculate the derivatives of the objective function (equation (6)) with respect to the generative (θ) and inference (ϕ) parameters. The naive evaluation of ∂_{ϕ} of terms of the type $\mathbb{E}_{\zeta \sim q_{\phi}} [f(\zeta)]$ is called REINFORCE [68]. With the use of the identity $\partial_{\phi} q_{\phi} = q_{\phi} \partial_{\phi} \log q_{\phi}$, one has:

$$\partial_{\phi} \mathbb{E}_{\zeta \sim q_{\phi}} [f(\zeta)] = \mathbb{E}_{\zeta \sim q_{\phi}} [f(\zeta) \partial_{\phi} \log q_{\phi}]. \quad (7)$$

However, the term above has high variance and requires intricate variance-reduction mechanisms to be of practical use [69].

A better approach is to write the random variable ζ as a deterministic function of the distribution parameters ϕ and of an additional auxiliary random variable ρ . The latter is given by a probability distribution $p(\rho)$ that does not depend on ϕ . This reparameterization $\zeta(\phi, \rho)$ is appropriately chosen so that one can write $\mathbb{E}_{\zeta \sim q_{\phi}} [f(\zeta)] = \mathbb{E}_{\rho \sim p(\rho)} [f(\zeta(\phi, \rho))]$. Therefore, we can move the derivative inside the expectation with no difficulties:

$$\partial_{\phi} \mathbb{E}_{\zeta \sim q_{\phi}} [f(\zeta)] = \mathbb{E}_{\rho \sim p(\rho)} [\partial_{\phi} f(\zeta(\phi, \rho))]. \quad (8)$$

This is called the *reparameterization trick* [56] and its efficient implementation is responsible for the recent success and proliferation of VAE.

2.3. VAE with discrete latent variables

The application of the reparameterization trick as in equation (8) requires that $f(\zeta(\phi, \rho))$ be differentiable, so the latent variables ζ are continuous. However, discrete latent units can be indispensable to represent the right distributions, such as in attention models, language modeling and reinforcement learning [66, 70, 71]. For example, a latent space composed of discrete variables can learn to disentangle content and style information of images in an unsupervised fashion [72]. Several methods have thus been developed to circumvent the non-differentiability of discrete latent units [69, 73–75]. In the context of VAE, the reparameterization trick has been extended to discrete variables by either relaxation of discrete variables into continuous variables [59, 70, 76] or by introducing smoothing functions [58]. In reference [1], QVAE was introduced based on the implementation of reference [58]. In this work, we follow the implementation of reference [59], which gives biased estimates but provides a much simpler and flexible implementation.

To set up a notation that we keep throughout the paper, we now assume the prior distribution is defined on a set of discrete variables $\zeta \sim p_\theta(\zeta)$, with $\zeta \in \{0, 1\}^L$. Given a discrete variable z with mean q and logit $l = \sigma^{-1}(q) = \log(q) - \log(1 - q)$ (where $\sigma = 1/[1 + \exp(-)]$ is the sigmoid function), a non-differentiable implementation of equation (8) for discrete variables can be obtained as

$$z = \Theta[\rho - (1 - q)] = \Theta[\sigma^{-1}(\rho) + l], \tag{9}$$

where Θ is the Heaviside function and the random variable $\rho \in [0, 1]$ is distributed according to a uniform distribution \mathcal{U} . In the second equality, we have used the fact that the inverse sigmoid function is monotonic.

A continuous smoothing (also known as the Gumbel trick [76]), is performed by replacing the Heaviside function with the sigmoid function:

$$z = \Theta[\sigma^{-1}(\rho) + l] \rightsquigarrow \zeta = \sigma\left(\frac{\sigma^{-1}(\rho) + l}{\tau}\right), \tag{10}$$

where τ is a temperature parameter introduced to control the smoothing. Typically, τ is annealed from large to low values during training. For large values of τ , the bias introduced by substituting ζ with ζ everywhere in the loss function is large, but the gradients propagating through ζ are also large, facilitating training. Conversely, for low values of τ the bias is reduced but gradients vanish and training stops. Evaluation of trained models is done in the limit $\tau \rightarrow 0$, where $\zeta \rightarrow \zeta$. Throughout this paper, we will use BMs to provide powerful and expressive prior distributions defined on discrete variables. For CBMs:

$$\begin{aligned} p_\theta(\zeta) &\equiv e^{-\mathcal{H}_\theta(\zeta)} / Z_\theta, & Z_\theta &\equiv \sum_{\zeta} e^{-\mathcal{H}_\theta(\zeta)}, \\ \mathcal{H}_\theta(\zeta) &\equiv \sum_l z_l b_l + \sum_{l < m} W_{lm} z_l z_m. \end{aligned} \tag{11}$$

To train a VAE with CBM prior, following the prescription of the previous section, we formally replace $p_\theta(\zeta) \rightsquigarrow p_\theta(\zeta)$. As usual, the gradient of log-probability is given by the difference between a positive and negative phase:

$$\partial \log p_\theta(\zeta_\phi) = -\partial \mathcal{H}_\theta(\zeta_\phi) + \mathbb{E}_{\tilde{\zeta} \sim p_\theta} [\partial \mathcal{H}_\theta(\tilde{\zeta})]. \tag{12}$$

In the equation above, we have highlighted the fact that the smoothed latent samples ζ_ϕ depend on the variational parameters ϕ . The model samples $\tilde{\zeta}$, however, remain discrete variables sampled from the CBM and are thus not smoothed during training [59].

2.4. Hybridization with quantum prior

Once we have a framework to train VAEs with discrete latent variables, we can consider QCH VAEs in which the generative process $\zeta \sim p_\theta(\zeta)$ is realized by measuring the computational basis on a given quantum state ρ_θ . Such quantum states can be realized by a quantum circuit or via a quantum annealing process controlled by a set of parameters θ we wish to adjust during training of the model.

As introduced in reference [1], QVAE is obtained by assuming the quantum state ρ_θ is a Gibbs state of a transverse field Ising model; i.e., a QBM [52]. The prior $p_\theta(\zeta)$ distribution is then given by:

$$\begin{aligned} p_\theta(\zeta) &\equiv \text{Tr}[\Lambda_\zeta e^{-\mathcal{H}_\theta}] / Z_\theta, & Z_\theta &\equiv \text{Tr}[e^{-\mathcal{H}_\theta}], \\ \mathcal{H}_\theta &= \sum_l \sigma_l^x \Gamma_l + \sum_l \sigma_l^z b_l + \sum_{l < m} W_{lm} \sigma_l^z \sigma_m^z, \end{aligned} \tag{13}$$

where $\Gamma, \mathbf{h}, \mathbf{W} \in \{\theta\}$, $\Lambda_\zeta \equiv |\zeta\rangle\langle\zeta|$ is the projector on the classical state ζ and $\sigma_i^{x,z}$ are Pauli operators. Unlike for a classical BM, the direct evaluation of the gradients of the $\log p_\theta(\zeta)$ above is intractable. As discussed in reference [52], a possible workaround is to perform the following substitution:

$$p_\theta(\zeta) = \text{Tr}[\Lambda_\zeta e^{-\mathcal{H}_\theta}]/Z_\theta \rightarrow \tilde{p}_\theta(\zeta) = e^{-\mathcal{H}_\theta(\zeta)}/Z_\theta \quad (14)$$

in the ELBO \mathcal{L} , where $\mathcal{H}_\theta(\zeta) \equiv \langle\zeta|\mathcal{H}_\theta|\zeta\rangle$, to obtain the so-called quantum ELBO (Q-ELBO) $\tilde{\mathcal{L}}$. As a consequence of the Golden-Thompson inequality $\text{Tr}[e^{A+B}] \geq \text{Tr}[e^{A+B}]$, one has:

$$p_\theta(\zeta) \geq \tilde{p}_\theta(\zeta) \Rightarrow \mathcal{L} \geq \tilde{\mathcal{L}}. \quad (15)$$

The Q-ELBO $\tilde{\mathcal{L}}$ is thus a lower bound to the ELBO with tractable gradients that can be used during training. The derivatives of the log-probabilities $\log \tilde{p}_\theta(\zeta)$ can be estimated via sampling from the QBM [52]:

$$\partial \log \tilde{p}_\theta(\zeta) = -\partial \mathcal{H}_\theta(\zeta) + \mathbb{E}_{\tilde{\zeta} \sim p_\theta}[\partial \mathcal{H}_\theta(\tilde{\zeta})], \quad (16)$$

where $\tilde{\zeta}$ are the model samples distributed according to the quantum Boltzmann distribution. The use of the Q-ELBO and its gradients precludes the training of the transverse field Γ [52], which is treated as a constant (hyper-parameter) throughout the training. Training via the Q-ELBO is performed as in the BM case, by smoothing $\zeta \rightsquigarrow \tilde{\zeta}$.

3. Sampling with quantum annealers

Currently manufactured quantum annealers physically implement a transverse-field Ising model:

$$\mathcal{H}(s) = A(s) \sum_l \sigma_l^x + B(s) \left[\sum_l \sigma_l^z h_l + \sum_{l < m} J_{lm} \sigma_l^z \sigma_m^z \right],$$

where $s \in [0, 1]$ is a control parameter and $A(s)$ and $B(s)$ are, respectively, decreasing and increasing monotonic functions of the parameter s with $A(0) \gg B(0)$ and $A(1) \ll B(1)$. Quantum annealers operate immersed in a thermal environment. There is theoretical and numerical evidence [37, 42, 77, 78] that when the anneal is performed sufficiently slowly the system above is in thermal equilibrium with the environment. This property can be exploited to turn quantum annealers into programmable Boltzmann samplers. Thermal relaxation rates are controlled by the intensity of the transverse field $A(s)$. At the beginning of the anneal, relaxation times are small and the system proceeds through a sequence of thermal states. As the anneal proceeds, relaxation times grow larger and eventually the state of the system freezes at the point s^* where relaxation times roughly become larger than the annealing time t_a .

With the above picture in mind, we can use quantum annealers to sample from the QBM defined in equation (13) with:

$$\begin{aligned} b_l &= \beta_{\text{eff}}^* h_l, & W_{lm} &= \beta_{\text{eff}}^* J_{lm}, & \Gamma_l &= \beta_{\text{eff}}^* \Gamma^*, \\ \beta_{\text{eff}} &\equiv B(s^*)/\beta_{\text{phys}}, & \Gamma^* &\equiv A(s^*)/B(s^*). \end{aligned} \quad (17)$$

Advanced control techniques for the anneal schedule (such as pauses and fast ramps present in the latest generation of D-Wave quantum annealers) allow in principle to control the freezing point s^* . To perform such sampling we used the publicly available Solver API provided by D-Wave [79].

Note however that the explicit knowledge of the effective transverse field Γ^* is not necessary. QBMs are trained using the gradients evaluated according to equation (16): the first term does not depend on the transverse field, while the second term depends only implicitly on the transverse field in determining the statistics of the model samples. This observation allows us to train the QBM using a quantum annealer via maximization of the Q-ELBO without explicitly knowing the value of the effective transverse field Γ^* at which the annealer samples. In the following we assume that for the models and datasets under consideration freezing happens late in the anneal. This means we effectively sample from a QBM that is very close to a CBM. This motivates and justifies our approach to validate the learned models using auxiliary CBMs, as described in the next section. Alternatively, our approach can be also seen as using quantum annealers to quantum-assist training of VAEs with latent-space CBMs.



