UNIVERSITY COLLEGE LONDON

DEPARTMENT OF COMPUTER SCIENCE

Quantum-classical generative models for machine learning

Author:

Marcello BENEDETTI

Supervisors:

Prof. Simone SEVERINI

Dr. Alejandro PERDOMO-ORTIZ

Submitted in partial fulfilment for the degree of **Doctor of Philosophy**

November 25, 2019

Abstract

The combination of quantum and classical computational resources towards more effective algorithms is one of the most promising research directions in computer science. In such a hybrid framework, existing quantum computers can be used to their fullest extent and for practical applications.

Generative modeling is one of the applications that could benefit the most, either by speeding up the underlying sampling methods or by unlocking more general models. In this work, we design a number of hybrid generative models and validate them on real hardware and datasets.

The *quantum-assisted Boltzmann machine* is trained to generate realistic artificial images on quantum annealers. Several challenges in state-of-the-art annealers shall be overcome before one can assess their actual performance. We attack some of the most pressing challenges such as the sparse qubit-to-qubit connectivity, the unknown effective-temperature, and the noise on the control parameters. In order to handle datasets of realistic size and complexity, we include latent variables and obtain a more general model called the *quantum-assisted Helmholtz machine*.

In the context of gate-based computers, the *quantum circuit Born machine* is trained to encode a target probability distribution in the wavefunction of a set of qubits. We implement this model on a trapped ion computer using low-depth circuits and native gates. We use the generative modeling performance on the canonical Bars-and-Stripes dataset to design a benchmark for hybrid systems.

It is reasonable to expect that quantum data, i.e., datasets of wavefunctions, will become available in the future. We derive a *quantum generative adversarial network* that works with quantum data. Here, two circuits are optimized in tandem: one tries to generate suitable quantum states, the other tries to distinguish between target and generated states.

Impact Statement

Unsupervised learning can extract salient spatio-temporal features from unlabeled data and is expected to become far more important than supervised learning in the long term. Indeed, the majority of data being collected every day does not come with task-specific informative labels (e.g., photos and videos uploaded to the Internet, medical imaging, tweets, audio recordings, financial time series, and sensor data in general), and labeling is an expensive process that requires human experts. The field of unsupervised learning abounds with computationally hard problems. This is an opportunity for quantum computation to demonstrate an advantage over classical computation.

In this thesis we select one of the main unsupervised learning problem, namely generative modeling, and approach it with the help of existing quantum computers. We utilize two different architectures, quantum annealing and gate-based trapped ion computers, and develop methods to overcome some of the severe challenges that affect existing hardware. This is important if we aim at benchmarking quantum computers on tasks of practical utility.

The models and the results presented in this thesis provide a starting point for junior researchers working in this novel field at the intersection of quantum computing and unsupervised learning. This work was carried out in a mixed academic/industrial setting. Researchers placed in either of these environments could find the reading interesting and useful.

The material presented in this thesis resulted in five publications on recognized peer-reviewed scientific journals. These provided the basis for further spin-off projects and publications where the author was involved.

Acknowledgements

Needless to say, my greatest appreciation goes to my family: Luana, Francesco, Mirella, Piero, Manila, Carolina, Tamara, and Justina.

For a variety of reasons, my gratitude goes to: Alberto Ottolenghi, Andrew G. Green, Andrew Hallam, Bert J. Kappen, Christopher Monroe, Danial Dervovic, Daiwei Zhu, Delfina Garcia-Pintos, Edward Grant, John Realpe-Gómez, Joshua Lockhart, Leonard Wossnig, Mateusz Ostaszewski, Mattia Fiorentini, Max Wilson, Miriam Cha, Norbert M. Linke, Oscar Perdomo, Raban Iten, Radu Serban, Shuxiang Cao, Vicente Leyton-Ortega, and Yunseong Nam.

A special thanks goes to Alejandro Perdomo-Ortiz and Simone Severini for being awesome supervisors, and to Ilyas Khan for being supportive at all times.

I thank the National Aeronautics and Space Administration (NASA) and the Quantum Artificial Intelligence Laboratory (QuAIL) for providing access to their quantum annealers and for hosting me in 2016 and 2017 as part of their internship program. I thank the University of Maryland (UMD) and the Trapped Ion Quantum Information group (TIQI) for providing access to their trapped ion quantum computer.

Finally, I acknowledge Cambridge Quantum Computing Limited (CQC) and the Engineering and Physical Sciences Research Council (EPSRC) for providing the financial support for this three years' work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With quantum computing technologies nearing the era of commercialisation and of quantum supremacy [1], it is important to think of potential applications that might benefit from these devices. Machine learning (ML) stands out as a powerful statistical framework to attack problems where exact algorithms are hard to develop. Examples of such problems include image recognition [2], speech recognition [3], autonomous systems [4], medical applications [5], biology [6], artificial intelligence [7], and many others. The development of quantum algorithms that can assist ML is an ongoing effort that has attracted a lot of interest in the scientific community.

Research in this field has been focusing on classification [8], regression [9, 10, 11], Gaussian models [12], vector quantization [13], principal component analysis [14] and other methods that are routinely used by ML practitioners. We do not think these approaches would be of practical use in near-term quantum computers. The reasons that make these techniques so popular is their scalability and algorithmic efficiency in tackling huge datasets. Therefore, even if polynomial and, in some cases, exponential speedups are expected for these algorithms, reaching interesting industrial scale applications would require fault-tolerant computers with millions of qubits. This makes the aforementioned algorithms less likely to become killer applications with devices in the range of 100-1000 noisy qubits.

As we elaborate in this Thesis, only a game changer might be able to make a dent in speeding up ML tasks. Our strategy is to select a problem that is considered hard by the ML community. We chose to focus on *unsupervised generative modeling*, which we introduce now.

The majority of data being collected daily is unlabeled. Examples of unlabeled data are photos and videos uploaded to the Internet, medical imaging, tweets, audio recordings, financial time series, and sensor data in general. Labeling is the process of data augmentation with task-specific informative tags, an expensive process that requires human experts. It is therefore important to design models and algorithms capable of extracting information and structures from unlabeled data; that is the focus of *unsupervised* learning.

But why is this important at all? The discovery of patterns is one of the central aspects of science. Scientists do not always know *a priori* which patterns they should look for and they need unsupervised tools to extract salient spatio-temporal features from unlabeled data. Unsupervised techniques are designed to capture the useful features of high-dimensional datasets, enforcing desirable properties such as simplicity, sparsity [15], and therefore interpretability. As highlighted by ML experts, it is expected that unsupervised learning will become far more important than purely supervised learning in the long term [16].

Back in February 1988, Richard Feynman wrote on his blackboard: 'What I cannot create, I do not understand' [17]. Since then this powerful dictum has been reused and reinterpreted in the context of many fields throughout science. In the context of unsupervised learning, it is often used to describe *generative models*, algorithms that can generate realistic artificial examples of their environment and therefore are likely to 'understand' such an environment. Generative models are trained to approximate the joint probability distribution of a set of variables, given a dataset of observations. The joint probability provides both a way to generate new artificial data resembling the dataset, and a way to infer marginal and conditional distributions of the variables. These possibilities make generative models extremely useful in practice.

Generative models are often thought of as graphs where vertices represent random variables, and edges represent the conditional dependence structure between variables. Exact learning and inference in these graphs is intractable in all but the most trivial topologies [18]. The bottleneck is in the computation of the partition function, or normalisation constant, which is needed to compute exact expectation values. Classically this is often approximated by variational methods which are fast, but may lack precision [19]. To go beyond this, one often resorts to Markov chain Monte Carlo (MCMC) methods that, however, may suffer from the slow-mixing problem [20]. It is indeed difficult for the Markov chain to jump from one mode of the distribution to another when these are separated by low-density regions of relevant size. MCMC is therefore hard to scale to large datasets.

We believe there is an opportunity here for quantum computers to assist sampling and improve classical models. This topic is explored in Chapter 2 of this Thesis, where we develop *quantum-assisted models* as well as techniques for an experimental validation of our hypothesis in near-term computers. We begin with a presentation of *quantum annealing* as an algorithm for sampling. This is in contrast to the standard view where annealing is considered an algorithm for combinatorial optimisation [21, 22]. Inspired by this different point of view, we design two generative models: the quantum-assisted Boltzmann machine (QABM) and the quantum-assisted Helmholtz machine (QAHM). We provide clear experimental evidence that quantum annealers can train these models robustly and can handle real-world datasets. To this extent, we employ D-Wave processors hosted by NASA Ames Research Center – the D-Wave 2X and the D-Wave 2000Q.

We emphasise that our objective is not to address the question of quantum speedup in sampling applications analytically. Instead, we design tools to attack this question empirically using near-term quantum annealers. Some of the challenges we overcome are the limited qubit-to-qubit connectivity, the unknown effective temperature, the intrinsic noise in the device, the uncertainty in the control parameters, the handling of continuous variables, and the need to tackle large-scale datasets. The results of Chapter 2 are based on the following peer-reviewed articles:

1. Benedetti, M., Realpe-Gómez, J., Biswas, R. and Perdomo-Ortiz, A., 2017. Quantum-assisted learning of hardware-embedded probabilistic graphical models. *Physical Review X*, 7(4), p.041052.

2. Benedetti, M., Realpe-Gómez, J. and Perdomo-Ortiz, A., 2018. Quantum-assisted Helmholtz machines: a quantum–classical deep learning framework for industrial datasets in near-term devices. *Quantum Science and Technology*, 3(3), p.034007.

3. Perdomo-Ortiz, A., Benedetti, M., Realpe-Gómez, J. and Biswas, R., 2018. Opportunities and challenges for quantum-assisted machine learning in near-term quantum computers. *Quantum Science and Technology*, 3(3), p.030502.

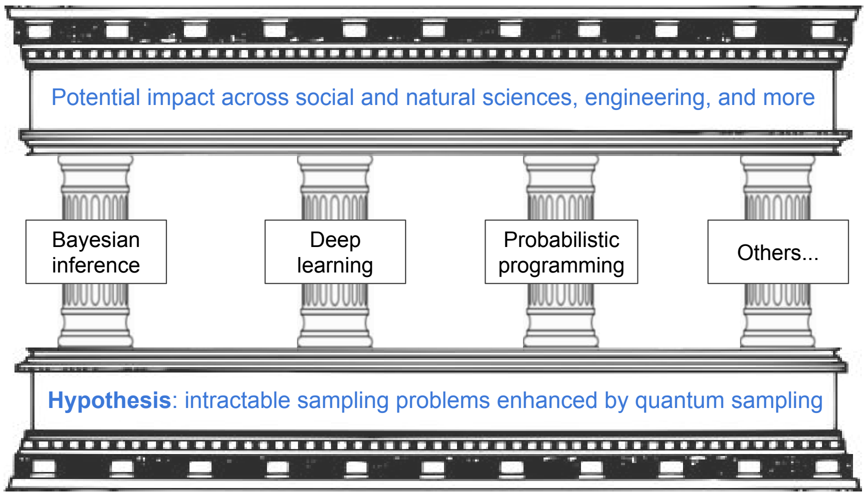

Potential impact across social and natural sciences, engineering, and more
Bayesian inference
Deep learning
Probabilistic programming
Others...
Hypothesis: intractable sampling problems enhanced by quantum sampling

Figure 1.1: Sampling applications in machine learning as an opportunity for quantum computers. Quantum computers have the potential to sample efficiently from complex probability distributions. This task is a computationally intractable step in many machine learning domains. A quantum advantage in these domains could in turn have a significant impact on science and engineering.

We shall remark that, if quantum computers were shown to outperform classical computers in sampling, the advantages would not be restricted to generative models. As illustrated in Figure 1.1, sampling is at the core of several leading-edge domains such as Bayesian inference [23], deep learning [24], and probabilistic programming [25]. With a quantum advantage in these domains, we would in turn expect a strong impact across science and engineering.

Let us now discuss a different type of quantum advantage. Researchers at Google demonstrated that quantum computers with as few as 50 qubits can attain quantum supremacy, although in a task with no obvious applications [26]. A highly relevant question then is whether *quantum models* exhibiting quantum supremacy can provide a benefit on real-world applications when using near-term hardware. To clarify, rather than assisting classical models as suggested before, one can design models whose inner working is naturally described by quantum mechanics.

Recent work showed that quantum mechanics can provide more parsimonious models of stochastic processes than classical models, as quantified by an entropic measure of complexity [27, 28, 29]. Other work showed that a quantum generalisation of maximum likelihood learning can yield results that are more accurate on some problems [30]. Quantum models could then substantially reduce the amount of other type of computational resources, e.g., memory requirements, and could be substantially simpler than classical models as quantified by standard model comparison techniques, e.g., the Akaike information criterion [31]. In a sense, this is a form of quantum advantage.

Chapter 3 of the Thesis is dedicated to the design of *quantum generative models* and the development of learning algorithms and heuristics for near-term computers. We think that these are important steps towards the validation of the hypothesis that quantum models can significantly outperform classical models.

But what kind of data can be handled by a quantum model? Datasets generated in experiments with quantum systems are an obvious fit. More generally though, it is possible to conceive inputs and outputs that are inherently quantum mechanical, i.e., already in superposition; these are often referred to as *quantum data* [32]. Quantum data could originate remotely, for example, from quantum computers transmitting over a quantum Internet [33]. Alternatively, quantum data can be prepared locally if a recipe is available, e.g., if a suitable state-preparation circuit is known. Assuming this preparation be efficient, one can extend the fields of supervised and unsupervised learning to quantum datasets and perform interesting quantum information tasks.

To this extent, let us go back to our main focus, generative modeling, and generalise it. Recall that the aim is to approximate the joint probability distribution of a set of variables from a dataset of observations. Conceptually, the generalisation is straightforward: quantum generative models are algorithms trained to approximate the density operator of a set of qubits, given a dataset of quantum states. It turns out that this process of approximately reconstructing a density operator is well-known to physicists under the name of quantum state tomography. Indeed, there already exist proposals of generative models for tomography such as the quantum principal component analysis [14], the quantum Boltzmann machine [34, 35], and the probably approximately correct learning [36, 37].

In summary, ML provides a set of new tools to physicists and, conversely, quantum mechanics provides a set of new tools to ML practitioners. This intersection is one of the most promising avenues of science. Chapter 3 takes concrete steps towards the validation of this hypothesis while working under the highly constrained environment of existing quantum hardware. We begin with a review of *parameterized quantum circuits* as a way to implement algorithms in near-term gate-based

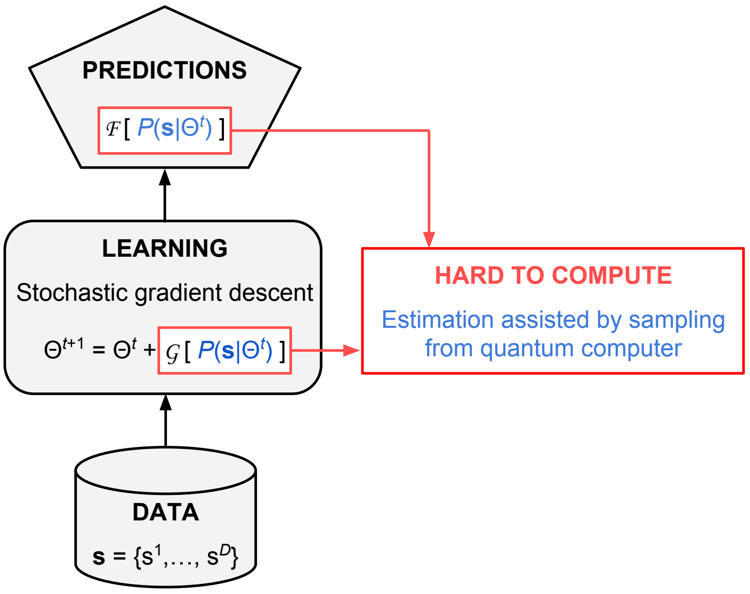

PREDICTIONS
$\mathcal{F}[\,P(\mathbf{s}|\Theta^t)\,]$
LEARNING
Stochastic gradient descent
$\Theta^{t+1} = \Theta^t + \mathcal{G}[\,P(\mathbf{s}|\Theta^t)\,]$
HARD TO COMPUTE
Estimation assisted by sampling from quantum computer
DATA
$\mathbf{s} = \{s^1, \dots, s^D\}$

Figure 1.2: Example of a hybrid quantum-classical system for machine learning. A dataset $\boldsymbol{s}$ drives the learning of model parameters Θ from time t to $t+1$. The learning algorithm requires the computation of a function $\mathcal{G}$ of the probability distribution implemented by the model $P(\mathbf{s}|\Theta^t)$. This computationally hard step can be estimated from samples and assisted by a quantum computer. Making predictions $\mathcal{F}$ out of the trained model is also hard and requires assistance of a quantum computer.

computers. We discuss in detail the hybrid setting where a classical computer is used for the tractable subroutines of the algorithm and a quantum computer is used only for the intractable steps. Figure 1.2 illustrates this concept with an example.

We then design two generative models: the quantum circuit Born machine (QCBM) and the quantum generative adversarial network (QGAN). We provide evidence that low-depth circuits driven by data alone learn to approximate classical probability distributions and prepare interesting quantum states. To this extent, we employ a trapped ion computer hosted by University of Maryland, as well as extensive *in silico* simulations. Based on the generative modeling performance, we design a metric for benchmarking both quantum and classical parts of the system.

Theoretical work has shown that, under well-believed complexity arguments, some classes of low-depth circuits cannot be simulated efficiently by classical means (e.g., instantaneous quantum polynomial-time circuits) [38, 39]. This implies that some classes of QCBMs and QGANs have strictly more expressive power than classical models [40, 41]. In part, this result justifies the work done in Chapter 3 of this Thesis; however, it does not necessarily imply a practical advantage for ML applications. Once again, we opt for an empirical approach towards answering the question of advantage in quantum models. The results of Chapter 3 are based on the following peer-reviewed articles:

4. Benedetti, M., Garcia-Pintos, D., Perdomo, O., Leyton-Ortega, V., Nam, Y. and Perdomo-Ortiz, A., 2019. A generative modeling approach for benchmarking and training shallow quantum circuits. *npj Quantum Information*, 5(1), p.45.

5. Benedetti, M., Grant, E., Wossnig, L. and Severini, S., 2019. Adversarial quantum circuit learning for pure state approximation. *New Journal of Physics*, 21(4), p.043023.

6. Benedetti, M., Lloyd, E., Sack, S., and Fiorentini, M., 2019. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4), p.043001.

Chapters 2 and 3 are meant to be self-contained and can be read in reverse order if desired. We do assume familiarity with basic machine learning definitions and methods (see Mehta *et al.* [42] for a physics-oriented introduction), and basic working knowledge on quantum annealing (see Hauke *et al.* [43]) and quantum computing (see Nielsen and Chuang [44], Chapter 2). We conclude the Thesis in Chapter 4 with a brief outlook of the field.

The author of this Thesis was also involved in several spin-off projects and publications which are *not* included in this Thesis. For completeness, we list these publications here:

- Serban, R., Wilson, M., Benedetti, M., Realpe-Gómez, J., Perdomo-Ortiz, A., Petukhov, A. and Jayakumar, P., 2018. Quantum annealing for mobility studies: go/no-go maps via quantum-assisted machine learning. *Proceedings of the 2018 Ground Vehicle Systems Engineering and Technology Symposium (GVSETS)*

- Grant, E., Benedetti, M., Cao, S., Hallam, A., Lockhart, J., Stojevic, V., Green, A.G. and Severini, S., 2018. Hierarchical quantum classifiers. *npj Quantum Information*, 4(1), p.65.

- Zhu, D., Linke, N.M., Benedetti, M., Landsman, K.A., Nguyen, N.H., Alderete, C.H., Perdomo-Ortiz, A., Korda, N., Garfoot, A., Brecque, C. and Egan, L., 2019. Training of quantum circuits on a hybrid quantum computer. *Science advances*, 5(10), p.eaaw9918.

- Grant, E., Wossnig, L., Ostaszewski, M. and Benedetti, M., 2019. An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *arXiv preprint arXiv:1903.05076.*

- Ostaszewski, M., Grant, E. and Benedetti, M., 2019. Quantum circuit structure learning. *arXiv preprint arXiv:1905.09692.*

Chapter 2

Annealing-based generative models

2.1 The framework

Generative models rely on a sampling engine that is used for both inference and learning. Because of the intractability of traditional sampling techniques like the Markov chain Monte Carlo (MCMC) method, finding good generative models is among the hardest problems in machine learning [16, 24, 45, 46, 47].

Sampling goes beyond the original focus of the quantum annealing computational paradigm [22, 21, 48], which was to solve discrete optimisation problems [49, 50, 51, 52, 53, 54, 55, 56, 57, 58]. However, state-of-the-art quantum annealers have a strong interaction with the environment leading to relatively fast thermalisation and decoherence. Theory suggests that in these cases the relevant quantum dynamics essentially freezes during annealing somewhere between the critical point associated with the minimum gap and the end of the annealing schedule [59, 60]. Indeed, empirical results showed that under certain conditions quantum annealers sample from a Gibbs distribution [61, 62, 63].

More formally, the dynamics of a quantum annealer is characterised by the time-dependent Hamiltonian

$$\mathcal{H}(\tau) = -A(\tau)\sum_{i\in\mathcal{V}}\hat{X}_i + B(\tau)\mathcal{H}_P, \tag{2.1}$$

where $\tau = t/t_a$ is the ratio between time t and annealing time t_a, and where $A(\tau)$ and $B(\tau)$ are monotonic functions satisfying $A(0) \gg B(0) \approx 0$ and $B(1) \gg A(1) \approx 0$. The first term in Eq. (2.1) corresponds to the transverse field in the x direction, characterised by the Pauli operators $\hat{X}_i$ for each qubit i in the set of qubits $\mathcal{V}$. The second term in Eq. (2.1) corresponds to the problem-encoding Hamiltonian

$$\mathcal{H}_P = -\sum_{(i,j)\in\mathcal{E}} J_{ij}\hat{Z}_i\hat{Z}_j - \sum_{i\in\mathcal{V}} h_i\hat{Z}_i, \tag{2.2}$$

where Pauli operator $\hat{Z}_i$ refers to the z direction for the i-th qubit, and where J_{ij} and h_i are the control parameters. The expression above is defined on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of qubits and $\mathcal{E}$ is the set of qubit-to-qubit interactions available in hardware.

As discussed in Ref. [61], the dynamics of a quantum annealer is expected to remain close to equilibrium until they slow down and deviate from equilibrium to finally freeze out. If the time between the dynamical slow-down and the freeze-out is small enough, the final state of the quantum annealer is expected to be close to the Gibbs state

$$\rho = \frac{e^{-\beta_{\mathrm{QA}}\mathcal{H}(\tau^*)}}{\mathcal{Z}}, \tag{2.3}$$

corresponding to the Hamiltonian in Eq. (2.1) evaluated at $\tau = \tau^*$, also known as the *freeze-out time*. Here, β_{QA} is the inverse physical temperature of the quantum annealer, and $\mathcal{Z}$ is the normalisation constant. The density in Eq. (2.3) is fully specified by the effective parameters βJ_{ij}, βh_i, and $\beta\Gamma$, where $\beta = \beta_{\mathrm{QA}} B(\tau^*)$ is the effective inverse temperature and $\Gamma = A(\tau^*)/B(\tau^*)$ is the effective transverse field.

In the case where $A(\tau^*) \ll B(\tau^*)$, the final state of the quantum annealer is close to a classical Boltzmann distribution over a vector of binary variables $\boldsymbol{z} \in \{-1, +1\}^N$,

$$P(\boldsymbol{z}) = \frac{e^{-\beta E(\boldsymbol{z})}}{\mathcal{Z}}, \tag{2.4}$$

where

$$E(\boldsymbol{z}) = -\sum_{(i,j)\in\mathcal{E}} J_{ij} z_i z_j - \sum_{i\in\mathcal{V}} h_i z_i, \tag{2.5}$$

is the energy function given by the eigenvalues of $\mathcal{H}_P$. In other words, z_i are eigenvalues of $\hat{Z}_i$. The intuition is that the dominant coupling of a qubit to the environment is via the $\hat{Z}$ operator (see the supplementary material of Refs. [64, 65]), and since

$A(\tau^*) \ll B(\tau^*)$ by assumption, the interaction with the bath lacks a strong $\hat{X}$ component capable of causing relaxation between the states of the computational basis. In other words, the population dynamics is frozen. This suggests that, in principle, the user could adjust the control parameters J_{ij} and h_i so that the device samples from a desired Boltzmann distribution[1].

Figure 2.1 shows an example where the binary distribution implemented by a quantum annealer is used for the generative modeling of black-and-white handwritten digits. Panel (a) shows the learning phase. Here, samples generated by the quantum annealer are compared with samples from the dataset of digits. The control parameters are then modified according to a learning rule in an iterative process. Panel (b) shows the inference phase. The learned model is used to reconstruct missing information in a corrupted image. To do this, we start by programming the quantum annealer with the learned control parameters. Then, for those qubits that represent known pixels, we set the control parameters to large values $\pm h_{\max}$ of the correct sign. These qubits are 'clamped' in the sense that with high probability they are sampled to the known values of the corresponding pixels. Finally, we sample from the annealer to infer the most likely values of the unknown pixels. Each sample provides a reconstructed image. Yet why is quantum annealing expected to help in the computational task of sampling from complex probability distributions?

Tunneling, the quantum effect at the core of annealing, is a powerful computational resource for keeping the system close to its ground state [65]. It is this quantum resource, available before the freeze-out time, that might speed up the thermalisation process and make sampling more efficient than classical MCMC. We shall not, however, expect such a quantum advantage for all energy landscapes. There will be instances that cannot be sampled efficiently by both classical and quantum resources. Hence, the answer to this question depends on both the quantum resources available and the complexity of the energy landscape. Generative modeling and ML in general provide a variety of real-world instances to validate this hypothesis.

In practice, however, there exist device-dependent limitations that complicate

[1]When $A(\tau^*)$ cannot be neglected, the quantum annealer samples from a quantum distribution. This could enable the implementation of some classes of quantum models, for example, the quantum Boltzmann machine [34, 66, 35, 30].

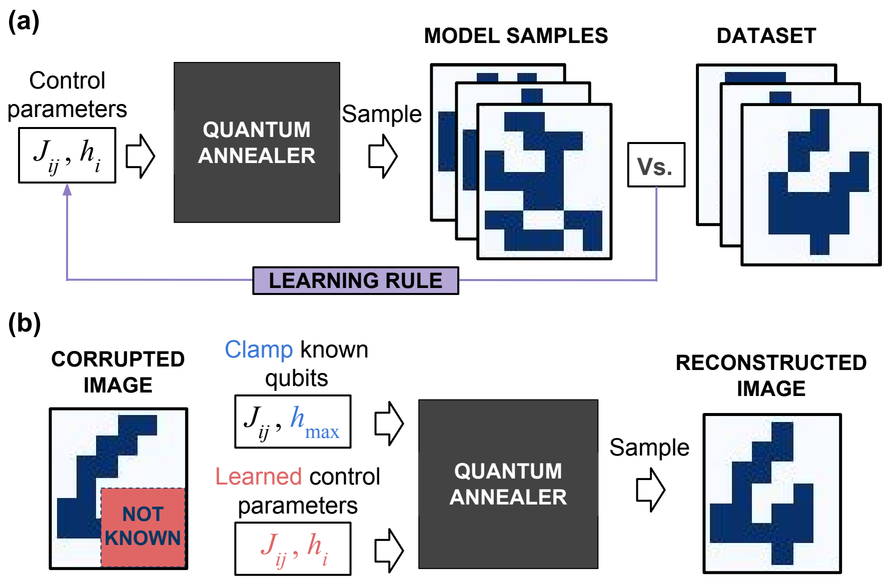


Figure 2.1: Example of annealing-based generative models for binary images. (a) Learning consists of adjusting control parameters so that the generated samples become similar to those in the dataset. (b) Inference consists of using the learned model to sample parts of a corrupted image and obtain a reconstruction.

the process just described. The most pressing ones are as follows [62, 63, 67, 68, 69]:

(i) the qubit-to-qubit interaction graph $\mathcal{E}$ in Eq. (2.2) is sparse;

(ii) the freeze-out time τ^* and the effective inverse temperature β are unknown and depend on the values of the control parameters J_{ij} and h_i;

(iii) the control parameters J_{ij} and h_i are noisy; and

(iv) the dynamical range of J_{ij} and h_i is finite.