Figure 2. Images generated by sampling latent configurations with a quantum annealer that are subsequently transformed by a classical deconvolutional decoder. The classical networks and the quantum annealer weights have been trained end-to-end for 2000 epochs on the MNIST dataset using the quantum annealer as a sampler for estimating the gradients of the annealing parameters.

4. Training VAE with quantum annealers

We have implemented a convolutional VAE whose prior is implemented by a QBM. To improve the performance of the model, we use several techniques such as learning-rate and KL-term annealing, importance-weight annealing, convolution gating and batch normalization. We give a detailed description of the model in appendix A. In this section, we restrict ourselves to a Chimera structured QBM with 288 latent units (a six-by-six patch of Chimera cells) and present our results with models trained end-to-end by using samples drawn with D-Wave 2000Q quantum annealers on the common handwritten digit dataset MNIST [54]. Samples used to estimate the negative phase (second term of equation (16)) are obtained following the prescription given in appendix B.

The effective temperature β_{eff}^* must be chosen appropriately to correctly train the parameters of the inference network q_{ϕ} (see appendix C for a more detailed discussion). The parameter β_{eff}^* can be considered as a multiplicative correction for the learning rate of the prior parameters $b, W \in \theta$. However, this observation is not true for the inference parameters ϕ , whose gradients also propagate through the first term in equation (16) via $\zeta(\phi, \rho)$. Due to our simple forward-anneal schedule, we expect the value of β_{eff}^* to change during training. To account for this effect, in this work we employ a real-time β_{eff}^* estimation as explained in appendix C. In future works, we expect to be able to train at a fixed-temperature with appropriate pause-and-ramp annealing schedules such as those available with D-Wave quantum annealers [50, 51].

Training is performed jointly on the parameters of the classical networks and on the parameters of the quantum device. The gradients of the latter parameters require estimation of the negative phase (a thermal expectation of the energy) in equation (16). At each gradient update, such expectations are computed using samples from the quantum annealer only and do not involve any classical Gibbs sampling such as PVD, or any classical post-processing of the samples obtained by the annealer. We typically trained our models for 2000 epochs and a batch size of 1000. Figure 2 shows a set of images generated by a VAE trained end-to-end using a D-Wave 2000Q system. The set of images, obtained by generating latent samples ζ with the quantum annealer and subsequently decoded as $\mathbf{x} \sim p_{\theta}(\mathbf{x}|\zeta)$, shows a good amount of global consistency and consistent statistical variety.

4.1. Validation of training

In this section we give more evidence that we have successfully exploited the Chimera-structured QBM prior in the latent space of our convolutional VAE. Validating the training of QCH generative models can be nontrivial and must be assessed carefully, especially when training uses a large amount of classical processing. We trained the deep networks of our model using GTX 1080 Ti GPUs and the quantum annealer is called only to estimate the negative phase. A principled validation strategy is thus to compare a model trained with quantum assistance to a simpler baseline for which the quantum hardware is not required. In our case, a

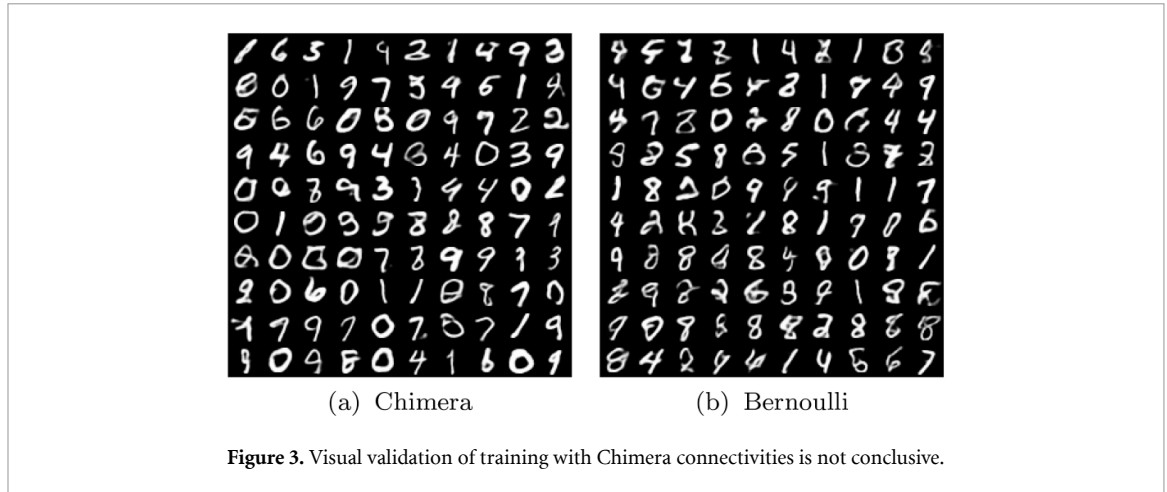


Figure 3. Visual validation of training with Chimera connectivities is not conclusive.

Table 1. Log-likelihood of convolutional VAEs trained with samples coming from either D-Wave 2000Q or PA. All models share the same encoding and decoding networks, but are trained independently for 2000 epochs on the MNIST dataset.

MNIST (dynamic binarization) LL		
Sampler	Chimera	Bernoulli
DW2000Q	-82.8 ± 0.2	-83.7 ± 0.2
PA	-82.8 ± 0.1	-84.2 ± 0.05

convenient baseline is a model that has the same classical networks but whose prior is trivial. As a trivial prior, we choose a set of independent Bernoulli variables, which is equivalent to a QBM with vanishing weights between latent units. For simplicity, in the following we refer to such prior as Bernoulli prior⁹.

For image processing, comparison between different generative models could be done qualitatively by visually inspecting the generated samples [60]. In our research however, visual comparison is inconclusive. In figure 3, for example, we compare samples generated by a trained model with Chimera (figure 3(a)) and Bernoulli (figure 3(b)) priors. Both models have been trained by evaluating the negative phase with a D-Wave 2000Q system. The images are also generated using samples coming from the annealers. Given our specific implementation, it is difficult to discern an improvement of visual quality when using a Chimera-structured rather than a Bernoulli prior. To overcome this difficulty, we seek to evaluate quantitatively the generative performance of our model by computing the ELBO defined in equation (3) or by estimating the log-likelihood via an importance sampling technique as described in reference [57]. These quantities are however not accessible when training with analog devices as samplers. In fact, while we assume that samples generated by quantum annealers are distributed according to the required Boltzmann distribution, we must treat quantum annealers as black-box samplers during testing and validation. In other words, the log-probabilities for the quantum generative process $\log p_{\theta}^{\text{DW}}(\zeta)$ must be assumed to be unknown. Therefore, we validate results by replacing the unknown hardware log-probabilities with those of an auxiliary CBM whose weights are given by the relations in equation (17)¹⁰:

$$\log p_{h,j}^{\text{DW}}(\zeta) \rightsquigarrow \log p_{b,w}^{\text{CBM}}(\zeta). \quad (18)$$

This approach can be more rigorously interpreted as validating a fully classical model in which we replace the quantum annealer by the auxiliary CBM defined by the relations above. We can consistently evaluate training using a fully classical auxiliary model since the quantum bias due to the presence of a transverse field is small enough that the estimation of the negative phase of the QBM [80] obtained with the quantum annealer is a good approximation of the negative phase of the auxiliary CBM.

In table 1 we report the LL of the auxiliary VAE with 288 latent unit on a Chimera connectivity trained with a D-Wave 2000Q quantum annealer. We compare it to a VAE with a CBM trained end-to-end with PA [1]. We also compare each model with its respective Bernoulli baseline. We have reported the mean and the standard error over 5 independent training runs.

⁹Both CBMs and QBMs with no weights are trivially equivalent to a product of independent Bernoulli variables.

¹⁰The computation of the log-probabilities requires the estimation of the partition function, which is done using annealed population sampling as in reference [1].

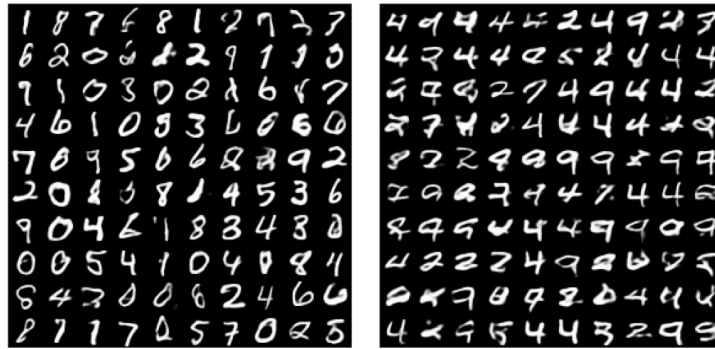


Figure 4. Left panel: Samples generated by D-Wave 2000Q using a fully trained model. Right panel: Samples generated by D-Wave 2000Q after setting the couplings of the annealer to zero.

Training on Chimera improves significantly the log-likelihood of the auxiliary model over the Bernoulli baseline. Moreover, the auxiliary models trained with quantum annealers achieved the same log-likelihood as the models trained with PA. Notice that each model and its baseline employed exactly the same amount of classical computational resources. Models trained with structured BMs achieve better performance by requiring thermal sampling, a computational task that can be offloaded to a quantum annealer.

In general we would like to use quantum annealers to sample from the trained generative model. As explained above, treating quantum annealers as black-boxes means we cannot quantitatively evaluate such a model. However, we argue that the log-likelihood of such a QVAE is likely very close to that of the auxiliary VAE. After all, the training assumes the hardware samples are distributed according to a Boltzmann distribution of the auxiliary restricted BM (RBM) and in the previous section we have shown that this assumption is accurate enough to correctly train the auxiliary CBM. We perform an additional visual analysis in figure 4. On the left panel we show a set of digits generated by sampling from a D-Wave 2000Q. On the right panel we use the same trained model but sample from a D-Wave 2000Q after setting its weights to zero. We see that while the annealer still generates plausible digits, it does not generate an ensemble of digits with the correct statistics (in the right panel of figure 4, digits 9 and 4 seem to dominate the scene). This gives evidence that D-Wave 2000Q quantum annealers sampled consistently, such that the classical networks were able to correctly learn the correlations between latent units existing due to the QBM with non-vanishing weights.