Suitable strategies to tackle all of these limitations are needed before we can assess whether quantum annealers can sample more efficiently than traditional techniques on classical computers, or whether they can implement more effective models. As an example, relatively simple techniques for the estimation of parameter-dependent effective temperature were shown to enable the learning of some Boltzmann models via quantum annealing [62, 63]. However, the need to estimate temperature at each learning iteration implied a significant computational overhead.

In Section 2.2, we put forward an approach that mitigates all these limitations. It consists of using available graph embedding techniques to *effectively* implement all pairwise interactions between variables in quantum hardware, hence improving on limitation (i). This introduces a number of additional control parameters to be adjusted. To this extent we first transform the dataset into another dataset with higher and redundant resolution; then we learn all parameters by minimising a divergence between this new dataset and the quantum annealer distribution. This does not require estimation of the effective temperature, hence sidestepping limitation (ii), and is robust to the noise in the control parameters, hence improving on limitation (iii). More specifically, the learning algorithm has the potential to correct for errors due to non-equilibrium deviations [61], systematic biases in the parameters [70], and sampling biases which are common to state-of-the-art quantum annealers [71]. Finally, available regularisation techniques are used to control the magnitude of all parameters, hence improving on limitation (iv).

We will refer to this approach as the *grey-box approach*. This is because although both the model and the learning algorithm rely on the assumption that the quantum annealer is sampling from a Gibbs distribution, we do not expect this assumption to be strictly valid for our approach to work well. This shall become clear in Section 2.2 where we use the grey-box approach to experimentally demonstrate a quantum-assisted Boltzmann machine (QABM) for small binarized datasets.

The natural extension of this approach is to develop techniques that can handle large realistic datasets. This would open up the possibility to use quantum-assisted models in real-world domains and to benchmark them against extensively studied classical models. In particular, this means tackling the following additional problems:

(v) the model could be augmented with hidden variables;

(vi) the variables in the dataset could be continuous or discrete, but the annealer produces binary strings;

(vii) the number of variables in the dataset could be much larger than the number of qubits in the annealer; and

(viii) the dataset could be made up of a large number of observations (i.e., data vectors).

To clarify problem (vi), in Appendix 5.1 we argue why a direct implementation of stochastic continuous variables in state-of-the-art quantum annealers would be challenging. Problems (vii) and (viii) are clear from their definition. We shall now clarify problem (v).

Deep generative models, i.e., models with many layers of hidden variables, have the ability to learn multi-modal distributions over high-dimensional datasets [72]. Each additional layer provides an increasingly abstract representation of the data and improves the generalisation capability of the model [73].

More formally, a deep generative model implements a probability distribution $P(\boldsymbol{v}) = \sum_{\boldsymbol{u}} P(\boldsymbol{v}|\boldsymbol{u})P(\boldsymbol{u})$, where $\boldsymbol{v} = \{v_1, \ldots, v_N\}$ are visible variables encoding the data (e.g., pixels) and $\boldsymbol{u} = \{u_1, \ldots, u_M\}$ are unobserved, or hidden, variables that serve to capture non-trivial correlations by encoding high-level features. To perform both inference and learning on these models, we need to sample from the posterior distribution $P(\boldsymbol{u}|\boldsymbol{v})$, and this is intractable in general. A standard approach consists of introducing a distribution $Q(\boldsymbol{u}|\boldsymbol{v})$ to approximate the true posterior [19]. Such a distribution should be chosen from a family that is both expressive and tractable. The learning algorithm is then in charge of adjusting $Q(\boldsymbol{u}|\boldsymbol{v})$ to approximate $P(\boldsymbol{u}|\boldsymbol{v})$, and adjusting $P(\boldsymbol{v})$ to model the data.

Let us now introduce some deep architectures that extend the QABM and that can work in synergy with quantum devices. In Figure 2.2, generative models are represented as graphs of stochastic vertices. Edges can be directed and undirected, respectively indicating conditional and joint distributions. We use the blue colour for vertices that can be implemented by qubits on a quantum device, and we use an edge marked at both ends to indicate a quantum interaction. Panel (a) shows the quantum-assisted version of the Helmholtz machine [74, 75, 76] (QAHM), which consists of two networks: a *recognition network* to do approximate inference on hidden variables, and a *generator network* to generate artificial data. The recognition network implements the aforementioned distribution $Q(\boldsymbol{u}|\boldsymbol{v})$ and performs bottom-up sampling starting from visible variables $\boldsymbol{v}$. This network may be entirely classical or quantum-assisted. The generator network, instead, implements the distribution $P(\boldsymbol{u}, \boldsymbol{v})$ and is used to perform top-down sampling starting from the deepest hidden layer (e.g., $\boldsymbol{u}^2$ in Fig. 2.2). Here the deepest hidden layer is modeled by qubits and

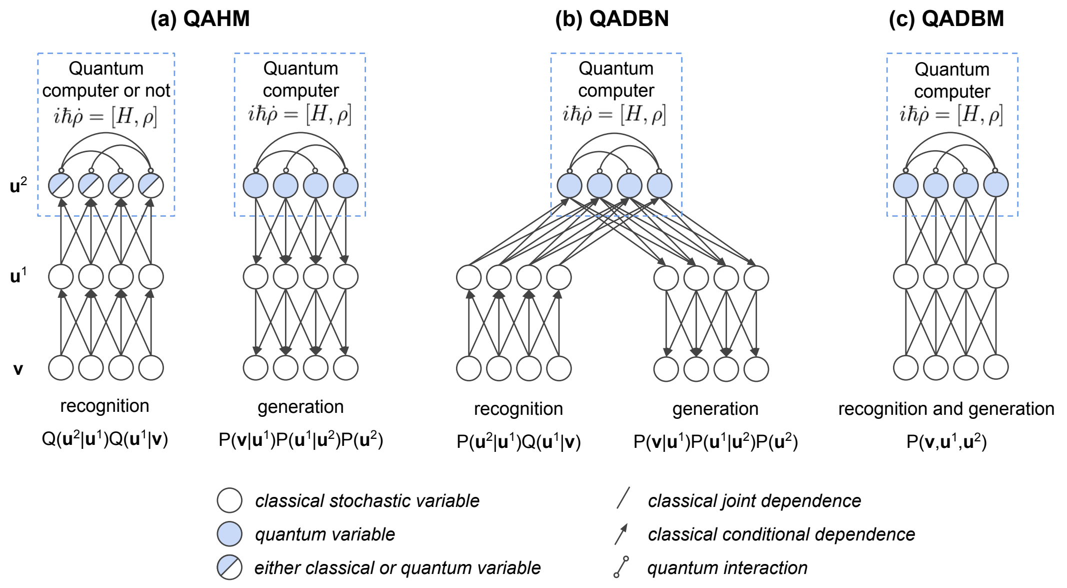

(a) QAHM
(b) QADBN
(c) QADBM
Quantum computer or not
Quantum computer
$i\hbar\dot{\rho} = [H, \rho]$
$\mathbf{u}^2$
$\mathbf{u}^1$
$\mathbf{v}$
recognition
$Q(\mathbf{u}^2|\mathbf{u}^1)Q(\mathbf{u}^1|\mathbf{v})$
generation
$P(\mathbf{v}|\mathbf{u}^1)P(\mathbf{u}^1|\mathbf{u}^2)P(\mathbf{u}^2)$
recognition
$P(\mathbf{u}^2|\mathbf{u}^1)Q(\mathbf{u}^1|\mathbf{v})$
generation
$P(\mathbf{v}|\mathbf{u}^1)P(\mathbf{u}^1|\mathbf{u}^2)P(\mathbf{u}^2)$
recognition and generation
$P(\mathbf{v},\mathbf{u}^1,\mathbf{u}^2)$
classical stochastic variable
quantum variable
either classical or quantum variable
classical joint dependence
classical conditional dependence
quantum interaction

Figure 2.2: Deep architectures for quantum-assisted generative modeling: (a) quantum-assisted Helmholtz machine (QAHM); (b) quantum-assisted deep belief network (QADBN); and (c) quantum-assisted deep Boltzmann machine (QADBM).

quantum interactions.

If the recognition and generator networks share the same quantum layer, we obtain the quantum-assisted version of a deep belief network [73, 24] (QADBN), as shown in Panel (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 the quantum-assisted version of a deep Boltzmann machine [77, 78] (QADBM), which we show in Panel (c).

In other words, our proposal consists in using the quantum annealer to sample the most abstract representation of the data, that is, the deepest layer of a deep generative model. 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; this effectively tackles problem (v) – the need for more expressive power through hidden variables. The lower layers of the network are classical components (e.g., pre- and post-processing stochastic functions) that effectively transform samples from the quantum annealer to data vectors, and vice-versa. Hence, visible variables could be continuous variables, discrete variables, or other objects, effectively solving problem (vi). 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 – hence improving on problem (vii).

All three quantum-assisted deep architectures described above can be readily implemented and tested on available quantum annealers. However, problem (viii) is a practical caveat related to the fact that learning the large number of parameters in a deep architecture may require large datasets [79]. For each data vector, we then need to infer the corresponding value of hidden variables using the recognition network, 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 large datasets[2]. The more flexible framework of QAHM opens up the possibility of using a completely classical recognition network, sidestepping problem (viii) altogether.

2.2 The quantum-assisted Boltzmann machine

2.2.1 Model definition

Let us consider a dataset $\mathcal{D} = \{\boldsymbol{s}^1, \ldots, \boldsymbol{s}^D\}$, where each data vector can be represented as an array of binary variables, i.e., $\boldsymbol{s}^d = (s_1^d, \ldots, s_N^d)$, with $s_i^d \in \{-1, +1\}$, for $i = 1, \ldots, N$. The Boltzmann machine [77] assumes that the correlations in the dataset can be modeled by a Boltzmann probability distribution. That is

$$P(\boldsymbol{s}) = \frac{e^{-\beta E(\boldsymbol{s})}}{\mathcal{Z}}, \tag{2.6}$$

where

$$E(\boldsymbol{s}) = -\sum_{(i,j)\in\mathcal{E}} J_{ij} s_i s_j - \sum_{i\in\mathcal{V}} h_i s_i, \tag{2.7}$$

is the energy function. These Equations are identical to Eqs. (2.4) and (2.5) in the previous Section, except for $\boldsymbol{s}$ which here represents the *logical* variables in the generative tasks, and not qubits.

We would like to use a Boltzmann model where the interactions are described

[2]For example, the canonical MNIST dataset is composed of 60 000 training vectors, hence we would need to program the quantum device at least 60 000 times.

by a graph $\mathcal{E}$ with complete connectivity (i.e., all-to-all). As this is the most general case, our derivations include any other graph topology with pairwise connectivity.

In the previous Section, we justified why quantum annealers are expected to approximately sample from a Gibbs distribution. However, the sparse qubit-to-qubit interaction graph of state-of-the-art quantum annealers strongly limits their capacity to model complex data, as shown in previous simulations [69] and experiments [62]. The typical strategy to embed dense graphs in quantum hardware is to map each logical variable to a several physical qubits forming a *subgraph*, therefore increasing the effective connectivity. In other words, one replaces each vertex in graph $\mathcal{E}_{\text{dense}}$ with a connected component in graph $\mathcal{E}_{\text{sparse}}$ so that graph $\mathcal{E}_{\text{dense}}$ can be recovered by contracting edges in $\mathcal{E}_{\text{sparse}}$. The problem of finding this map is called the *minor-embedding* problem [80, 81].

Finding the optimal minor-embedding with respect to the number of qubits is computationally intractable [82, 83]. Our algorithm, however, does not require an optimal embedding. Instead, it requires *any* minor-embedding of the complete graph into the interaction graph of the quantum annealer. In many practical scenarios, this can be found efficiently by off-the-shelf heuristics [84][3].

Let us then assume we have a minor-embedding of the complete graph of the N logical variables into the sparse graph of M physical qubits. We now use it to define a map f from the data space to the qubit space. The map allows us to convert the original dataset $\mathcal{D} = \{\boldsymbol{s}^1, \dots, \boldsymbol{s}^D\}$ to an extended dataset $\widetilde{\mathcal{D}} = \{\boldsymbol{z}^1, \dots, \boldsymbol{z}^D\}$, where $\boldsymbol{z}^d = f(\boldsymbol{s}^d)$. Note that the number of vectors in the dataset is preserved, but the dimension of each vector increases from N to M. Here, we choose a map that copies the state of each logical variable, i.e.,

$$z_i^{(k)} = s_i, \quad \text{for} \quad k = 1, \dots, Q_i, \tag{2.8}$$

where Q_i is the number of qubits in subgraph i.

The generative task now turns into the problem of learning a model for the extended dataset $\widetilde{\mathcal{D}}$. Let us define a new problem Hamiltonian over $M = \sum_{i=1}^{N} Q_i$

[3]When possible one could also use known minor-embedding prescriptions. For example, the schema in Ref. [81] embeds a complete graph of N variables into D-Wave's Chimera graph using $M \sim \mathcal{O}(N^2)$ qubits.

qubits,

$$\widetilde{\mathcal{H}}_P = -\frac{1}{2}\sum_{i,j=1}^{N}\sum_{k,l=1}^{Q_i,Q_j} J_{ij}^{(kl)}\hat{Z}_i^{(k)}\hat{Z}_j^{(l)} - \sum_{i=1}^{N}\sum_{k=1}^{Q_i} h_i^{(k)}\hat{Z}_i^{(k)}. \tag{2.9}$$

where $\hat{Z}_i^{(k)}$ is the Pauli matrix in the z direction for qubit k of subgraph i, $h_i^{(k)}$ is the field parameter for qubit k of subgraph i, and $J_{ij}^{(kl)}$ is the coupling parameter between qubit k of subgraph i and qubit l of subgraph j. When $i = j$, the coupling specifies the interaction *within* the subgraph; when $i \neq j$, it specifies the interaction *among* subgraphs. Note that $J_{ij}^{(kl)} = 0$ if there is no available interaction between the corresponding qubits in the quantum hardware. Variables $z_i^{(k)}$ encoding the extended dataset are then interpreted as the eigenvalues of the Pauli operator $\hat{Z}_i^{(k)}$.

Finally, we introduce a map g from qubit space to data space. Essentially, it transforms samples generated by the quantum annealer into samples that resemble the original dataset. Here, we choose a map that takes the majority vote of physical variables, i.e.,

$$s_i = \operatorname{sign}\left(\sum_{k=1}^{Q_i} z_i^{(k)}\right). \tag{2.10}$$

The rationale behind this choice is that an ideal learned model is expected to sample the same value for all physical variables $z_i^{(k)}$ in a subgraph i, i.e., $z_i^{(k)} = z_i^{(l)}$ for $k, l = 1, \dots, Q_i$. In this case, we could pick whichever $z_i^{(k)}$ as representative of the logical variable s_i, and this choice would be equivalent to the choice in Eq. (2.10). However, we expect Eq. (2.10) to be more robust to the different sources of noise in quantum annealers as it exploits redundancy in the spirit of error-correction codes [85, 86].

At this point, we notice that a number of *auxiliary* control parameters J_{ij} and h_i have been introduced by the minor-embedding. We shall now discuss how to deal with those. In optimisation, which is the original focus of quantum annealing, one seeks the vector $\boldsymbol{s}^*$ associated with the lowest energy in Eq. (2.7). In this case, the value of all auxiliary parameters in Eq. (2.9) should be fine-tuned so that the ground state of the original problem is preserved, and therefore still favoured in the physical implementation of quantum annealing. This problem is known as the *parameter-*

setting problem [80, 56, 55]. In sampling, the scenario is different because all the 2^N vectors are significant as well as their corresponding probabilities. Hence, one seeks the value of auxiliary parameters that preserves the entire distribution defined in Eq. (2.6).

Previous research [56] suggests a possible mechanism underlying the parameter-setting problem. The authors investigated the Sherrington-Kirkpatrick model [87] and found that the optimal choice of the auxiliary parameters could be obtained by forcing both the spin glass and the subgraphs to cross the quantum critical point together during the annealing. Roughly speaking, the quantum phase transition happens when the energies associated with the problem-encoding system and the transverse field are of the same order of magnitude. This implies that the optimal auxiliary parameters are $\mathcal{O}(J_{SG}\sqrt{N})$, where N is the number of spins and J_{SG} is the typical value of the couplings, that is, the standard deviation (see Fig. 2 in Ref. [56]). However, the intuition provided by this study does not necessarily apply to realistic problems. In generative modeling, even when starting from a dense model, the learning process could lead to a sparse model that is substantially different from Sherrington-Kirkpatrick's.

Our solution uses the extended dataset defined in Eq. (2.8) to simultaneously address the parameter-setting problem and to guide the learning. Moreover, our algorithm implicitly correct for noise and defects on the control parameters, problems that are expected to affect any near-term quantum computer. This is achieved by adjusting *all* parameters in Eq. (2.9) to minimise the "error" made by the annealer in producing samples that resemble the dataset. In the next Section we discuss our solution in detail; let us now recap the QABM using the example illustrated in Figure 2.3.

In Panel (a), we define a complete graph for the logical variables, i.e., the four pixels shown in Panel (c). Using a suitable heuristic, we embed the graph into the quantum annealer in Panel (b). This introduces auxiliary variables and parameters. In this example, the logical variable 1 (red vertex) is mapped to two qubits, $1A$ and $1B$, which are then connected by an auxiliary coupler J_{11}^{AB} (red edge) to form a subgraph. Variable 2 is mapped similarly. The remaining couplers (black edges) represent interactions between different subgraphs. The embedding is also used to

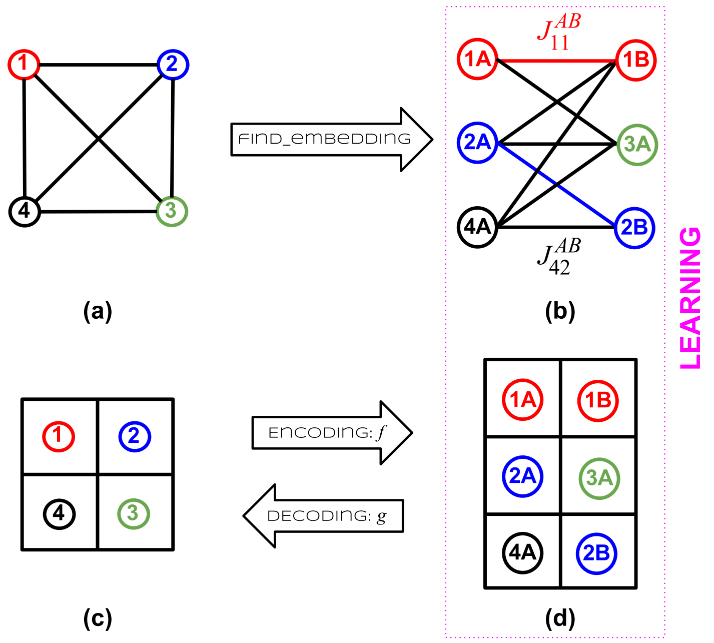


Figure 2.3: Example of a quantum-assisted Boltzmann machine. (a) Complete graph for the logical variables, i.e., the four pixels in (c). An heuristic embeds the graph into the interaction graph of the quantum annealer in (b) at the cost of introducing auxiliary variables and parameters. The embedding is also used to define an encoding f and a decoding g between the original data space in (c) and the extended data space in (d). The learning algorithm adjusts all the parameters tackling both the parameter-setting problem and the generative task.

define encoding and decoding mappings f and g, respectively, that transform between data space in Panel (c) and qubit space in Panel (d). The learning algorithm, which we describe next, adjusts all control parameters, hence automatically tackling both the parameter-setting and the generative modeling problems.

2.2.2 Learning algorithm

The QABM can be trained by minimising a suitable measure of distinguishability between mixed quantum state. Let $\rho_{\widetilde{\mathcal{D}}}$ be the diagonal density matrix whose diagonal elements encode the distribution of the extended dataset $\widetilde{\mathcal{D}}$. Let ρ be the Gibbs state in Eq. (2.3) approximately prepared by the quantum annealer. A particularly convenient measure of distinguishability in our case is the quantum relative entropy

$$S\left(\rho_{\widetilde{\mathcal{D}}} \middle\| \rho\right) = \mathrm{tr}\left(\rho_{\widetilde{\mathcal{D}}} \ln \rho_{\widetilde{\mathcal{D}}}\right) - \mathrm{tr}\left(\rho_{\widetilde{\mathcal{D}}} \ln \rho\right). \tag{2.11}$$

This quantity is non-negative, and it is zero if and only if $\rho = \rho_{\widetilde{\mathcal{D}}}$. The convenience stems from the logarithm in the second term which cancels the exponential in the definition of the Gibbs state.

We can now use gradient descent to minimise this quantity with respect to the control parameters. The *learning rule* based on gradient descent is given by

$$J_{ij}^{(kl)}(t+1) \longleftarrow J_{ij}^{(kl)}(t) - \eta \frac{\partial S}{\partial J_{ij}^{(kl)}}, \tag{2.12}$$

$$h_i^{(k)}(t+1) \longleftarrow h_i^{(k)}(t) - \eta \frac{\partial S}{\partial h_i^{(k)}}, \tag{2.13}$$

where t indicates the iteration number and $\eta > 0$ is the learning rate. Calculating the derivatives we obtain

$$-\frac{1}{\beta}\frac{\partial S}{\partial J_{ij}^{(kl)}} = \left\langle \hat{Z}_i^{(k)} \hat{Z}_j^{(l)} \right\rangle_{\rho_{\widetilde{\mathcal{D}}}} - \left\langle \hat{Z}_i^{(k)} \hat{Z}_j^{(l)} \right\rangle_{\rho}, \tag{2.14}$$

$$-\frac{1}{\beta}\frac{\partial S}{\partial h_i^{(k)}} = \left\langle \hat{Z}_i^{(k)} \right\rangle_{\rho_{\widetilde{\mathcal{D}}}} - \left\langle \hat{Z}_i^{(k)} \right\rangle_{\rho}. \tag{2.15}$$

Here, $\langle \cdot \rangle_{\rho_{\widetilde{\mathcal{D}}}}$ denotes the ensemble average with respect to the density matrix $\rho_{\widetilde{\mathcal{D}}}$ that involves only the data – this term is commonly referred to as the *positive phase*[4]. Similarly, $\langle \cdot \rangle_{\rho}$ denotes the ensemble average with respect to the density matrix ρ that exclusively involves the model – this term is usually called the *negative phase.*

We now discuss the learning rule in details. The Gibbs state approximately implemented by the annealer is characterised by the variables $\beta J_{ij}^{(kl)}$ and $\beta h_i^{(k)}$. Since the effective inverse temperature β is unknown, we wrote it to the left-hand side of Eqs. (2.14) and (2.15). Using the left-hand side as estimators for the partial derivatives in Eqs. (2.12) and (2.13) we observe that the learning takes place at an effective learning rate $\eta\beta$. Note that β may vary depending on the instance [62]. Our approach completely sidesteps the need to estimate it by absorbing it into a varying learning rate.

Recall that if $A(\tau^*) \ll B(\tau^*)$ in Eq. (2.1) during experiments, the annealer samples from a classical probability distribution. In this case, the minimisation

[4]The positive phases for all the parameters can be pre-calculated at the beginning of learning. This is a special case as there are no hidden variables in our model. In models with hidden variables this term cannot be pre-computed.

of the quantum relative entropy coincides with the maximisation of the classical log-likelihood. However, the quantum relative entropy is more general and the corresponding learning rules work also when the effective transverse field cannot be neglected. In fact, the transverse field could be treated as a further parameter to be learned (see Ref. [34] for a similar result). Unfortunately, state-of-the-art quantum annealers do not allow us to control the effective transverse field.

The exact computation of the statistics in the negative phase is a computational bottleneck due to the intractability of computing the normalisation constant of the Gibbs state. For classical distributions on simple graphs, there exist fast [88] approximations to the required statistics. For the complete graph considered here, the estimation is usually carried out by MCMC methods [77, 89]. Instead, our learning algorithm is designed on the working assumption that samples from quantum annealers are naturally described by Gibbs states. This assumption is reflected in that the second moment between two variables influences only the update of the coupling $J_{ij}^{(kl)}$ between them. If such a second moment increases (decreases), so does the corresponding coupling. This leaves open the possibility for the model to effectively self-correct for relatively small deviations from equilibrium, persistent biases, noise, and lack of precision, as long as the estimated gradient has a positive projection in the right direction, in the spirit of simultaneous perturbation stochastic approximation (SPSA) [90, 91, 68].

Equation (2.12) implies that couplings $J_{ii}^{(kl)}$ belonging to the same subgraph increase at a varying rate proportional to $1 - \left\langle \hat{Z}_i^{(k)} \hat{Z}_i^{(l)} \right\rangle_{\rho_{\widetilde{\mathcal{D}}}}$, which, in principle, leads to infinite values in the long term. However, the rate of growth decreases as the learning progresses since the samples generated by the annealer resemble the data more and more. Learning rules based on standard gradient-descent tend to produce large values for *all* the parameters as they push as much as possible towards a distribution with all the mass concentrated on the data vector. This problem is known as overfitting, and it is usually approached by *regularisation*. A possible regularisation method consists of penalising large parameters by adding a suitable term to Eq. (2.11). Another approach is to employ a stopping criterion based on some measure of generalisation, or predictive capability, evaluated at each iteration on a test dataset that was not used for learning. Under a proper choice of regularisation, the auxiliary parameters should not grow indefinitely anymore.

Finally, the learning rule can be interpreted as quantum *entropy maximisation* under constraints on the first- and second-order moments [92, 93, 94]. In Ref. [95], a maximum entropy approach was implemented on a quantum annealer in the context of information decoding, which is a hard optimisation problem. In contrast, we use a maximum entropy approach to learn a generative model, which is a hard machine learning problem.

Regarding the complexity of the algorithm, a complete graph with N logical variables has $\mathcal{O}(N^2)$ parameters. The embedding of this into a sparse graph like D-Wave's Chimera graph requires $\mathcal{O}(N^2)$ qubits [81]. However, the total number of parameters is still $\mathcal{O}(N^2)$. This occurs because when going from a dense graph to a sparse graph the number of auxiliary parameters is a small constant factor, e.g., each qubit in the Chimera graph interacts with six neighbours at most. In our experiments, this factor turned out to be ≈ 3 (see Table 2.1) which is a rather small computational overhead for learning the auxiliary parameters. This overhead can be neglected since the main bottleneck of the generative problem is still in the generation of samples which is at least as hard as any non-deterministic polynomial time problem (NP-hard)[46, 96, 97].

2.2.3 Implementation

2.2.3.1 Device and embeddings

We run experiments on the D-Wave 2X quantum annealer produced by D-Wave Systems and located at NASA Ames Research Center. D-Wave 2X processors are equipped with 1152 qubits interacting according to a Chimera graph. The minor-embedding schema in Ref. [81] can map a complete graph of 48 logical variables into an Chimera graph of this size. However, only 1097 qubits are functional and available in the D-Wave 2X hosted by NASA Ames Research Center. Due to this further restrictions on the connectivity of the processor, we expect the size of the largest graph that can be embedded to be reduced.

We resorted to the `find_embedding` heuristic [84] provided by the D-Wave programming interface. For graphs of size 15, 42 and 46 we ran the heuristic 500 times and selected the best embedding found. Table 2.1 shows details for each of the embeddings selected for our experiments.

Logical variables N	**Logical parameters** $N(N+1)/2$	**Physical variables** $\mathcal{O}(N^2)$	**Physical parameters** $\mathcal{O}(N^2)$
15	120	76	252
42	903	739	2644
46	1081	940	3389

Table 2.1: Main characteristics of the selected embeddings for complete graphs into the Chimera interaction graph of the D-Wave 2X. The number of required physical variables, i.e., qubits, is expected to scale as the square of the number of logical variables. As explained in the text, the number of physical parameters, i.e., couplings and local fields, is also expected to scale as the square of the number of logical variables. The overhead in terms of parameters, that is, the ratio between the number of physical and logical parameters, is a small constant factor of ≈ 3 in our case.