5. A path toward quantum advantage with VAE

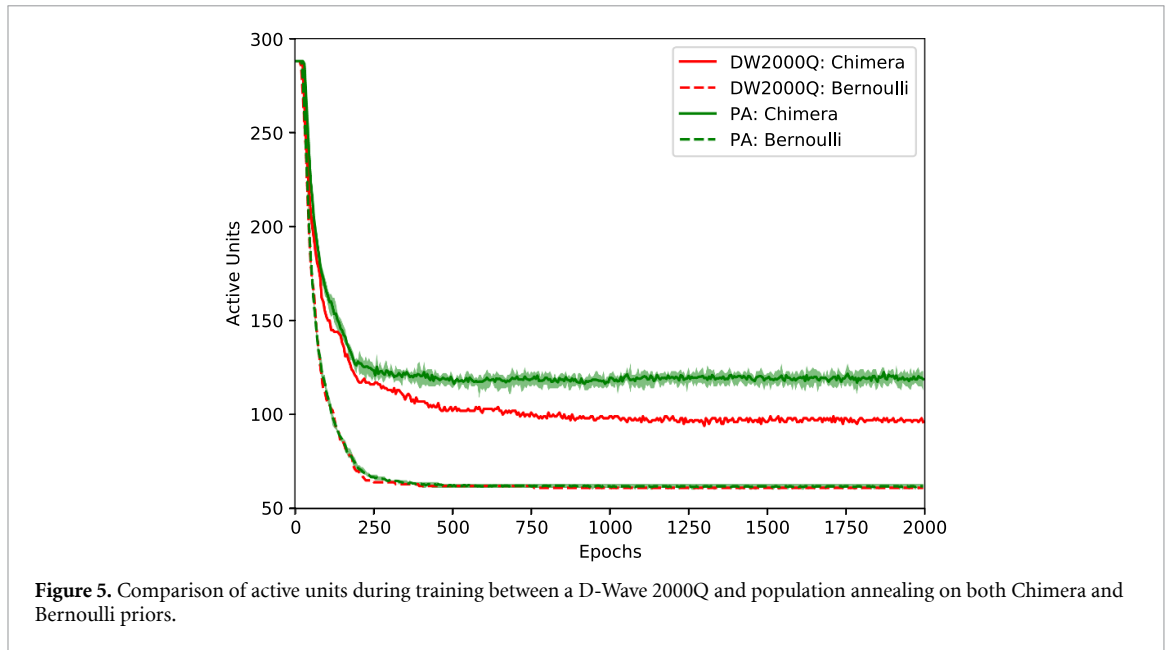
In the previous section we have shown that it is possible to use quantum annealers as Boltzmann samplers to train QVAE. In the experiments presented, we have settled on relatively small, Chimera-structured QBMs with 288 latent units. We found that larger QBMs did not appreciably improve performance of the overall VAE when training on the MNIST dataset. We will explain why this is the case in this section. Sampling from CBMs with a few hundred units can still be done classically with relative ease and the natural question that arises is whether we can obtain a quantum advantage in this hybrid setup.

A necessary condition to obtaining quantum advantage with our QCH setup is to engineer models that can exploit very large BMs with complex, multimodal distributions. Sampling from CBMs with complex landscapes is slow and inefficient for classical approaches such as Gibbs sampling, PCD, PA. Sampling from QBMs is especially computationally intensive since it requires Quantum Monte Carlo. Another important requirement is to be able to reliably sample from thermal states using quantum annealers with sufficiently low control errors.

In the next sections, we give evidence of the existence of a natural path toward obtaining quantum advantage by applying quantum annealing to generative modeling within the proposed VAE framework. Our discussion is based on a set of numerical experiments in which we have trained VAE with CBM, but it equally applies to QVAEs models with integrated latent-space QBMs.

5.1. Exploit large latent-space RBMs

Exploiting a larger number of latent units to improve the generative performance of a VAE is a popular and active research area. One known obstacle in achieving this is the loss function used for training. We rewrite



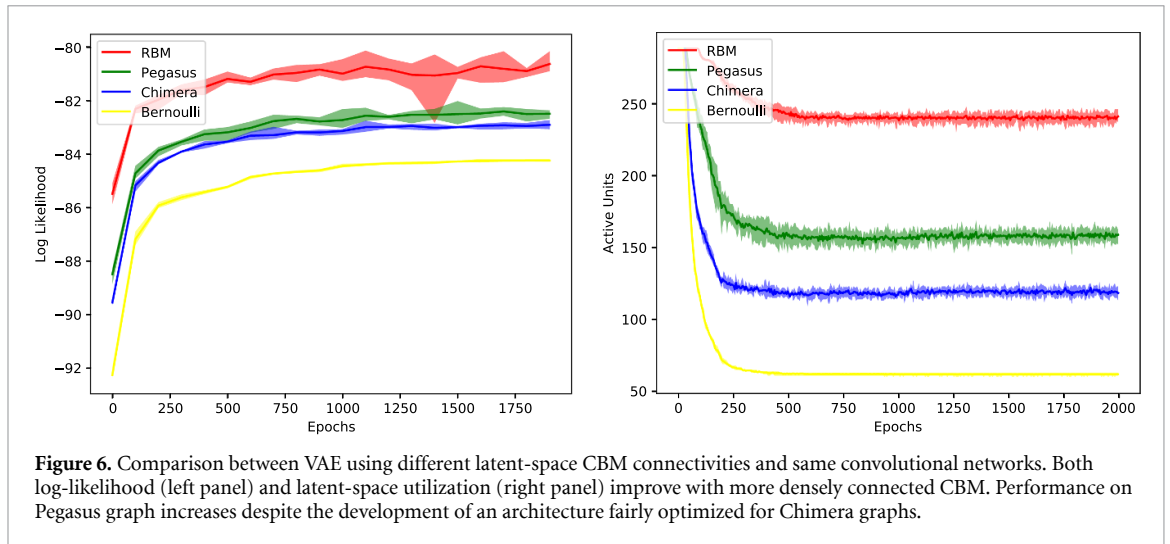
the ELBO here for convenience by highlighting its two terms:

$$\begin{aligned} \mathcal{L}(\mathbf{x}, \zeta, \phi) &= \underbrace{\mathbb{E}_{\zeta \sim q_{\phi}(\zeta|\mathbf{x})}[\log p_{\theta}(\mathbf{x}|\zeta)]}_{\text{autoencoding term}} + \\ &- \underbrace{D_{KL}(q_{\phi}(\zeta|\mathbf{x})||p_{\zeta}(\zeta))}_{\text{KL-regularization}}. \end{aligned} \quad (19)$$

The first term is sometimes called the ‘autoencoding’ term and can be thought of as a reconstruction error: an encoded latent configuration ζ is first sampled from the encoder $q_{\phi}(\zeta|\mathbf{x})$ and is subsequently decoded by $p_{\theta}(\mathbf{x}|\zeta)$. When both approximating posterior and marginal distributions are factorized distributions, this term can be also interpreted as a reconstruction error. Maximizing the ELBO results in maximizing this term, which tends to maximize the number of latent units used to prevent information loss during the encoding-decoding steps. The second term, the KL divergence between the approximating posterior and the prior, has the effect of a regularization term and it is sometimes also called KL regularization. Maximizing the ELBO results in minimizing the KL term, which pushes the approximating posterior close to the prior. This also means the approximating posterior depends less sharply on the inputs \mathbf{x} . In the case of factorized distributions, this usually means some latent units are conditionally independent from the input (‘inactive units’): $z_{inact} \sim q_{\theta}(z_{inact}|\mathbf{x}) = q_{\theta}(z_{inact})$.

The balance between the autoencoding and KL terms means using VAE can be efficient at lossy data compression by using the right number of latent units. However, the tension between KL and autoencoding terms results in an optimization challenge: during training the model is usually stuck in a local minimum with a suboptimal number of latent units. The main takeaway, for our purposes, is that the number of latent units effectively used is highly dependent on the model, the optimization technique and the training set. To exploit a larger BM, one thus has to work on all these elements. In figure 5 we show the number of active units during a training run of 2000 epochs when the models are trained with either PA or D-Wave 2000Q, with either a Chimera-structured BM or a Bernoulli prior. Lines (bold or dashed) are the means over 5 independent runs while light-color areas delimit the smallest and largest values among the 5 runs. To identify whether a latent unit is active or not, we compute the variance σ of the value of each unit z over the test set and we set a threshold of $\sigma > 0.01$ as definition of an active unit.

Figure 5 shows several key points that we will expand upon in the next sections. First, it shows the use of a KL annealing technique: the KL term is turned off at the beginning of the training and it is slowly (linearly) turned on within 200 epochs. The figure clearly shows the effect of the KL term in shutting down a large number of active units. Since the number of active units plateaus around a number much lower than 288, using a larger BM usually does not improve performance of our implementation on MNIST. It also shows that connectivity of the BM plays a major role in determining the number of active units, which is much higher with a Chimera-structured BM. Notice also that in the Bernoulli case, both samplers (PA and D-Wave 2000Q) train a model that uses a very similar number of latent units. However, when sampling is nontrivial



(as in a Chimera-structured BM) the model trained with the quantum annealer uses a number of units larger than the Bernoulli case, but smaller than the model trained with PA. This is a manifestation (which we will discuss later) of biased sampling with the quantum annealer: sampling quality is good enough to train the model (indeed we obtained a log-likelihood as good as that of the model trained with PA) but exploits a smaller number of latent units.

In the next three sections we discuss three important elements that would allow to build QCH VAEs that can effectively exploit large latent-space BMs. This is a necessary condition to search for quantum advantage in these models: speed-up and scale-up sampling from large BMs using quantum annealers rather than inefficient classical sampling techniques.

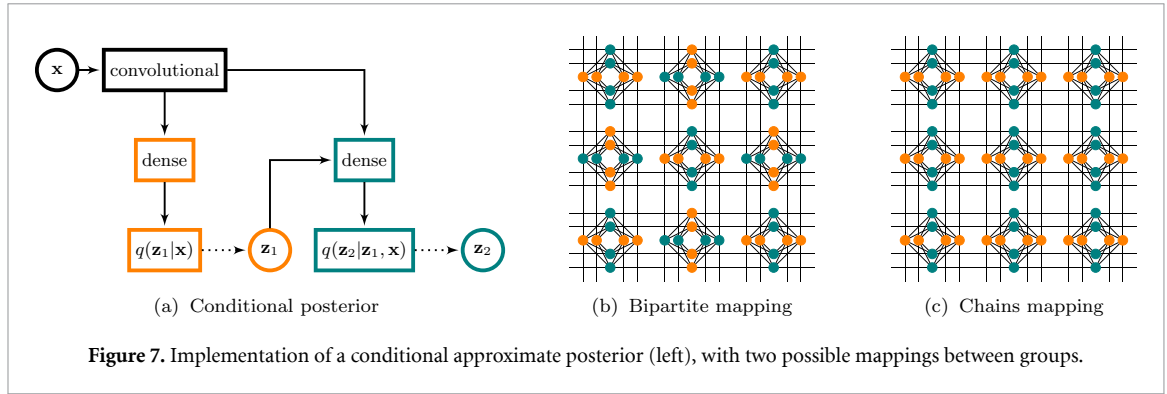
5.1.1. Denser connectivities

Connectivity of the BM in the latent space plays an important role in determining both performance of the generative model and number of active latent units. While these two elements are not directly related, we typically observe a correlation between them. In this section, we investigate in more detail the effects of implementing denser connectivities by performing numerical experiments in four different cases: Bernoulli prior and Chimera, Pegasus and fully-connected RBM. Together with Bernoulli and RBM, we pick the connectivities of currently available D-Wave 2000Q quantum annealers and D-Wave's next-generation Pegasus architecture [81] (see also appendix D).