We judge the quality of an embedding not only by the total number of physical variables (i.e., qubits) needed, but also by the maximum subgraph size for each logical variable. For example, in the case of the 46-variable complete graph, we found an embedding with 917 qubits and a maximum subgraph size of 34. We selected, instead, an embedding with a larger number of qubits, 940, but with a considerably smaller maximum subgraph size of 28. Figure 2.4 shows the selected 46-variable embedding. Each logical variable is represented by qubits with the same label and edges of the same colour. Black edges describe interactions among subgraphs and, hence, among logical variables.

2.2.3.2 Datasets and pre-processing

We tested our ideas on the real OptDigits dataset [98], the synthetic Bars-and-Stripes (BAS) dataset [99], and synthetic Ising instances based on the Sherrington-Kirkpatrick model [87]. Table 2.2 summarises the characteristics of each dataset.

OptDigits is a real dataset of 8×8-pixel pictures of handwritten digits belonging to classes 'zero' to 'nine'. Using standard 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 picture d), we would need to embed a complete graph of 74 variables – 64 for the pixels and 10 for the class. We do not expect to be able to embed graphs of this size in the D-Wave 2X. Therefore, we removed the leftmost and rightmost columns as well as the bottom row from each picture, reducing the size to 7×6. We retained only pictures belonging to classes 'one', 'two', 'three' and 'four', reducing the requirements for one-hot encoding to four variables. The selected classes account for 1545 pictures in the training set and 723 pictures in the test set, and they are in almost equal proportion in both sets.

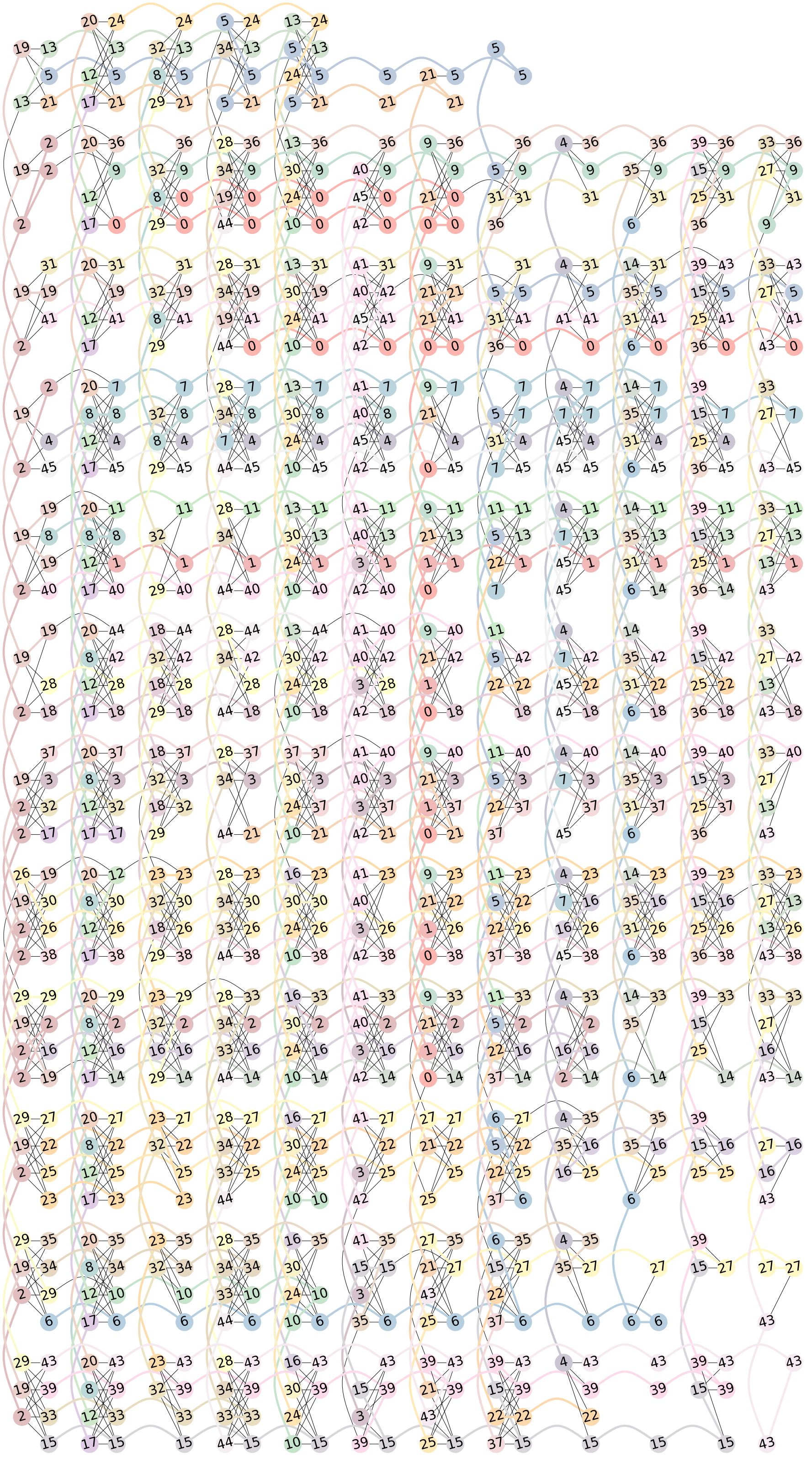


Figure 2.4: Minor-embedding of 46 logical variables into the Chimera interaction graph of the D-Wave 2X. This solution utilises 940 out of 1097 physical variables, that is, 86% of the available qubits. Subgraphs are characterised by the same number and the same colour.

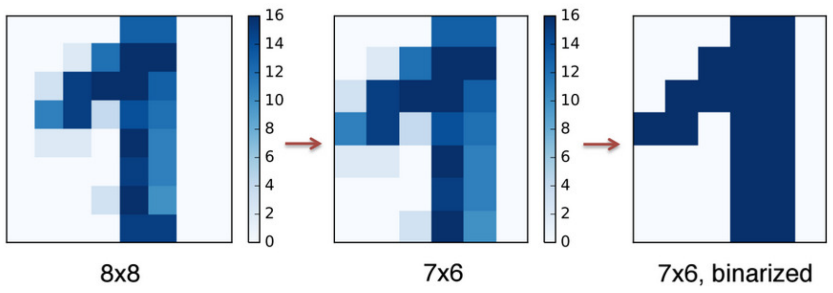


Figure 2.5: Pre-processing of the OptDigits dataset. The 8×8 pictures are cropped to 7×6 by removing columns from the left and the right, and by deleting a row from the bottom. The 4-bit grey scale is thresholded at the midpoint and binarized to $\{-1, +1\}$.

Finally, the original 4-bit grey scale of each pixel was thresholded at the midpoint and binarized to $\{-1, +1\}$ in order for the data to be represented by qubits in the D-Wave 2X. The pre-processing steps are illustrated in Fig. 2.5. Some pictures from the test set can be seen in Figure 2.6 (a).

Bars-and-Stripes (BAS) is a synthetic dataset of $N \times M$-pixel pictures generated by setting the pixels of each row (or column) to either black (-1) or white $(+1)$, at random. A reason to use this synthetic dataset is that it can be adapted to the number of available qubits in the D-Wave 2X. Having found an embedding for the 42-variable complete graph, we generated a 7×6 BAS dataset consisting of 190 pictures of 42 binary variables each. Then, we randomly shuffled the pictures and split the dataset into training and test sets of size 95 each. Figure 2.7 (a) shows some pictures from the test set.

For the *Ising* synthetic datasets, we preferred to work with instances of small size. We chose 10 random instances of the Sherrington-Kirkpatrick model with $N = 15$ logical variables. Parameters J_{ij} in Eq. (2.7) were sampled from a Gaussian distribution with mean $\mu = 0$ and standard deviation $\sigma = \zeta/\sqrt{N}$, parameters h_i were set to 0. A spin-glass transition is expected when $\zeta_c = 1$ in the thermodynamic limit, although finite-size corrections may become relevant for this small size. In order to obtain interesting instances, we chose $\zeta = 2$ and verified that the overlap distribution [100, 85] of each instance was indeed non-trivial. Moreover, we checked the performance of closed-form solutions provided by the mean-field technique in Ref. [101]. The method failed to produce (real-valued) solutions in 7 out of 10 instances, while it performed well in the remaining 3, adding further evidence that

Dataset	**Synthetic**	**Variables**	**Training vectors**	**Test vectors**
OptDigits	No	$42+4$	1545	723
BAS 7×6	Yes	42	95	95
Ising	Yes	15	150	Not applicable

Table 2.2: Main characteristics of the datasets used in experiments.

Domain	**Hyper-parameter**	**Value**
device	annealing time	$5\mu s$
	programming thermalisation	$1\mu s$
	readout thermalisation	$1\mu s$
	auto scale	False
learning	learning rate	0.0025
	$L2$ regularisation	10^{-5}
	momentum	0.5
	random initialisation	Unif$(-10^{-6}, +10^{-6})$

Table 2.3: Hyper-parameters used in all the experiments unless otherwise stated.

our instances had non-trivial features in their energy landscape. Finally, for each instance, we generated a training set of $D = 150$ samples by exact sampling from the Boltzmann distribution in Eq. (2.6) with $\beta = 1$.

2.2.3.3 Choice of hyper-parameters

We distinguish two kinds of hyper-parameters: those affecting the device operation and those affecting the learning rule. Device hyper-parameters affect the time needed to obtain samples from the D-Wave 2X. We set them to their corresponding minimum values in order to obtain samples as fast as possible.

Learning hyper-parameters come from the inclusion of $L2$ regularisation and momentum in Eqs. (2.12) and (2.13). These are standard techniques known to improve over standard gradient descent in terms of generalisation and convergence. We do not discuss these techniques further and refer to Ref. [89] for discussions about implementation and best practices. For these hyper-parameters, we tried a small grid of values and chose the value that would allow the quantum-assisted algorithm to produce visually appealing samples.

Finally, the parameter range allowed by D-Wave 2X is $J_{ij}^{(kl)} \in [-1, +1]$ and $h_i^{(k)} \in [-2, +2]$. We initialised all the parameters uniformly at random to rather small values in order to break the symmetry.

All the experiments were performed using the hyper-parameters shown in Table 2.3 unless otherwise stated.

2.2.4 Results

2.2.4.1 Reconstruction of pictures

The first task is meant to verify that the QABM is able to learn the joint probability distribution of the variables given a dataset. One way to do this is to check whether the model reconstructs corrupted pictures during learning.

To reconstruct a picture we first need to enforce the value of known pixels to their corresponding subgraphs, as previously illustrated in Fig. 2.1 (b). A strong local field $\pm h_{\text{max}}$ can be used to 'clamp' a qubit, that is, to constrain it to the desired value. Note that in quantum annealing a clamped variable behaves somewhat differently from its classical counterpart. Although the strong local field can substantially bias towards the desired value, the qubit remains a dynamical variable. In classical computation, a clamped variable is completely frozen. After sampling from the annealer, we assigned values to each corrupted pixel using the majority vote mapping in Eq. (2.10). To further mitigate the noise associated with this, we generated multiple reconstructions, 100 for each corrupted picture, and took a second majority vote over them. We believe this approach is very robust as we did not observe any mismatch between the clamped pixels and the corresponding reconstructions.

We chose to stop the learning algorithm as soon as any of the parameters went outside the dynamical range of the D-Wave 2X. Since the auxiliary parameters in a subgraph always increase, we expect these to be the first to get out of range.

In the first experiment, we used the QABM to model the 42 pixels of OptDigits without using the class information. A sample of the dataset is shown in Fig. 2.6 (a). Since the training set contains a relatively large number of data vectors, we opted for a minibatch learning approach [89], where 200 data vectors were used at each iteration to compute the positive phase of the gradient. The negative phase is computed on 200 samples from the D-Wave 2X. We trained for 6000 iterations, after which an auxiliary parameter went outside the dynamical range of the device.

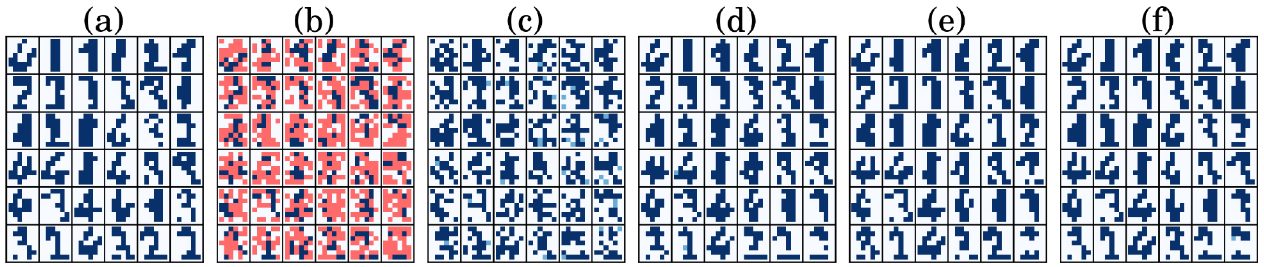


Figure 2.6: Reconstruction of OptDigits. (a) 36 pictures from the test set, with each pixel being either dark blue (+1) or white (−1). (b) A uniform salt-and-pepper noise shown in red corrupts each picture. (c)-(f) Reconstructions obtained after 1, 10, 1000, and 6000 learning iterations. A light blue pixel indicates a tie of the majority vote. Visually, we can verify that the model has learned to reconstruct digits.