In figure 6 we compare the log-likelihood and the number of active units along training runs obtained using the same classical networks but using a different connectivity for the latent-space RBM prior. At each step during the training run, we employed roughly the same amount of classical resources for back-propagation. Unsurprisingly, using a more capable (dense) BM results in better generative performance (log-likelihood) of the model (left panel). It also results in using a larger number of latent units (right panel). Notice how Bernoulli and Chimera priors effectively use a number of latent units well below 150, while Pegasus and fully connected use well above 150. Because of this, we could not improve generative performance of Bernoulli and Chimera models by just using larger graphs, at least when training on MNIST. On the other hand, using larger Pegasus and fully connected RBMs would likely have improved the log-likelihood of the model. In figure 6 we show results on the same number of latent units (288) for proper comparison.

Working with VAEs allows us to easily take advantage of new connectivities without having to implement a new convolutional VAE. In fact, our model has been fairly optimized to improve performance on Chimera graphs (see the next section). Despite this architecture-specific optimization, just using the denser Pegasus graphs improve performance of the model. Moreover, the flexibility of the VAE hybrid approach allows us to easily adapt the implementation to the slightly different working graphs of different processors with different active/inactive qubits. By using the same implementation, during training the model naturally learns to deactivate latent units corresponding to uncalibrated qubits. We have indeed seamlessly used the same model to train on different D-Wave 2000Q processors, as well as using different groups of qubits within the same processor. In no case was a hard-coded connectivity (which would change for each processor) necessary.

The results of figure 6 show that developing quantum annealers with denser connectivities (such as Pegasus) naturally leads to exploiting larger latent-space BMs, possibly getting us closer to a regime where quantum advantage is possible.



5.1.2. Hardware-specific optimization of classical networks

In our implementation, we have used fairly conventional convolutional NN. As we will discuss in this section, we have implemented only one specific architecture-dependent element in the encoder network that turned out to be very effective at improving performance on the Chimera graph. In general, we believe more elaborate, hardware-specific implementations of both the approximating posterior $q_\phi(\zeta|\mathbf{x})$ and the marginal distribution $p_\theta(\mathbf{x}|\zeta)$ will significantly improve performance of VAE trained with analog devices with quasi two-dimensional connectivities. This is not just a problem of building a model with more capacity. As we discussed in the case of latent-unit use, it is also a problem of optimizing the model hyperparameters. When working with sparsely connected BMs, it is easier for the KL term to push both approximating posterior and BM prior to a local minimum of the loss function in which they are both trivial. Developing hardware-specific hybrid models will thus also aim at reaching local minima during training in which the BM prior is as expressive as possible, exploiting the largest amount of correlations between latent units.

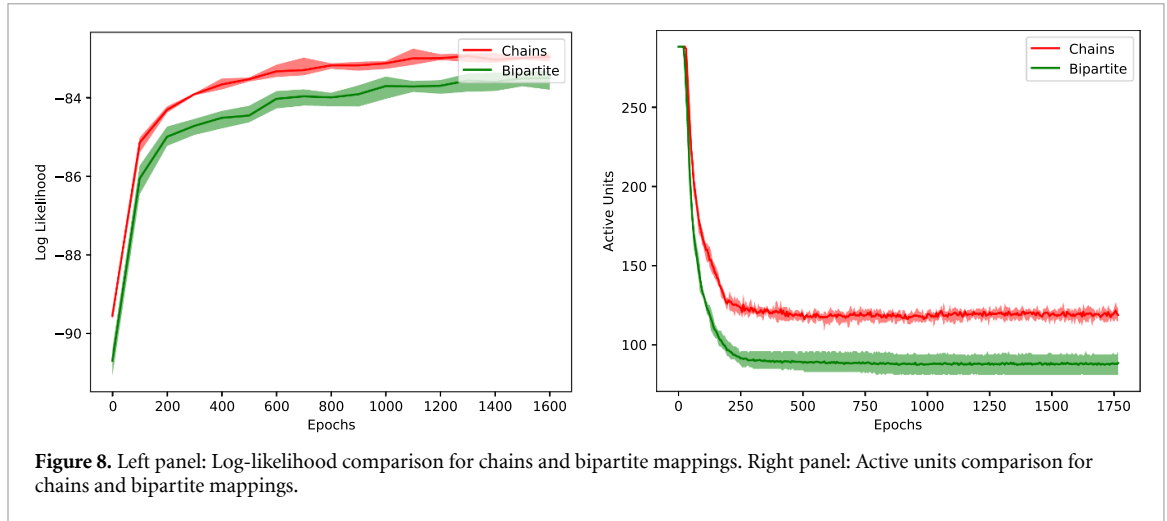
In the context of hybrid VAE models trained with quantum annealers, the considerations above could replace more standard mapping techniques used in the quantum annealing community such as minor embedding [82, 83] and majority voting. The latter techniques typically require a hard-coded specification of the hardware connectivity of each processor, which makes adapting the code to different processors cumbersome. Our perspective in the contest of hybrid generative modeling is to work by adapting classical networks using a high-level specification of the connectivity and letting the model, through stochastic gradient descent, learn the details of the connectivity of each processor (such as the locations of uncalibrated qubits).

We now discuss an example of how the classical networks can be optimized for a given architecture (in our case the Chimera graph). A common technique to implement a more expressive approximating posterior $q_\phi(\zeta|\mathbf{x})$ (with a tighter variational bound) is to introduce conditional relationships among latent units, also called hierarchies. In the case of two hierarchies, we first define an approximating posterior for a subset of latent units ζ_1 and sample from it, then define a second approximating posterior for the remaining latent units ζ_2 that depends conditionally on both the input data and the sampled values of the first group of latent variables:

$$\zeta_1 \sim q_{1,\phi}(\zeta_1|\mathbf{x}), \quad \zeta_2 \sim q_{2,\phi}(\zeta_2|\zeta_1, \mathbf{x}). \quad (20)$$

The schematic of our implementation is shown in figure 7(a). We notice that models with a large number of hierarchies (possibly as large as the number of latent units) are possible. Such models, also referred to as autoregressive models [84], are very powerful but have inefficient inference, since sampling must be performed sequentially and cannot be parallelized on modern GPU hardware.

The two hierarchical groups (ζ_1, ζ_2) , as well as the physical qubits on the Chimera connectivity, are not equivalent. We can thus build different models by simply choosing the mapping between the two hierarchical groups and an arbitrary bipartition of the physical qubits. Notice that training and deploying each of these models will involve exactly the same amount of classical and quantum computational resources. In our experiments we consider two possible mappings of the two hierarchical groups onto the physical qubits of the Chimera graph, which we call ‘Bipartite’ and ‘Chains’. The Bipartite mapping corresponds to the bipartite structure of the Chimera graph (see figure 7(b)). The Chains mapping corresponds to the vertical and horizontal physical layout of qubits of the Chimera architecture (see figure 7(c)). The identification of vertical and horizontal chains of qubits is commonly used to perform a minor embedding of an RBM on the Chimera graph. We stress again, however, that we never perform any minor embedding and we always sample from the native Chimera graph in all cases.



Comparative results of the two mappings, obtained with classical sampling, are shown in figure 8. In the left panel we see that there is a sizeable difference in generative performance between the two mappings, with the Chains mapping performing remarkably better. This better performance is also reflected in the much higher number of latent units exploited by the Chains mapping (see right panel of figure 8). An intuitive explanation of the results above can be given as follows. Let us first write the KL term as:

$$D_{KL}(q_{\phi}(\zeta|\mathbf{x})||p_{\theta}(\zeta)) = D_{KL}(q_{1,\phi}(\zeta_1|\mathbf{x})||p_{\theta}(\zeta_1)) + D_{KL}(q_{2,\phi}(\zeta_2|\zeta_1,\mathbf{x})||p_{\theta}(\zeta_2|\zeta_1)). \quad (21)$$

In the Bipartite mapping, the conditional $p_{\theta}(\zeta_2|\zeta_1)$ has a simple form that can be computed analytically due to the bipartite structure of the Chimera graph. During training, it is very easy for the model to use the capacity of $q_{2,\phi}(\zeta_2|\zeta_1,\mathbf{x})$ to match the simple prior marginal $p_{\theta}(\zeta_2|\zeta_1)$ and be independent from \mathbf{x} . As a consequence, a large portion of the representational capacity of the approximating posterior is wasted in representing the simple marginal $p_{\theta}(\zeta_2|\zeta_1)$. In the Chains mapping, in contrast, the marginal $p_{\theta}(\zeta_2|\zeta_1)$ is nontrivial and the approximating posterior $q_{2,\phi}(\zeta_2|\zeta_1,\mathbf{x})$ has more difficulties in matching it and decoupling \mathbf{x} . As a consequence, the model ends up using more efficiently all the variational parameters ϕ . This is another manifestation of the optimization challenge present with VAE models mentioned before, which in this case it is exploited to find better local minima of the loss function.

In this section we have shown how a simple architecture-aware modification of the encoder network allows us to train better models and to exploit a given architecture more efficiently. We expect that architecture-aware model-engineering will be crucial to fully exploit large physical connectivities in the latent space of VAE.

5.1.3. Training on larger datasets

Implementing more highly connected BMs and developing classical encoders and decoders tailored to a given connectivity can only go so far in helping to exploit larger latent spaces. Together with other techniques such as KL-term anneal, the ideas mentioned in the previous two sections help reduce the pressure of the KL term to reach suboptimal local minima. In essence, VAE are also efficient lossy encoders. An alternative direction to increase latent space utilization is thus to train on more complex datasets. By doing so, a larger number of latent units is necessary to store enough information such that the reconstruction term (first term in equation (19)) is large enough.

We give numerical evidence of the intuition above by training the same VAE models used in the previous sections on the Fashion MNIST (FMNIST) dataset. A set of images from the FMNIST dataset is shown in the left panel of figure 9. FMNIST is the same size as MNIST (60 000, 28×28 images) and has the same number of classes. However, its images are more complex with more fine details, including grey-scale features that are important for correct image classification (whereas MNIST digits are substantially black and white). In the right panel of figure 9, we train the same models on MNIST (shaded) and FMNIST (solid) and compare the number of active units during training. We see that, apart from the case with fully connected RBMs, all other models use a substantially larger number of latent units.

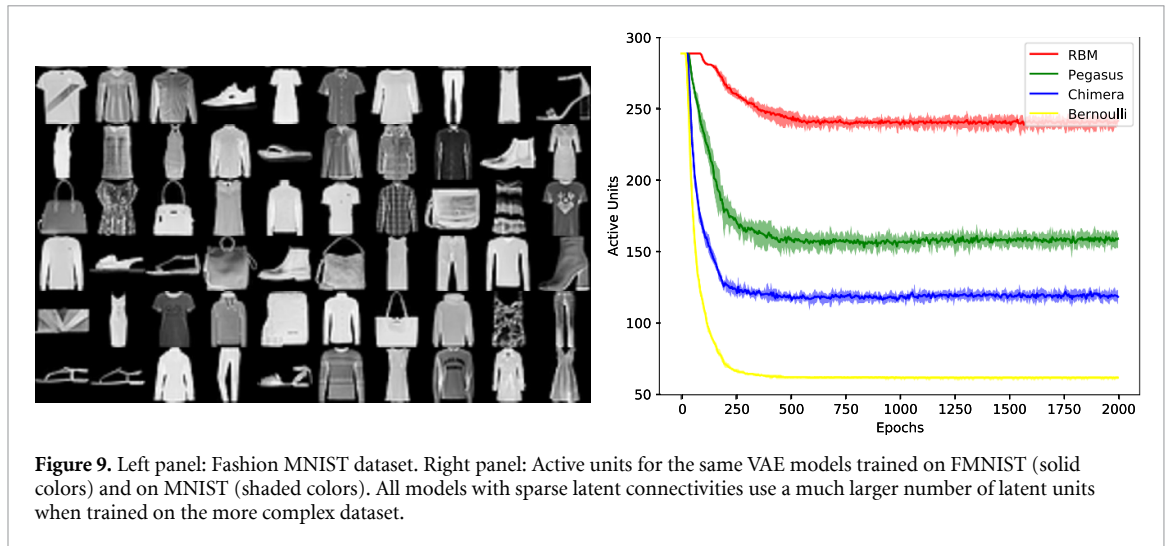


Figure 9. Left panel: Fashion MNIST dataset. Right panel: Active units for the same VAE models trained on FMNIST (solid colors) and on MNIST (shaded colors). All models with sparse latent connectivities use a much larger number of latent units when trained on the more complex dataset.

5.2. Multi-modality of latent-space RBMs

Exploiting large latent-space BMs is a necessary condition to eventually achieve quantum advantage when sampling with quantum annealers. This condition is however not sufficient. The typical computational bottleneck in training a BM is due to the appearance of well-defined modes. These modes make classical sampling techniques inefficient and slow-mixing, resulting in highly correlated samples used both during training and generation. While making sampling harder, the development of multi-modal distributions is actually an appealing property of BMs, since it allows such models to represent complex and powerful probability distributions. The idea behind searching for quantum advantage in training BM with quantum annealers is, indeed, to exploit quantum resources (such as tunneling) to more efficiently mix between different modes in the landscape defined by the BM.

When trained on visible data, a BM naturally develops complex landscapes to match the complexity of the statistical relationship present in the training data. However, while BMs trained on latent representations can potentially develop well-defined modes, they do not necessarily do so. In fact, one of the capabilities of generative models with latent variables is finding a set of statistically independent latent features [85]. This is typically enforced during model building by using trivial priors such as the product of independent Gaussian (for continuous latent units) or the product of independent Bernoulli (for discrete latent spaces). Even when the prior is potentially complex and trainable, as a BM, the presence of the KL term can push the model during training into local minima in which the trained BM develops a trivial landscape.

In this section we give evidence that BMs trained in the latent space of a VAE model do indeed develop a nontrivial landscape with well-defined modes. As we have shown in section 5.1.1 (see figure 6), BMs with denser connectivities naturally lead to better performing VAEs. This is an indirect indication that we are indeed exploiting the additional capacity and expressivity of more connected BMs. In this section we give more explicit evidence of this. In figure 10, we generate a sequence of images via block Gibbs sampling. The top left image is generated by picking a latent configuration ζ out of a uniform distribution over all configurations. This latent sample is then sent through the decoder. Going from left-to-right, top-to-bottom, each subsequent image is obtained after updating all latent units with a sequence of block Gibbs updates (one for Bernoulli, two for the bipartite connectivities Chimera and RBM, four for the quadripartite Pegasus connectivity). As expected, in the Bernoulli case (figure 10(a)), each update results in uncorrelated samples. The Chimera connectivity (figure 10(b)) is able to develop weakly correlated samples, as shown by short sequences of similar images. Correlated samples with well-defined modes are more clearly visible with the Pegasus connectivity ((figure 10(c))). Finally, we confirm that increasing the connectivity up to a fully connected RBM (figure 10(d)) results in long sequences of correlated samples and related to the deep valleys of the RBM energy landscape.

The results shown in figure 10 show that BMs trained as priors of generative models with latent variables naturally learn multi-modal, nontrivial probability distributions. These distributions are expressive, making the whole VAE more expressive, while at the same time developing the same types of computational bottlenecks that make classical sampling algorithms inefficient. This paves the way to effectively use quantum annealers as means to more scalable quantum-assisted sampling, enabling us to sample from BMs of sizes and complexity that are infeasible with classical methods.

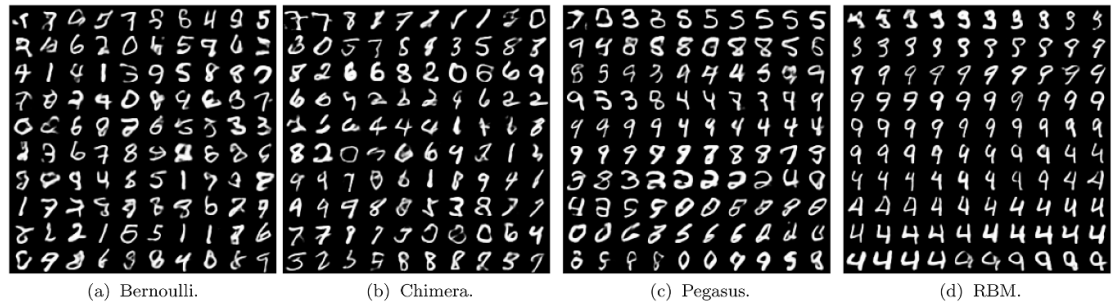


Figure 10. Block Gibbs sampling with different connectivities. Going from left to right, denser connectivities result in more well-defined modes developed in the trained RBM. Especially in the case of Pegasus and an RBM, for example, it is clearly visible how the block Gibbs chain is trapped in a typical basin of the landscape for MNIST connected to the digits 4 and 9.

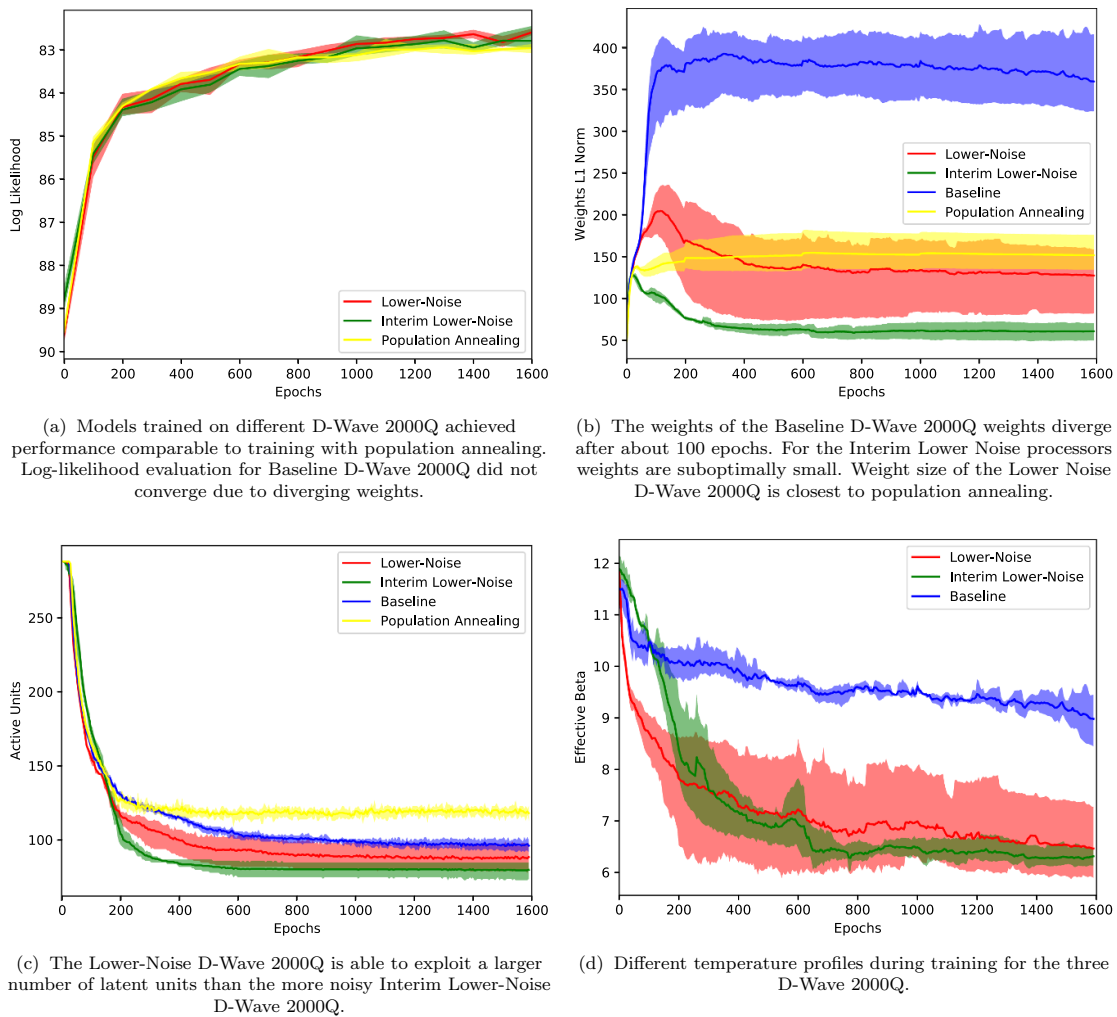


Figure 11. Training on different quantum annealers with different noise profiles.

5.3. Robustness to noise and control errors

Using quantum annealers to train large BMs directly on complex data remains challenging. Apart from the unsatisfying performance of using BMs with quasi two-dimensional connectivities on visible data, a major difficulty is biased sampling (and thus inaccurate gradients) obtained with quantum annealers. There are two main sources of bias: control errors and imperfect or incomplete thermalization at the freezing point. While the latter can be improved with appropriate pause-and-ramp annealing schedules, the former can only be improved with technological advancements. Despite these known difficulties, we have shown in the previous sections we have successfully trained large BM (hundreds of units) in the latent space of a VAE

solely using samples coming from a D-Wave 2000Q, without using any hard-coded pre or post-processing to the raw data obtained from the annealer.

We interpret our positive results as an indication that training BMs with our setup is relatively robust to noise and control errors. In fact, we can interpret both the encoder and decoder as powerful tools to pre- and post-process data to be sent to the quantum annealer. Using stochastic gradient descent, we train the encoder and decoder to generate a set of latent features that are more easily modeled by the latent-space BM. For example, real images might have strongly correlated, sharp features (such as regions with black or white pixels), which require large weights to be modeled correctly. A precise implementation of such large weights might be challenging for analog devices with finite range such as quantum annealers. Additionally, both encoders and decoders might be able to learn and correct, or at least reduce the effects of, systematically biased sampling.