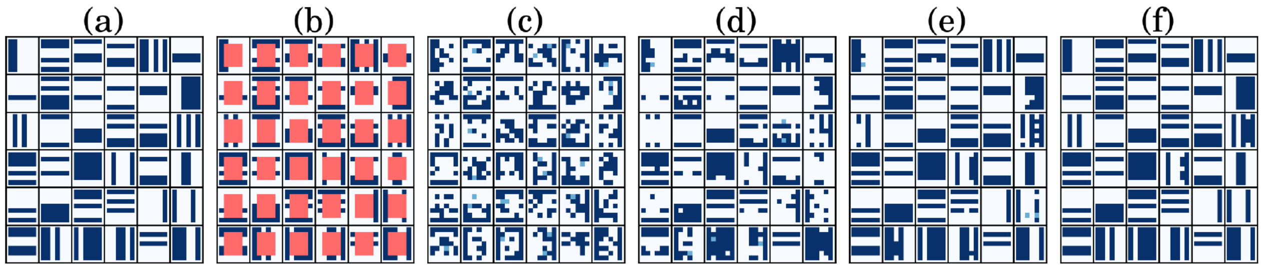


Figure 2.7: Reconstruction of Bars-and-Stripes. (a) 36 pictures from the test set, with each pixel being either dark blue (+1) or white (−1). (b) A 5×4 block of noise shown in red corrupts each picture. The remaining pixels contain enough information to reconstruct the pictures. (c)-(f) Reconstructions obtained after 1, 10, 1000, and 3850 learning iterations. The average number of mistaken pixels is 50% in (c), 18.6% in (d), 2.95% in (e), and finally 0.65% in (f).

To evaluate the model, we added a 50% uniformly distributed 'salt-and-pepper' noise (Fig. 2.6 (b), red pixels) to each picture in the test set and used the model to reconstruct it. It is not always possible to recover the exact original picture, and several reconstructions could be considered correct. Hence, we do not compute any error measure – we visually inspect the reconstructions instead. Figures 2.6 (c)-(f) show reconstructions obtained by models learned after 1, 100, 1000, and 6000 iterations, respectively. We observe that qualitatively good reconstructions are available since the early stages. However, the large degree of corruption in the original image gives rise to thicker reconstructions (Fig. 2.6 (f), third row, fourth column), thinner reconstructions (Fig. 2.6 (e), fourth row, second column), change of digit 'three' to 'one' (Fig. 2.6 (e), third row, fifth column), and other interesting phenomena.

We performed a similar test on BAS 7×6, a sample of which is shown in Fig. 2.7 (a). In this case we pre-computed the positive phase using all 96 training data vectors. Then, we ran the learning algorithm, and for each iteration, we computed the negative phase out of 96 samples obtained from the D-Wave 2X. The learning stopped at iteration 3850, after which an auxiliary parameter went outside the dynamical range of the device.

To evaluate the model, we blacked-out a 5×4 block (Fig. 2.7 (b), red pixels corresponding to 47.6% of the image) from each of the 96 test pictures and used the model to reconstruct it. We can observe from Fig. 2.7 (e) that reconstructed pictures are qualitatively similar to the original ones. For this dataset, it makes sense to report an error measure since the known pixels contain enough information for the exact recovery of the original picture. As a quantitative estimate of the quality, we computed the average number of mistaken pixels per reconstruction. After one iteration (Fig. 2.7 (c)), we obtained a rate of 10.45 mistakes out of 20 corrupted pixels, corresponding to about 50% performance as expected. The number of mistakes decreased to 3.73 (18.6%) after 100 iterations (Fig. 2.7 (d)), 0.59 (2.95%) after 1000 (Fig. 2.7 (f)), and finally 0.13 (0.65%) at the end of learning (Fig. 2.7 (e)). The latter result corresponds to almost perfect reconstruction. By definition, the test set is never used in the learning rule. Hence, these results provide evidence that the joint probability distribution of the pixels has been correctly modeled, and we can rule out a simple memorisation of the patterns.

2.2.4.2 Generation and classification of pictures

To investigate the generative and classification capabilities of the model, we introduced the one-hot encoding of 4 classes for a total of 46 logical variables. We trained this larger model on the OptDigits dataset, also including the classes.

For classification of a test picture, we clamped all the pixels $\boldsymbol{s}$ and sampled the four class variables $\boldsymbol{c} \in \{-1,+1\}^4$. We classified the picture as $c^* = \arg\max_c P(\boldsymbol{c}|\boldsymbol{s})$, where the probabilities are estimated using 100 samples.

After 6000 learning iterations, this simple procedure led to an accuracy of 90% on the test set. This is a significant result, given that a random guess achieves approximately 25% accuracy.

For the generative task, we clamped the class variable and sampled the pixels instead. Figure 2.8 shows samples obtained from the QABM along with human-generated pictures from the test set. Columns indicate digits 'one' to 'four'. Rows correspond to either human-generated pictures taken from the test set or machine-generated pictures. The Reader is invited to guess which is which. The solution is given in a footnote [5]. Machine-generated pictures are remarkably similar, though not identical, to those drawn by humans. Notice that human-generated digits may be ambiguous because of a variety of calligraphy styles encoded in low-resolution pictures. This ambiguity has been captured by the model.

2.2.4.3 Learning of an Ising model

We now compare models where the learning rule relies on quantum annealing (QA), simulated thermal annealing (SA) [102], and exact gradient. For this experiment, we use the synthetic Ising datasets of 15 logical variables.

This task is similar in spirit to the inverse Ising problem [103, 104, 101]. Differently from what is usually done in the literature, we do not quantify the quality of the model using the quadratic error of the parameters. This follows three reasons. First, the QABM has a larger number of parameters than the Ising model from which the datasets are generated, and a direct comparison is not straightforward.

[5] We have not performed the standard Turing test, where each pair of figures is shown in isolation. Ours is, in principle, a harder test for the machine as the redundancy of having all human- and machine-generated images together enhances the probability of the human spotting differences. This is compensated by the low resolution of the images, which might hint at an easier test for the machine, if shown one by one, given the distortion of the images. *Solution:* Blocks (a)-(d) show machine-generated pictures while blocks (e)-(h) show human-generated pictures.

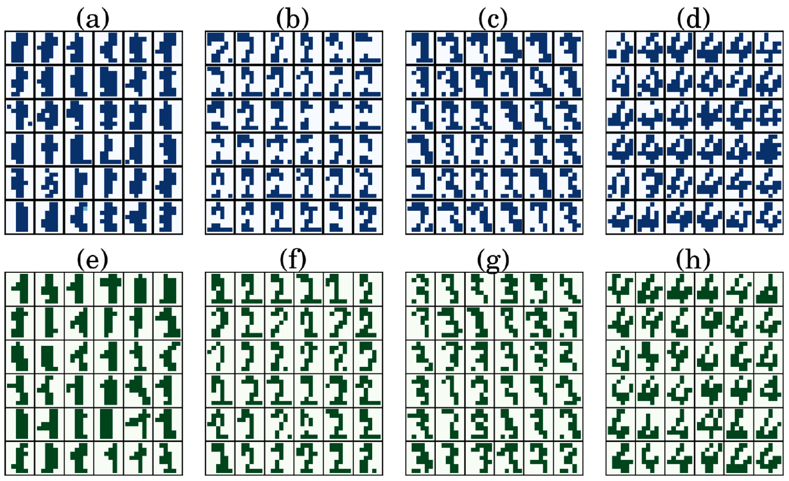


Figure 2.8: Visual Turing test. (a)-(h) The Reader is invited to distinguish between human-generated pictures and machine-generated pictures. Columns identify classes 'one' to 'four'. Rows identify the source, either 'human' or 'machine'. The solution is given in a footnote.

Second, we do not have access to the effective parameters implemented in the quantum annealer, unless we estimate the effective temperature, and that can introduce errors. Third, to our knowledge, there is no connection between generic distances in parameter space, as measured by the quadratic error, and distances in probability space, which are those having an actual operational meaning. For instance, it is known that, close to a critical point, a slight variation in the parameters can lead to drastically different probability distributions [105].

Our evaluation strategy exploits the fact that we have full knowledge of the true distribution $Q(\boldsymbol{s})$ that generated the data. At each learning iteration, we obtain a set $\mathcal{S} = \{\boldsymbol{s}^1, \ldots, \boldsymbol{s}^L\}$ of L samples from the QABM distribution $P(\boldsymbol{s})$ and evaluate the average log-likelihood that these were generated by $Q(\boldsymbol{s})$,

$$\Lambda_{\mathrm{av}}(\mathcal{S}) = \frac{1}{L} \sum_{\ell=1}^{L} \log Q\left(\boldsymbol{s}^{\ell}\right) \tag{2.16}$$

where, for simplicity, we chose $L = D = 150$. Note that $\Lambda_{\mathrm{av}}(\mathcal{S})$ is not expected to be maximised by the generated samples, but rather to match the value $\Lambda_{\mathrm{av}}(\mathcal{D})$ of the original dataset. We shall stress that Eq. (2.16) requires full knowledge of the distribution that generated the data. This is unfeasible for real datasets since the whole point of learning a model is precisely that we do not know the true underlying distribution.

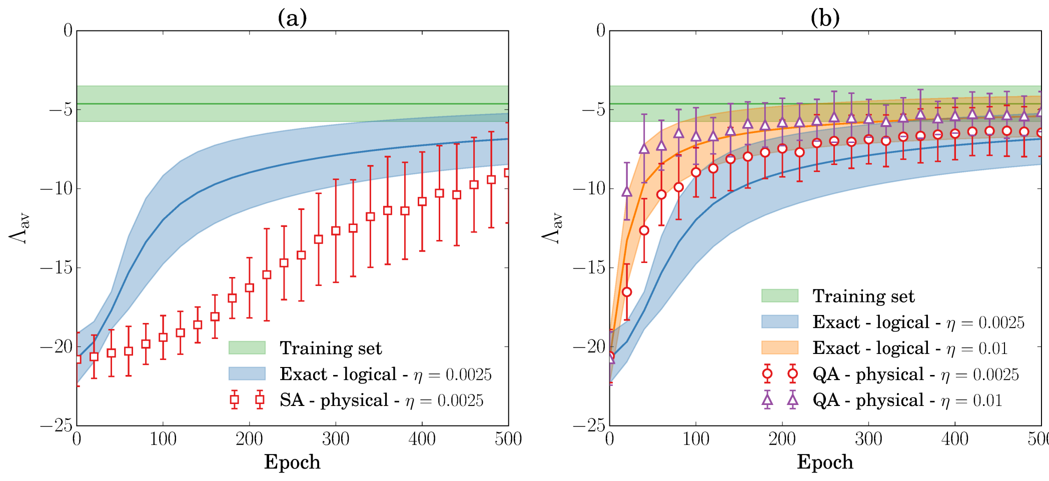


Figure 2.9: Mean and one standard deviation of the generalisation proxy Λ_{av} for 10 random instances and for different learning algorithms. An algorithm is considered successful if it matches the proxy of the training set (green band). (a) The 15-variable logical model trained with exact gradient (blue band) matches faster than the 76-qubit physical model trained with simulated annealing (red squares) when the same learning rate is used. This suggests that the larger number of parameters does not help the physical model. However, (b) quantum annealing on the physical graph (QA) matches even faster than exact gradient on the logical graph (Exact) when the same learning rates are used.

In this set of experiments, we performed 500 learning iterations and did not use gradient descent enhancements, such as momentum and regularisation, in order to simplify the quantitative analysis.

First, we verified whether the larger number of parameters in the physical graph provides a practical advantage. While exact gradient calculations are feasible for the 15-variable logical graph, they are unfeasible for the corresponding 76-qubit physical graph considered here (see embedding details in Table 2.1). We opted for a version of SA where each sample follows its own independent linear schedule, therefore avoiding the problem of auto-correlation among samples. We used a linear schedule $\beta(t) = t/t_{max}$ for the inverse temperature and performed a preliminary study in order to set the optimal number of Monte Carlo spin flips per sample, t_{max}. We incrementally increased this number and observed the learning performance via the proxy Λ_{av}. We chose $t_{max} = 15200$ Monte Carlo spin flips, as multiples of this number did not result in improved learning speed nor in better values of Λ_{av}. We expect this to be equivalent to learning by exact gradient, within the 500 learning iterations considered here.

Figure 2.9 shows mean and 1 standard deviation of the performance indicator for the 10 synthetic datasets considered here. Panel (a) shows that SA on the physical graph (red squares) results in slower learning than exact gradient on the logical graph (blue band) when the same learning rate $\eta = 0.0025$ is used. That is, even though both methods approach the optimal Λ_{av} of the dataset (green band), the larger number of parameters in the physical graph does not provide an advantage. Interestingly, Panel (b) shows that QA on the physical graph (red circles) does outperform exact gradient. This is an indication of the varying effective learning rate previously discussed in Section 2.2.2. Indeed, by increasing the learning rate of the exact gradient method to $\eta = 0.01$ (orange band), we were able to outperform QA. In turn, however, QA outperformed exact gradient if the same larger learning rate were used (purple triangles).

Because of the interplay between effective temperature and learning rate, the experiments presented here cannot confirm nor rule out the presence of quantum effects. Indeed, the fast initial learning could also be caused by a non-vanishing transverse field at the freeze-out time.

2.2.5 Discussion

Whether quantum annealers can improve algorithms that rely on sampling and provide more effective models are important open questions. State-of-the-art quantum annealers face several challenges that need to be overcome before we can address these questions from an experimental perspective. Besides the problem of temperature estimation, some of the most pressing challenges are the sparse qubit-to-qubit interaction graph, the low precision and limited range of the control parameters, and the different sources of noise.

By combining minor-embedding techniques and a data-driven setting of parameters, we improve the robustness and the complexity of machine learning models that can be assisted by quantum annealers. By requiring only partial information about the distribution from which the quantum annealer is sampling (i.e., the grey-box approach), we avoid the need for estimating temperature during learning and potentially mitigate the different sources of noise on the device. The resulting model is a quantum-assisted Boltzmann machine (QABM) with all pairwise interactions and no hidden variables. We validated the QABM qualitatively on the reconstruction, classification and generation of pictures, and quantitatively by computing a proxy for the generalisation ability.

An advantage of our approach is that the learning rules are agnostic to the embedding, which can therefore be obtained by either heuristic algorithms [84] or by known recipes [81, 106]. The learning rules can also be extended to other proposed quantum annealing architectures. For example, the Lechner-Hauke-Zoller schema (LHZ) [107] implements a complete graph of logical variables using physical qubits on a square-lattice geometry and requiring four-body interactions. In this case, the learning rules would rely on gradients similar to those in Eqs. (2.14) and (2.15), and with the inclusion of some quartic terms. Our QABM generative model would not face any additional challenge in this architecture. The question of whether the LHZ schema provides any advantage for generative modeling shall be addressed in future work.

From a more fundamental perspective, several key questions remain open. When and why does quantum annealing outperform classical Monte Carlo methods? When and why does quantum annealing provide more effective models? Our results show that the quantum-assisted model learns faster than the classical one during the initial stage, under the same setting of hyper-parameters. Given that an instance-dependent effective temperature can imply a varying learning rate, this faster learning is probably due to the quantum-assisted algorithm automatically adjusting its learning rate. It is important to investigate if such a learning schedule is optimal and, if so, whether it can be simulated classically. Yet, we cannot discard that non-trivial quantum effects played a role in our experiments; indeed, if the quantum annealer were sampling at fixed transverse field, our learning rules would still be valid.

In summary, we provided experimental evidence that quantum annealers can be used for complex machine learning tasks. The most important open question is whether there is a quantum advantage. Years of experience in benchmarking quantum annealing for combinatorial optimisation [108, 109, 58] suggests that the answer may not be straightforward. A benchmarking study for machine learning could follow well-established guidelines used in optimisation (see Ref. [110]), but the iterative nature of learning makes the task far more time-consuming. First, almost all the hyper-parameters (e.g., learning rate, annealing time, number of samples per iteration, etc.) should be optimised. Second, the study should be carried out on several datasets and for different system sizes in order to obtain acceptable statistics. If such a daunting study were to be performed, our QABM generative model would be an appealing choice, as the objective function in Eq. (2.11) is convex in the parameter space. This means that the performance of our algorithm mostly depends on the quality of the samples produced by the quantum annealer. This is in contrast with non-convex problems where the learning algorithm can find sub-optimal solutions even if samples are of the highest quality.

2.3 The quantum-assisted Helmholtz machine

2.3.1 Model definition and learning algorithm

In the previous Section we introduced the QABM generative model. Natural extensions of the QABM are the inclusion of latent variables, also known as hidden variables, and the support for continuous variables. Hidden variables are needed, for example, if data vectors require constraints that cannot be enforced by pairwise interactions alone [77]. Continuous variables are needed for a correct modeling of real datasets. In this Section we propose the quantum-assisted Helmholtz machine (QAHM), a generative model that implements these extensions.

Let us consider a dataset $\mathcal{S} = \{\boldsymbol{v}^1, \ldots, \boldsymbol{v}^D\}$ described by the empirical distribution $Q_{\mathcal{S}}(\boldsymbol{v})$. We seek a generative model $P(\boldsymbol{v}) = \sum_{\boldsymbol{u}} P(\boldsymbol{u}, \boldsymbol{v})$, where $\boldsymbol{u}$ is a set of binary hidden variables. We can write $P(\boldsymbol{u}, \boldsymbol{v}) = P(\boldsymbol{v}|\boldsymbol{u}) P_{QC}(\boldsymbol{u})$ and consider a prior distribution $P_{QC}(\boldsymbol{u}) = \langle \boldsymbol{u}|\rho|\boldsymbol{u}\rangle$ sampled by a quantum computer in the computational basis $\{|\boldsymbol{u}\rangle\}_{\boldsymbol{u}}$.

Let us focus on distributions implemented by quantum annealers. Using our framework, the prior corresponds to the diagonal elements of a Gibbs distribution $\rho = e^{-\beta\mathcal{H}}/\mathcal{Z}$, where $\mathcal{H}$ is the Hamiltonian, β is the inverse temperature and $\mathcal{Z}$ is the normalisation constant. Recall that we can implement Hamiltonians with pairwise interactions of the type

$$\mathcal{H} = -\sum_{(i,j)\in\mathcal{E}} J_{ij} \hat{Z}_i \hat{Z}_j - \sum_{i\in\mathcal{V}} h_i \hat{Z}_i - \Gamma \sum_{i\in\mathcal{V}} \hat{X}_i \tag{2.17}$$

where $\hat{Z}_i$ and $\hat{X}_i$ denote Pauli operators in the z and x direction, respectively, J_{ij} and h_i are control parameters, Γ is the residual transverse field, $\mathcal{V}$ is the set of qubits and $\mathcal{E}$ is the set of qubit-to-qubit interactions.

The conditional distribution $P(\boldsymbol{v}|\boldsymbol{u})$ stochastically translates samples from the quantum computer into samples on the domain of the data. Hence, $\boldsymbol{v}$ could be a vector of continuous variables, binary variables, or other objects. This is a significant advantage over the QABM where the visible variables are directly mapped to qubits.

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

$$\mathcal{L} = \sum_{\boldsymbol{v}} Q_{\mathcal{S}}(\boldsymbol{v}) \ln P(\boldsymbol{v}). \tag{2.18}$$

However, the learning of a Helmholtz machine [75] is based on the lower bound

$$\sum_{\boldsymbol{v}} Q_{\mathcal{S}}(\boldsymbol{v}) \ln P(\boldsymbol{v}) \geq \sum_{\boldsymbol{v},\boldsymbol{u}} Q_{\mathcal{S}}(\boldsymbol{v}) Q(\boldsymbol{u}|\boldsymbol{v}) \ln \frac{P(\boldsymbol{u},\boldsymbol{v})}{Q(\boldsymbol{u}|\boldsymbol{v})}, \tag{2.19}$$

where $Q(\boldsymbol{u}|\boldsymbol{v})$ is an auxiliary recognition network that approximates the intractable true posterior $P(\boldsymbol{u}|\boldsymbol{v})$ [6]. It is easy to see that the bound in Eq. (2.19) is tight for $Q(\boldsymbol{u}|\boldsymbol{v}) = P(\boldsymbol{u}|\boldsymbol{v})$.

The term $\ln \langle \boldsymbol{u}|\rho|\boldsymbol{u}\rangle$ arising from $\ln P(\boldsymbol{u},\boldsymbol{v})$ in Eq. (2.19) is intractable due to the projection of the Gibbs state on the computational basis states $|\boldsymbol{u}\rangle$. A bound for this term was derived in Ref. [34] using the Golden-Thompson inequality. Instead, we use a simpler bound based on Jensen's inequality (see Appendix 5.2 for a derivation)

$$\ln \langle \boldsymbol{u}|\rho|\boldsymbol{u}\rangle \geq \langle \boldsymbol{u}|\ln \rho|\boldsymbol{u}\rangle \,. \tag{2.20}$$

Combining Eqs. (2.19) and (2.20), we get the tractable lower bound

$$\mathcal{G}(\theta_G, \theta_{QC}) = \sum_{\boldsymbol{v},\boldsymbol{u}} Q_{\mathcal{S}}(\boldsymbol{v}) Q(\boldsymbol{u}|\boldsymbol{v}) \Big(\ln P(\boldsymbol{v}|\boldsymbol{u}) + \langle \boldsymbol{u}|\ln \rho|\boldsymbol{u}\rangle \Big), \tag{2.21}$$

where θ_G and θ_{QC} denote the parameters of generator network $P(\boldsymbol{v}|\boldsymbol{u})$ and Gibbs state ρ, respectively. Part of the learning algorithm consists in maximising this bound with respect to the parameters. In Eq. (2.21) we neglected terms that do *not* depend on either θ_G or θ_{QC}, as they vanish when computing the gradient of $\mathcal{G}$.

[6] The name Helmholtz machine comes from the minimisation of the non-equilibrium Helmholtz free energy which is contained in Eq. (2.19).

As mentioned before, the recognition network $Q(\boldsymbol{u}|\boldsymbol{v})$ has to closely approximate the true posterior $P(\boldsymbol{u}|\boldsymbol{v})$ throughout learning. Unfortunately, the maximisation of the lower bound in Eq. (2.19) with respect to the parameters of the recognition network is often intractable. The *wake-sleep algorithm* [74] attempts to bring $Q(\boldsymbol{u}|\boldsymbol{v})$ close to $P(\boldsymbol{u}|\boldsymbol{v})$ by minimising a more tractable function, that is, the Kullback-Leibler (KL) divergence

$$D_{KL}\left(P(\boldsymbol{u}|\boldsymbol{v})||Q(\boldsymbol{u}|\boldsymbol{v})\right) = \sum_{\boldsymbol{u}} P(\boldsymbol{u}|\boldsymbol{v}) \ln \frac{P(\boldsymbol{u}|\boldsymbol{v})}{Q(\boldsymbol{u}|\boldsymbol{v})}, \tag{2.22}$$

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

$$\mathcal{R}(\theta_R) = \sum_{\boldsymbol{u},\boldsymbol{v}} P(\boldsymbol{u},\boldsymbol{v}) \ln Q(\boldsymbol{u}|\boldsymbol{v}), \tag{2.23}$$

where θ_R denotes the parameters of the recognition network $Q(\boldsymbol{u}|\boldsymbol{v})$. In Eq. (2.23) we neglected terms that do not depend on θ_R, as they vanish when computing the gradient of $\mathcal{R}$.

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

$$P(\boldsymbol{v}|\boldsymbol{u}) = \sum_{\boldsymbol{u}^1,\dots,\boldsymbol{u}^L} P_0(\boldsymbol{v}|\boldsymbol{u}^1)P_1(\boldsymbol{u}^1|\boldsymbol{u}^2)\cdots P_L(\boldsymbol{u}^L|\boldsymbol{u}), \tag{2.24}$$

$$Q(\boldsymbol{u}|\boldsymbol{v}) = \sum_{\boldsymbol{u}^1,\dots,\boldsymbol{u}^L} Q_L(\boldsymbol{u}|\boldsymbol{u}^L)\cdots Q_1(\boldsymbol{u}^2|\boldsymbol{u}^1)Q_0(\boldsymbol{u}^1|\boldsymbol{v}), \tag{2.25}$$

in terms of L additional sets of hidden variables $\boldsymbol{u}^1,\dots,\boldsymbol{u}^L$ that connect the variables $\boldsymbol{v} \equiv \boldsymbol{u}^0$ in the visible layer with $\boldsymbol{u} \equiv \boldsymbol{u}^{L+1}$ in the last hidden layer. Note that here we use superscripts to index the layers. Using factorised layers of binary variables $u_i^\ell \in \{-1,+1\}$ (e.g., see architectures in Fig. 2.2), we have

$$P_\ell(\boldsymbol{u}^\ell|\boldsymbol{u}^{\ell+1}) = \prod_i \pi(u_i^\ell|\boldsymbol{u}^{\ell+1}; A^\ell, a^\ell), \tag{2.26}$$

$$Q_\ell(\boldsymbol{u}^\ell|\boldsymbol{u}^{\ell-1}) = \prod_i \pi(u_i^\ell|\boldsymbol{u}^{\ell-1}; B^\ell, b^\ell), \tag{2.27}$$

where

$$\pi(u_i|\boldsymbol{u}';C,c) = \left[1+e^{-2u_i\left(\sum_j C_{ij}u'_j+c_i\right)}\right]^{-1}. \tag{2.28}$$

In practice, all expectation values in Eqs. (2.21) and (2.23) are estimated from samples. The sampling process is carried out as follows; a configuration of hidden variables can be sampled from the annealer and propagated top-down through the generator network P to obtain an artificial data vector. Conversely, a data vector can be propagated bottom-up through the recognition network Q to obtain a configuration of hidden variables. However, in order to sidestep the need to sample from a quantum device for each data vector, here we employ a classical recognition network Q. To the best of our knowledge, this bottleneck is intrinsic in all the other quantum-assisted models with hidden variables (e.g., see the QADBN and QADBM in Fig. 2.2, or see Ref. [111] for another such proposals).

Let us now discuss the learning rules based on standard gradient ascent. The updates have structure $\theta^{(t+1)} \longleftarrow \theta^{(t)} + \eta\nabla_\theta\mathcal{F}$, where θ stands for the parameters being updated, η is the learning rate, ∇_θ is the differential operator, and $\mathcal{F}$ stands for either $\mathcal{G}$ or $\mathcal{R}$, accordingly.

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

$$-\frac{1}{\beta}\frac{\partial\mathcal{G}}{\partial J_{ij}} = \langle u_i u_j\rangle_Q - \langle u_i u_j\rangle_\rho, \tag{2.29}$$

$$-\frac{1}{\beta}\frac{\partial\mathcal{G}}{\partial h_i} = \langle u_i\rangle_Q - \langle u_i\rangle_\rho, \tag{2.30}$$

where $\langle\cdot\rangle_Q$ denotes expectation values with respect to $Q(\boldsymbol{u}) = \sum_{\boldsymbol{v}} Q(\boldsymbol{u}|\boldsymbol{v})Q_{\mathcal{S}}(\boldsymbol{v})$ and $\langle\cdot\rangle_\rho$ denotes those with respect to $P_{QC}(\boldsymbol{u}) = \langle\boldsymbol{u}|\rho|\boldsymbol{u}\rangle$. Here we used the property $\hat{Z}_i\,|u_i\rangle = u_i\,|u_i\rangle$. The partial derivatives for the generator network are

$$\frac{\partial\mathcal{G}}{\partial A^\ell_{ij}} = \langle u^\ell_i u^{\ell+1}_j\rangle_Q - \langle u^\ell_i\rangle_P\langle u^{\ell+1}_j\rangle_Q, \tag{2.31}$$

$$\frac{\partial\mathcal{G}}{\partial a^\ell_i} = \langle u^\ell_i\rangle_Q - \langle u^\ell_i\rangle_P, \tag{2.32}$$

and similarly for the recognition network

$$\frac{\partial \mathcal{R}}{\partial B^{\ell}_{ij}} = \langle u^{\ell}_i u^{\ell-1}_j \rangle_P - \langle u^{\ell}_i \rangle_Q \langle u^{\ell-1}_j \rangle_P, \tag{2.33}$$

$$\frac{\partial \mathcal{R}}{\partial b^{\ell}_i} = \langle u^{\ell}_i \rangle_P - \langle u^{\ell}_i \rangle_Q. \tag{2.34}$$

We now briefly discuss some alternatives and improvements that can be found in the literature of deep generative models. A generalisation of the wake-sleep algorithm, called reweighted wake-sleep, was introduced in Ref. [112]. 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. [113] 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 the gradient. Finally, there exists a contrastive version of the wake-sleep algorithm that was introduced in Ref. [73] 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 thermalisation.