To investigate the role of noise and control errors in determining sampling quality and performance of the trained models, we perform a set of comparative experiments in which we train the same model on MNIST dataset, using samples coming from three D-Wave 2000Q with different noise profiles. The Baseline and Lower-Noise D-Wave 2000Q are both publicly available on D-Wave's LeapTM cloud service. We have also included an Interim Lower-Noise processor with an intermediate noise profile that is internally available at D-Wave.

Results are shown in figure 11. In figure 11(a) we report the log-likelihood during training. Models trained on the Lower and Interim Lower-Noise D-Wave 2000Q achieved performance comparable to training with PA. The evaluation of the log-likelihood for the Baseline D-Wave 2000Q diverged due to diverging weights, as can be seen in figure 11(b). The weights of the Baseline processor start diverging after about 100 epochs. The Interim Lower-Noise processor shows an opposite behavior, with weights getting small and plateauing after about 500 epochs. For the Lower-Noise processor, the L1 of the weights plateaus at a value that is closer to that obtained with 'noiseless' PA. Notice that a consistent comparison in figure 11(b) we have reported the weights W rescaled by the effective temperature (see equation (17) and not the 'bare' annealing values J . Figure 11(b) highlights the remarkably different response of three different quantum annealers to our model. Despite such differences, the performance of our hybrid implementation is robust (as shown in figure 11(a)) and does not require any hardware-specific adaptations or fine-tuning. Only while training with the Baseline D-Wave 2000Q, we needed a more aggressive clipping (that is restricting the weights and biases to have narrower range than the maximum allowed) to achieve similar performance and converged log-likelihood estimation.

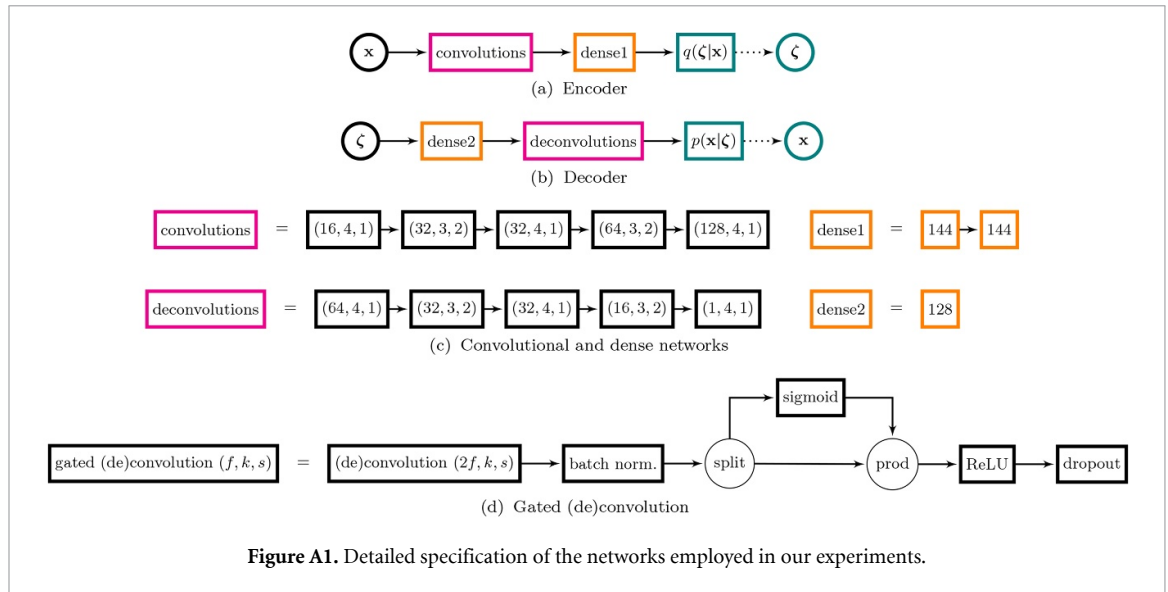
Noise and control errors also manifest in a less efficient use of the latent space, as seen in figure 11(c). All models trained with D-Wave 2000Q use fewer active units than PA. Since the estimate of the log-likelihood of the models trained with the Baseline processor did not converge, we focus on the comparison between the Interim and Lower Noise processor: the latter can exploit a larger number of latent units. We finally show in figure 11(d) the profile for the extracted effective temperature during training, which is remarkably different for the three D-Wave 2000Q. The results shown in figure 11(d) underlines the importance of developing advanced annealing techniques to stabilize temperature during training.

In this section we have demonstrated the robustness to noise of our implementation, as highlighted in figure 11(a). At the same time, figure 11(c) shows an important effect of noise, which is to make it harder for our hybrid model to exploit the optimal number of latent units. We thus anticipate that exploiting large BMs (with thousands of units, eventually) following the directions indicated in the previous sections must be accompanied by continued efforts in reducing sampling bias due to noise and control errors of future-generation quantum annealing devices.

6. Conclusions

In this work, we have demonstrated the use of quantum annealers as Boltzmann samplers to estimate the negative phase of BMs placed in the latent space of deep convolutional VAE. This setup allows for the construction of QCH generative models that can be scaled to large, realistic datasets. We have mostly experimented with MNIST, a common testbed dataset which includes 60,000, 28×28 binarized handwritten digits to achieve a log-likelihood of about -82.2 ± 0.2 nats, which compares favorably to state-of-the-art achieved with autoregressive models (-78.5 nats (natural unit of information)). In addition to demonstrating scalability and performance, we have discussed several other features of our hybrid approach.

First, we are able to use quantum annealers as 'native samplers', that is, samplers from their native graph: we do not use any hard-coded encoding-decoding scheme such as minor embedding or majority vote. Arguably this is one of the most effective ways to exploit the computational capabilities of quantum annealers. The encoding and decoding process is indeed efficiently performed by deep convolutional networks, which are trained to extract relevant feature via stochastic gradient descent. As we have shown, this



approach is particularly flexible, since it naturally adapts to different connectivities and arbitrary working graphs.

Second, by successfully training the same model on three quantum annealers with different noise profiles, we have shown that our implementation is fairly robust to noise and control errors. Indeed, the deep convolutional networks can be seen as learned pre- and post-processing steps that regularize both the visible data and the effects of noise. A key reason of the success of our implementation is indeed the fact that the weights and biases as implemented on the quantum annealers rarely grow (during training) beyond their allowed range, even with minimal or no regularization. This result is to be contrasted to training BMs directly on visible data, for which weights are typically much larger and regularization is critical to avoid overfitting. The latter case is much more challenging for analog devices with limited allowed range.

The QCH models we have considered in this work employ a large amount of classical computing power performed on modern GPUs. The computational task that we offloaded to the quantum annealer (sampling from the latent-space BMs) can still be performed classically at a fraction of the overall computational cost. To achieve any form of quantum advantage in this framework, we need to offload generative capacity to the prior, by exploiting large BMs capable of representing complex probability distributions from which classical sampling becomes too expensive. We have provided evidence that this path to quantum advantage is possible by deploying annealers with denser connectivities and lower noise, engineering classical neural nets that better exploit physical connectivities and by working with more complex datasets. All these improvements seem achievable in the near future and represent possible interesting lines of research that we leave for future work.

Acknowledgments

The authors would like to thank Arash Vahdat and Jason T Rolfe for useful discussions during the preparation of this work, Kelly Boothby for providing the figures for Pegasus and Chimera architectures and Fiona Hanington for editing the manuscript. For this work L B was supported by a grant from ‘Fondazione Angelo Della Riccia’. W V is grateful for support from NASA Ames Research Center, the Office of the Director of National Intelligence (ODNI) and the Intelligence Advanced Research Projects Activity (IARPA), via IAA 145 483. We also appreciate support from the AFRL Information Directorate under grant F4HBKC4162G001. 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 IARPA, AFRL, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purpose notwithstanding any copyright annotation thereon.

Appendix A. Convolutional VAE

The VAE employed in our experiments is schematically represented in figure A1. Both approximating posterior $q(\zeta|\mathbf{x})$ (encoder) and marginal $p(\mathbf{x}|\zeta)$ (decoder) are constructed using deep convolutional

networks, see figures A1(a) and A1(b). Although not technically necessary, we use (approximately) mirror implementations for encoder and decoder. In the encoder, down-sampling is achieved by employing strided convolutions, while in the decoder up-sampling is similarly obtained with strided deconvolutions. The last (first) layer of the encoder (decoder) network is a dense network with two (one) layers (see figure A1(c)). In the case of the encoder, a hierarchical (conditional) relationship among variables is implemented as described in figure 7(a). The convolutional networks are implemented as a simple sequence of five gated convolutions, whose detailed implementation is given in figure A1(c). Notice the use of batch normalization and dropout. The latter was only used in the decoder, to prevent over-fitting, with a drop-rate of 0.2.

We trained our models using batches of size 100 and the Adam optimizer with an initial learning rate of $3e^{-3}$, exponentially decaying to a minimum learning rate of $1e^{-4}$ after 1800 epochs. The temperature parameter τ defined in equation (10) for the Gumbel trick is typically annealed from large to small values. We however did not find a real advantage in doing so and we fixed the parameter to the low value $\tau = 1/7$ throughout the training. To improve training and avoid collapse of the approximating posterior to trivial local minima, we have linearly annealed the KL term from zero to its full value within 200 epochs.

In general we have trained our models using an importance-weighted estimate of the likelihood. As first described in reference [57], a K -sample weighting estimate of the log-likelihood can be written as:

$$\mathcal{L}_K = \mathbb{E}_{\zeta_1, \dots, \zeta \sim q_\phi(\zeta|\mathbf{x})} \left[\log \frac{1}{K} \sum_{k=1}^K \frac{p_\theta(\zeta, \mathbf{x})}{q_\phi(\zeta|\mathbf{x})} \right], \quad (\text{A1})$$

which is equivalent to the ELBO defined in equation (3) for $K = 1$ and converges to the exact log-likelihood for $K \rightarrow \infty$. We also found useful, to reduce the variance of the gradients of \mathcal{L}_K , to use a multi sample evaluation of the gradients per data point \mathbf{x} . In other words, we can use the following for training:

$$\mathcal{L}_{K,D} = \mathbb{E}_{\zeta_{k,d} \sim q_\phi(\zeta|\mathbf{x})} \left[\log \frac{1}{K} \sum_{k=1}^K \frac{p_\theta(\zeta_{k,d}, \mathbf{x})}{q_\phi(\zeta_{k,d}|\mathbf{x})} \right], \quad (\text{A2})$$

with $k = 1, \dots, K$ and $d = 1, \dots, D$. Notice that equation (A2) requires sampling KD latent configurations per data-point \mathbf{x} . This can be parallelized on GPU by effectively working, in our case, with batches of size $KD \times 100$. In our experiments we found it effective to have $KD = 8$ and to change the relative values of K and D while keeping their product constant. Every 200 epochs we changed their value as follows:

$(K, D) = (1, 8) \rightarrow (2, 4) \rightarrow (2, 4) \rightarrow (4, 2) \rightarrow (4, 2) \rightarrow (8, 1)$ and kept it constant afterwards. While a larger K results in a tighter variational lower bound, it also makes harder training the approximating posterior, the reason being that in the limit of large K the bound \mathcal{L}_K does not depend on $q_\phi(\zeta|\mathbf{x})$. We found this $K \leftrightarrow D$ anneal to be more efficient at both training the approximating posterior and training on a tighter bound to the log-likelihood. We used the same technique, with $K = 1000, D = 1$ as the estimate of the log-likelihood.

Appendix B. Sample collection with D-Wave 2000Q

To estimate the negative phase with D-Wave annealers, we used 1000 samples obtained with independent annealing runs. For each gradient evaluation, we performed 5 random spin-reversal transformations and collected 200 samples each time. We used a forward annealing schedule with a $1 \mu\text{s}$ forward anneal up to $s = 0.5$, where we paused for $10 \mu\text{s}$. After the pause we performed a 10 ns quench to finish the anneal. After a bit of experimentation, we found this particular annealing schedule to slightly improve training, although a simple forward annealing without pause-and-quench also worked well. We did not perform any post-processing of the samples, which we used as-is to compute the negative phase.

An important question for future works is whether a more careful choice of the annealing schedule, possibly with longer pauses, can stabilize the effective temperature at which samples are drawn from the hardware. We discuss the importance of this aspect in the next section.

Appendix C. Estimating β_{eff}^* during training

As noticed in reference [46], training a BM with a quantum annealer does not necessarily require the knowledge of the effective sampling temperature introduced in equation (17). Indeed, β_{eff}^* can be absorbed into the learning rate γ :

$$\begin{aligned} \partial \log \tilde{p}_{b,W}(\zeta) &= \gamma \left(-\partial \mathcal{H}_{b,W}(\zeta) + \mathbb{E}_{\bar{\zeta} \sim p_{b,W}} [\partial \mathcal{H}_{b,W}(\bar{\zeta})] \right) \\ &= \gamma' \left(-\partial \mathcal{H}_{h,J}(\zeta) + \mathbb{E}_{\bar{\zeta} \sim p_{b,W}} [\partial \mathcal{H}_{h,J}(\bar{\zeta})] \right) \end{aligned}$$

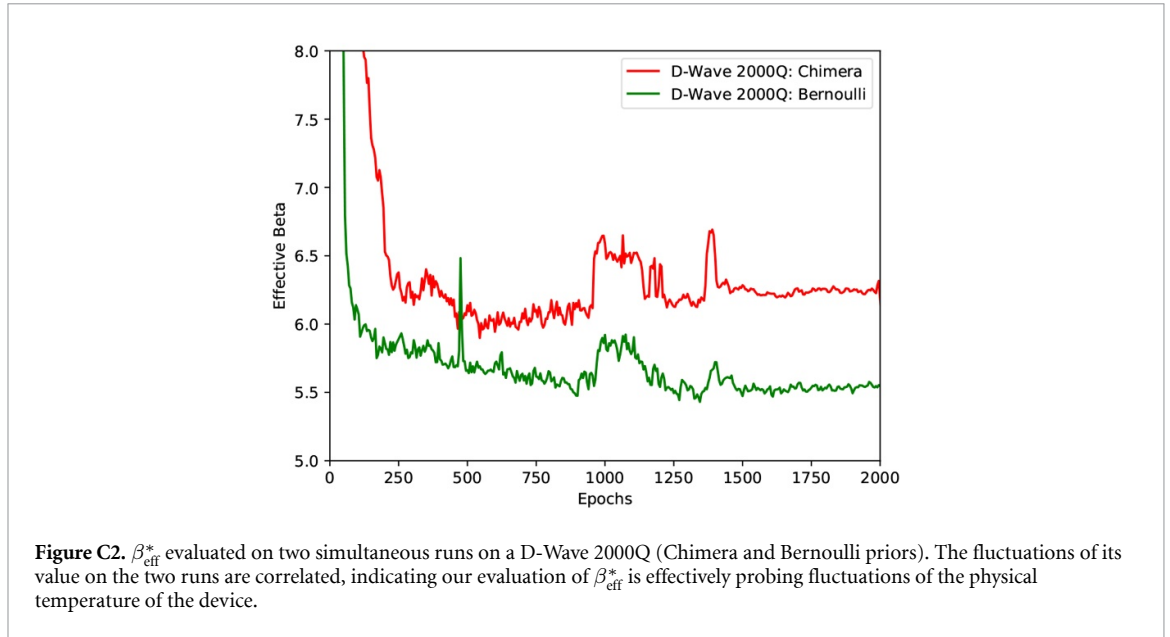


Figure C2. β_{eff}^* evaluated on two simultaneous runs on a D-Wave 2000Q (Chimera and Bernoulli priors). The fluctuations of its value on the two runs are correlated, indicating our evaluation of β_{eff}^* is effectively probing fluctuations of the physical temperature of the device.

$$\gamma' = \gamma\beta_{\text{eff}}^* \tag{C3}$$

While this is still true for the gradients of the parameters of the RBM placed in the latent space of a VAE, correctly evaluating the gradients of the inference parameters ϕ requires knowledge of β_{eff}^* . To see this it suffices to note that the samples in the positive phase depends on the inference parameters through the reparameterization trick. During training: $\zeta \rightarrow \zeta(\phi, \rho)$. Tracking where these gradients come from, we have:

$$\begin{aligned} \gamma\partial_{\phi}(\text{ELBO}) &= -\gamma\partial_{\phi} \log q_{\phi}(\zeta(\phi, \rho)|\mathbf{x}) + \\ &\quad -\gamma\beta_{\text{eff}}^*\partial_{\phi} \mathcal{H}_{h,j}(\zeta(\phi, \rho)), \end{aligned} \tag{C4}$$

so that the correct evaluation of the gradients with respect to the inference parameters requires the (approximate) knowledge of the effective temperature β_{eff}^* .

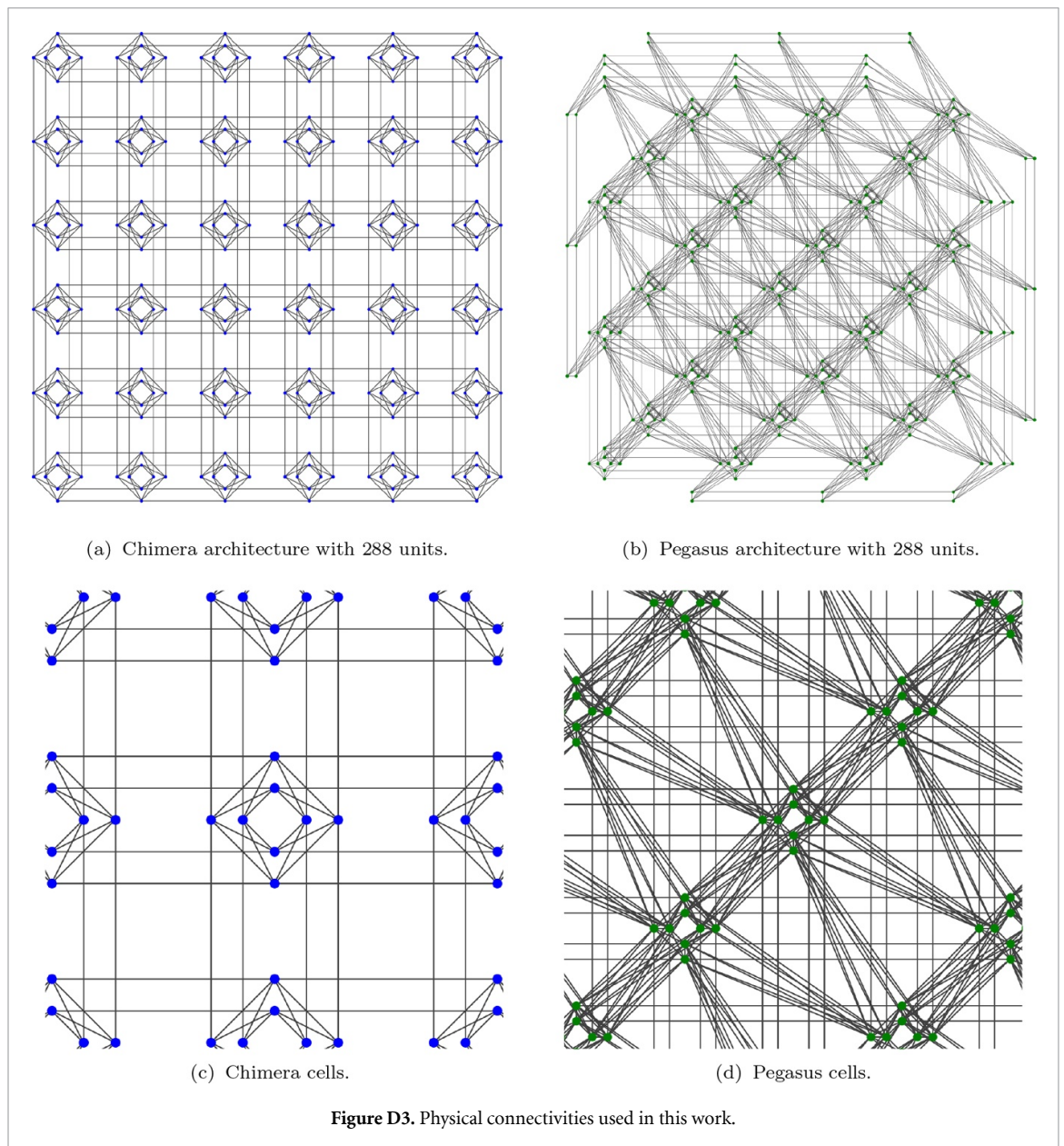
In our experiments, we have performed a real-time estimation of β_{eff}^* , which we used as in the equation above to correctly estimate the gradients for the inference parameters. To do so, we employed an auxiliary BM that we trained in parallel with the VAE on the negative samples obtained by the quantum annealers. The parameters of the BM are shared according to equation (17), with the only trainable parameter being β_{eff}^* . In other words, we update β_{eff}^* as follows:

$$\begin{aligned} \beta_{\text{eff}}^* &\rightarrow \beta_{\text{eff}}^* + \\ &\quad + \gamma \left(-\mathbb{E}_{\zeta \sim P_{h,j}^{\text{HW}}}[\mathcal{H}_{h,j}(\zeta)] + \mathbb{E}_{\zeta \sim P_{b,w}^{\text{BM}}}[\mathcal{H}_{h,j}(\zeta)] \right), \end{aligned} \tag{C5}$$

where the first expectation is evaluated with the hardware samples; the second, with thermal samples from the auxiliary BM (obtained with PA).

In figure C2 we show the value of β_{eff}^* estimated with the method above on two simultaneous runs on a D-Wave 2000Q. Its value typically drops while the KL term is annealed (200 epochs in our experiments) and subsequently stabilizes. Some fluctuations are correlated among independent runs and are related to real fluctuations of the physical temperature of the device.

We have noticed that, due the use of the KL anneal and the presence of a non-negligible change in β_{eff}^* during training, using a time-dependent evaluation of the effective temperature is important to stabilize training. While computing a single gradient as in equation (C5) is much more robust than training all the weights of a comparable BM, the method is not completely scalable and requires thermal sampling with classical algorithms. It will be critical, in future works, to implement training procedures with stable values of β_{eff}^* , which could be kept constant, using values predetermined by previous experiments or simply treated as a hyper parameter whose value must be appropriately fixed. Eventually, the use of more advanced annealing schedules, with longer pauses and more carefully chosen pause-points, should allow a direct connection between β_{eff}^* and the physical temperature of the annealer, thus removing the necessity of learning β_{eff}^* from experiments.



Appendix D. Chimera and Pegasus connectivities

In figure D3 we show the Chimera and Pegasus connectivities on 288 qubits used in all the experiments performed in this work. The Chimera graph (figure D3(a)) is a bipartite, two-dimensional tiling of a unit cell (figure D3(c)) with 8 qubits. The Pegasus graph (figure D3(b)) is a quadri-partite, two-dimensional tiling of a unit cell (figure D3(d)) with 8 qubits [81].

ORCID iD

Lorenzo Buffoni  <https://orcid.org/0000-0002-8191-3375>

References

- [1] Khoshaman A, Vinci W, Denis B, Andriyash E and Amin M H 2019 *Quantum Sci. Technol.* **4** 014001 <http://stacks.iop.org/2058-9565/4/i=1/a=014001>.
- [2] Rosenblatt F 1958 *Psychol. Rev.* **65** 386
- [3] Rumelhart D E, Hinton G E and Williams R J *et al* 1988 *Cognitive Modeling* **5** 1
- [4] Hinton G E, Osindero S and Teh Y-W 2006 *Neural Comput.* **18** 1527
- [5] Bengio Y, Lamblin P, Popovici D and Larochelle H 2007 *Advances in Neural Information Processing Systems* pp 153–60
- [6] LeCun Y, Bengio Y and Hinton G 2015 *Nature* **521** 436
- [7] Goodfellow I, Bengio Y and Courville A 2016 www.deeplearningbook.org

- [8] Vincent P, Larochelle H, Bengio Y and Manzagol P-A 2008 *Proc. of the 25th International Conference on Machine Learning* (New York: ACM) pp 1096–103
- [9] Hinton G E, Dayan P, Frey B J and Neal R M 1995 *Science* **268** 1158
- [10] Long P M and Servedio R A 2010 Random classification noise defeats all convex potential boosters *Mach. Learn.* **78** 287–304
- [11] Schuld M, Sinayskiy I and Petruccione F 2015 *Contemporary Physics* **56** 172
- [12] Wittek P 2014 *Quantum Machine Learning: What Quantum Computing Means to Data Mining* (New York: Academic)
- [13] Adcock J, Allen E, Day M, Frick S, Hinchliff J, Johnson M, Morley-Short S, Pallister S, Price A and Stanisis S 2015 arXiv preprint arXiv:1512.02900
- [14] Arunachalam S and de Wolf R 2017 arXiv preprint arXiv:1701.06806
- [15] Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N and Lloyd S 2017 *Nature* **549** 195
- [16] Harrow A W, Hassidim A and Lloyd S 2009 *Phys. Rev. Lett.* **103** 150502
- [17] Wiebe N, Braun D and Lloyd S 2012 *Phys. Rev. Lett.* **109** 050505
- [18] Childs A M, Kothari R and Somma R D 2015 *SIAM J. Comput.* **46** 1920–50
- [19] Lloyd S, Mohseni M and Rebentrost P 2014 *Nat. Phys.* **10** 631
- [20] Nielsen M A and Chuang I 2002 *Quantum Computation and Quantum Information* (College Park, MD: American Association of Physics Teachers)
- [21] Lidar D A and Brun T A 2013 *Quantum Error Correction* (Cambridge: Cambridge University Press)
- [22] Fowler A G, Mariantoni M, Martinis J M and Cleland A N 2012 *Phys. Rev. A* **86** 032324
- [23] Farhi E, Goldstone J, Gutmann S, Lapan J, Lundgren A and Preda D 2001 *Science* **292** 472
- [24] Johnson M W *et al* 2011 *Nature* **473** 194
- [25] Neill C, Roushan P, Kechedzhi K, Boixo S, Isakov S, Smelyanskiy V, Barends R, Burkett B, Chen Y, Chen Z, *et al* 2018 *Science* **360** 195–9
- [26] Kandala A, Mezzacapo A, Temme K, Takita M, Brink M, Chow J M and Gambetta J M 2017 *Nature* **549** 242
- [27] Kadowaki T and Nishimori H 1998 *Phys. Rev. E* **58** 5355
- [28] Santoro G E, Martonák R, Tosatti E and Car R 2002 *Science* **295** 2427
- [29] Brooke J, Rosenbaum T and Aeppli G 2001 *Nature* **413** 610
- [30] Farhi E, Goldstone J and Gutmann S 2014 arXiv preprint arXiv:1411.4028
- [31] Peruzzo A, McClean J, Shadbolt P, Yung M-H, Zhou X-Q, Love P J, Aspuru-Guzik A and O’Brien J L 2014 *Nat. Commun.* **5** 4213
- [32] Idotbach J *et al* 2017 arXiv preprint arXiv:1712.05771
- [33] Neven H, Denchev V S, Rose G and Macready W G 2008 arXiv preprint arXiv:0811.0416
- [34] Denchev V S, Ding N, Vishwanathan S and Neven H 2012 arXiv preprint arXiv:1205.1148
- [35] Pudenz K L and Lidar D A 2013 *Quantum information processing* **12** 2027
- [36] Mott A, Job J, Vlimant J-R, Lidar D and Spiropulu M 2017 *Nature* **550** 375
- [37] Amin M H 2015 *Phys. Rev. A* **92** 052323
- [38] Venuti L C, Albash T, Lidar D A and Zanardi P 2016 *Phys. Rev. A* **93** 032118
- [39] Albash T, Martin-Mayor V and Hen I 2017 *Phys. Rev. Lett.* **119** 110502
- [40] Vinci W and Lidar D A 2018 *Phys. Rev. A* **97** 022308
- [41] Denil M and De Freitas N in *NIPS 2011 2011 Deep Learning and Unsupervised Feature Learning Workshop*
- [42] Raymond J, Yarkoni S and Andriyash E 2016 *Frontiers in ICT* **3** 1–23
- [43] Korenkevych D, Xue Y, Bian Z, Chudak F, Macready W G, Rolfe J and Andriyash E 2016 arXiv preprint arXiv:1611.04528
- [44] Perdomo-Ortiz A, Benedetti M, Realpe-Gómez J and Biswas R 2018 *Quantum Sci. Technol.* **3** 030502
- [45] Adachi S H and Henderson M P 2015 arXiv preprint arXiv:1510.06356
- [46] Benedetti M, Realpe-Gómez J, Biswas R and Perdomo-Ortiz A 2016 *Phys. Rev. A* **94** 022308
- [47] Benedetti M, Realpe-Gómez J, Biswas R and Perdomo-Ortiz A 2017 *Phys. Rev. X* **7** 041052
- [48] Tieleman T 2008 *Proc. of the 25th International Conference on Machine Learning* (New York: ACM) pp 1064–71
- [49] Hukushima K and Iba Y 2003 *Conf. Proc.* (New York: AIP) vol 690 pp 200–6
- [50] Harris R *et al* 2018 *Science* **361** 162
- [51] King A D *et al* 2018 *Nature* **560** 456
- [52] Amin M H, Andriyash E, Rolfe J, Kulchitsky B and Melko R 2018 *Phys. Rev. X* **8** 021050
- [53] Dumoulin V, Goodfellow I J, Courville A and Bengio Y 2014 *Twenty-Eighth Conf. on Artificial Intelligence*
- [54] LeCun Y 1998 <http://yann.lecun.com/exdb/mnist/>
- [55] Benedetti M, Realpe-Gómez J, Perdomo-Ortiz A, Science Q and 2018 *Technology* **3** 034007
- [56] Kingma D P and Welling M 2013 arXiv preprint arXiv:1312.6114
- [57] Burda Y, Grosse R and Salakhutdinov R 2015 arXiv preprint arXiv:1509.00519
- [58] Rolfe J T 2016 arXiv preprint arXiv:1609.02200
- [59] Khoshaman A H and Amin M H 2018 arXiv preprint arXiv:1805.07349
- [60] Theis L, Oord A v d and Bethge M 2015 arXiv preprint arXiv:1511.01844
- [61] Goodfellow I J, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A and Bengio Y 2014 arXiv:1406.2661
- [62] Fergus R, Weiss Y and Torralba A 2009 *Advances in Neural Information Processing Systems* pp 522–30
- [63] Liu Y and Kirchhoff K 2013 *INTERSPEECH* 1840–3
- [64] Shi M and Zhang B 2011 *Bioinformatics* **27** 3017
- [65] Chen H and Zhang Z 2013 *PLoS one* **8** e62975
- [66] Kingma D P, Mohamed S, Rezende D J and Welling M 2014 *Advances in Neural Information Processing Systems* pp 3581–9
- [67] Hoffman M D, Blei D M, Wang C and Paisley J 2013 *The Journal of Machine Learning Research* **14** 1303
- [68] Williams R J 1992 *Mach. Learn.* **8** 229
- [69] Mnih A and Gregor K 2014 arXiv preprint arXiv:1402.0030
- [70] Jang E, Gu S and Poole B 2016 arXiv preprint arXiv:1611.01144
- [71] Maaløe L, Fraccaro M and Winther O 2017 arXiv preprint arXiv:1704.00637
- [72] Makhzani A and Frey B 2017 arXiv preprint arXiv:1706.00531
- [73] Paisley J, Blei D and Jordan M 2012 arXiv preprint arXiv:1206.6430
- [74] Gu S, Levine S, Sutskever I and Mnih A 2015 arXiv preprint arXiv:1511.05176
- [75] Bengio Y, Léonard N and Courville A 2013 arXiv preprint arXiv:1308.3432
- [76] Maddison C J, Mnih A and Teh Y W 2016 arXiv preprint arXiv:1611.00712

- [77] Marshall J, Rieffel E G and Hen I 2017 *Phys. Rev. Applied* **8** 064025
- [78] Marshall J, Venturelli D, Hen I and Rieffel E G 2019 *Phys. Rev. Applied* **11** 044083
- [79] *D-wave system documentation* <https://docs.dwavesys.com/docs/latest/index.html>.
- [80] Ackley D H, Hinton G E and Sejnowski T J 1985 *Cogn. Sci.* **9** 147
- [81] Boothby K, Bunyk P, Raymond J and Roy A 2019 *typeTech. Rep.* institutionTechnical report
- [82] Choi V 2008 *Quantum Information Processing* **7** 193
- [83] Choi V 2011 *Quantum Information Processing* **10** 343
- [84] Oord A v d, Kalchbrenner N and Kavukcuoglu K 2016 arXiv preprint arXiv:1601.06759
- [85] Dinh L, Krueger D and Bengio Y 2014 arXiv preprint arXiv:1410.8516