These techniques require full knowledge of the parameters implemented in quantum hardware. Recall from Section 2.1 that this may not be available in noisy quantum computers and without error correction. Using techniques developed in Section 2.2, we now show that the standard wake-sleep algorithm can be used to train Helmholtz machines assisted by noisy quantum annealers.

2.3.2 Implementation

We demonstrate the QAHM 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. (2.17) defined on a sparse graph of qubit-to-qubit interactions. In particular, the device is designed to exploits quantum tunneling to sample low-energy states at transverse field $\Gamma \approx 0$. As previously discussed, non-trivial non-equilibrium effects may make samples deviate from the corresponding classical Gibbs distribution. This scenario requires some engineering of the QAHM similar to that of the QABM. We would like to stress that the algorithm can be carried out on other quantum annealing architectures [107, 58], 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.

As done in Section 2.2, we use a grey-box approach so that we can update the parameters of the quantum annealer without the need to estimate deviations from the Gibbs distribution. Recall that 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 optimisation algorithms [114].

We now consider a complete graph describing the prior distribution $P_{QC}(\boldsymbol{u})$ over the hidden variables in the deepest layer. This connectivity is 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. Once again this expansion needs not be globally optimal, and can be found efficiently using minor-embedding heuristic techniques. The new dynamics is approximately described by the Hamiltonian

$$\widetilde{\mathcal{H}} = -\frac{1}{2}\sum_{i,j=1}^{N}\sum_{k,l=1}^{Q_i,Q_j} J_{ij}^{(kl)}\hat{Z}_i^{(k)}\hat{Z}_j^{(l)} - \sum_{i=1}^{N}\sum_{k=1}^{Q_i} h_i^{(k)}\hat{Z}_i^{(k)} - \Gamma\sum_{i=1}^{N}\sum_{k=1}^{Q_i}\hat{X}_i^{(k)}, \tag{2.35}$$

where N is the number of hidden variables in the deepest layer, which equals the number of subgraphs realised in hardware, Q_i is the number of qubits in 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. Recall 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 $\frac{1}{2}$ is required to avoid double counting. The partial derivatives required to learn the control parameters are similar to those in Eqs. (2.29) and (2.30), except that here we use physical variables $z_i^{(k)}$ such that $\hat{Z}_i^{(k)}\left|z_i^{(k)}\right\rangle = z_i^{(k)}\left|z_i^{(k)}\right\rangle$, instead of using logical variables u_i.

The model is then equipped with two functions that map back and forth between logical and qubit spaces. Similarly to the QABM we use a *copy* and a *majority vote* mappings

$$z_i^{(k)} = f(\boldsymbol{u}, i) = u_i \quad (\text{for } k = 1, \ldots, Q_i), \tag{2.36}$$

$$u_i = g(\boldsymbol{z}, i) = \operatorname{sign}\left(\sum_{k=1}^{Q_i} z_i^{(k)} \right). \tag{2.37}$$

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 $\boldsymbol{v}$ and two hidden layers $\boldsymbol{u}^1$ and $\boldsymbol{u}^2$, like the one shown in Figs. 2.10 (a) and (b). In the recognition network, the hidden variables $\boldsymbol{u}^2$ get copied into higher-dimensional vectors $\boldsymbol{z}$ (copies are shown with the same colour). 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 $\boldsymbol{z}$ from a Gibbs distribution. Samples are mapped back to the hidden variables $\boldsymbol{u}^2$ using the majority vote over subgraphs (subgraphs are shown with the same colour). Then, a top-down pass is used to sample the visible variables $\boldsymbol{v}$. Hence, every directed and undirected edge in Fig. 2.10 can be trained, except for the grey-coloured directed edges corresponding to the fixed mappings in Eqs. (2.36) and (2.37).

Now, because we do not 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 2.3.1 for a brief summary). We shall stress that this limitation is specific to our case-study and may not be present in fault-tolerant quantum computers.

Improved and faster learning can also be obtained by initialising the approximate posterior $Q(\boldsymbol{u}|\boldsymbol{v})$ close to true posterior $P(\boldsymbol{u}|\boldsymbol{v})$ when $\boldsymbol{v}$ is sampled from the dataset. This initialisation, also called *pre-training*, is often carried out by some greedy approximate algorithm [73, 113]. In principle, we could use pre-training to initialise all the trainable edges of our model in Fig. 2.10. We decided not to carry out pre-training in our small scale experiment as it could initialise the model to an almost-optimal configuration, hence hiding any contribution of the quantum device.

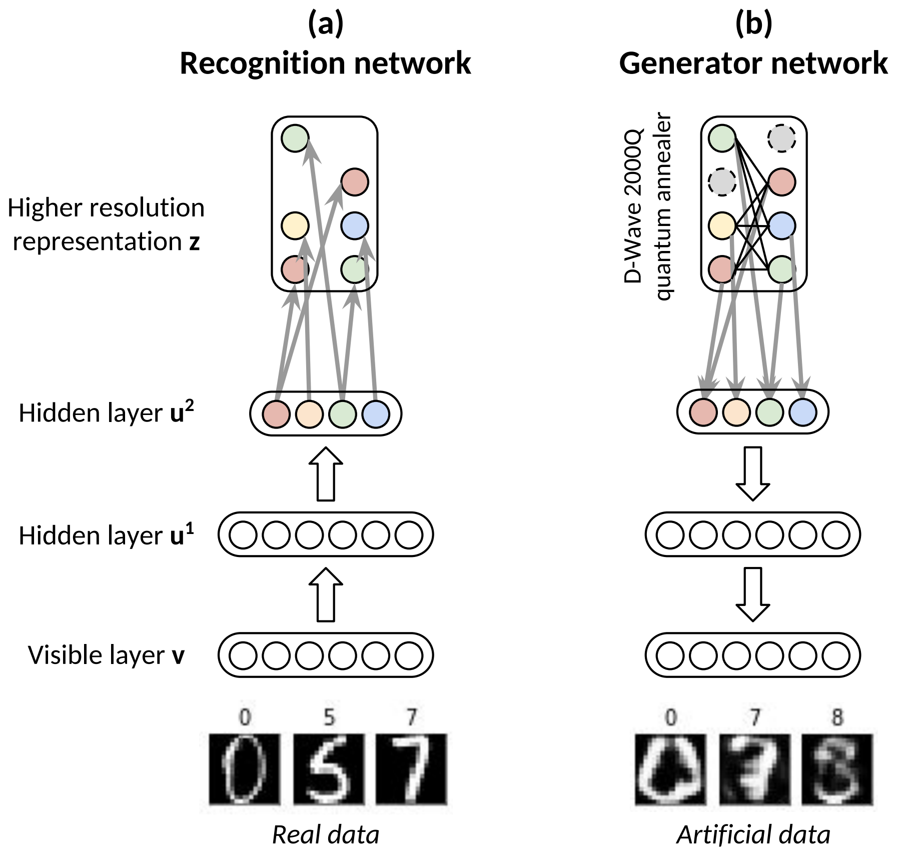


Figure 2.10: Experimental implementation of the QAHM on the D-Wave 2000Q quantum annealer for the sub-sampled MNIST dataset. The visible layer consists of 256 continuous variables $\boldsymbol{v}$ that encode the 16×16 grey-scale pixels, plus 10 binary variables that encode the class. There are two hidden layers, $\boldsymbol{u}^1$ and $\boldsymbol{u}^2$, with 120 and 60 binary variables, respectively. The variables $\boldsymbol{u}^2$ are effectively connected all-to-all through an embedding into 1644 qubits, $\boldsymbol{z}$, of the quantum annealer. The recognition network (a) is entirely classical to avoid calling the quantum device for each of the 7291 images in the dataset. The generator network (b) samples the deepest layer from the quantum annealer. The correspondence between recognition and generator networks is enforced by two mappings, here represented by grey-coloured edges.

For the reasons outlined above, we acknowledge that the standard wake-sleep algorithm may be slow and sub-optimal (this is further discussed in Section 2.3.4). The wake-sleep algorithm for Helmholtz machines on quantum annealers is summarised in Algorithm 1.

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

```
Require: an heuristic to embed in hardware a complete graph corresponding to
    the deepest hidden layer, mappings $f(\boldsymbol{u},i)$ and $g(\boldsymbol{z},i)$ from hidden variables to
    qubits and back
1: for number of learning iterations do
2:     sample $(\boldsymbol{v}^d,\boldsymbol{u}^d,\boldsymbol{z}^d)$ where $(\boldsymbol{v}^d,\boldsymbol{u}^d) \sim Q(\boldsymbol{u}|\boldsymbol{v})Q_{\mathcal{S}}(\boldsymbol{v})$ and $z^d_i = f(\boldsymbol{u}^d,i)$
3:     sample $(\boldsymbol{v}^k,\boldsymbol{u}^k,\boldsymbol{z}^k)$ where $\boldsymbol{z}^k \sim \langle \boldsymbol{z}|\rho|\boldsymbol{z}\rangle$, $u^k_i = g(\boldsymbol{z}^k,i)$ and $\boldsymbol{v}^k \sim P(\boldsymbol{v}|\boldsymbol{u}^k)$
4:     estimate $\nabla_\theta \mathcal{G}$ and $\nabla_\theta \mathcal{R}$ from samples
5:     update $\theta^{(t+1)}_{\mathcal{G}} = \theta^{(t)}_{\mathcal{G}} + \eta \nabla_\theta \mathcal{G}$
6:     update $\theta^{(t+1)}_{\mathcal{R}} = \theta^{(t)}_{\mathcal{R}} + \eta \nabla_\theta \mathcal{R}$
7:     decrease $\eta$
8: end for
```

2.3.3 Results

We tested our ideas on a sub-sampled version of the MNIST handwritten digits dataset [115]. Our training set consists of 7291 images of 16×16 grey-scale pixels, and a categorical variable indicating the corresponding class, 'zero' to 'nine'. First, we rescaled pixels to take real-values in $[-1,+1]$. Second, we used a one-hot encoding for the class (i.e., $c^d_i = -1$ for $i \neq j$, $c^d_j = +1$ 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 [84] to embed a complete 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 mappings in Eqs. (2.36) and (2.37) were set up accordingly.

Figure 2.10 shows the model composed of two networks and a quantum annealer implementing a prior over the second hidden layer $\boldsymbol{u}^2$. To implement the continuous variables $\boldsymbol{v}$ in the generator network, we used a deterministic layer of hyperbolic tangents $v_i = \tanh\left(\sum_j A^0_{ij} u^1_j + a^0\right)$ which is compatible with our rescaling of the pixels in the interval $[-1,1]$. Alternatively, one could use stochastic Gaussian variables and a different, compatible, rescaling. The final model is an engineered version of the model shown in Fig. 2.2 (a).

We ran the wake-sleep algorithm for 500 iterations with a learning rate of $\eta = 0.005$ for all the gradient updates. Subsequently, we trained for other 500 iterations by linearly decreasing the learning rate down to $\eta = 0.0005$. At each learning iteration, we inferred hidden configurations from the recognition network for all the data vectors in the training set, and sampled 1000 artificial data vectors from the generator network. These two sets are used to compute gradients as in Algorithm 1.

D-Wave hyper-parameters such as annealing time, programming thermalisation and readout thermalisation 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 system evolves towards the programmed Hamiltonian in Eq. (2.35). The use of the minimum annealing time is a well established practice due to extensive benchmarking by the combinatorial optimisation 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 grey-box approach considered here does not require knowledge of the exact form of the distribution, we chose the minimum annealing time of $5\mu s$.

Figure 2.11 (a) shows samples from the generator network after learning. For each of those, Panel (b) 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 generalisation capabilities of the model. Although these preliminary results cannot compete with state-of-the-art generative models, the generated data often resemble digits written by humans. Indeed, the problem of generating blurry artificial images affects many other approaches and only the recent development of generative adversarial networks [116] has led to much sharper artificial images.

Finally, Fig. 2.10 (c) shows some artificial samples along with their most prob-

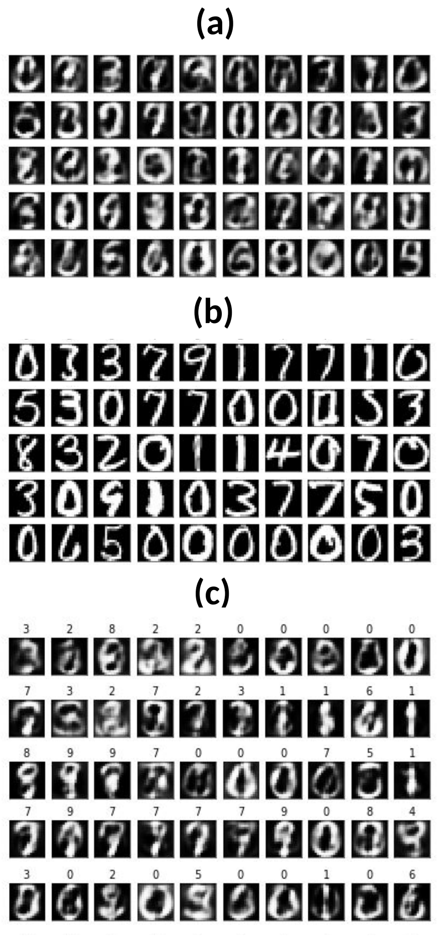


Figure 2.11: (a) Artificial images obtained from the learned generator network. (b) Images in the training set that are closest in Euclidean distance to the artificial ones in (a). This shows that the artificial images are not merely copies of the training images. (c) Additional artificial images along with their most probable class according to the model. Visually, the quantum-assisted model seems to correlate class and pixels most of the time.

able class according to the model. Visually, the model seems to correlate class and pixels most of the time. The process can be easily generalised to perform classification, where test images are provided through the recognition network and the most likely class is inferred through the generator network.

2.3.4 Discussion

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 grey-scale pictures of 16×16 pixels. In our case study, we used a noisy quantum annealer to learn a prior distribution for the latent variables, also known as hidden variables, of a deep generative model.

Here, we summarise some of the advantages and challenges with the current implementation of the quantum-assisted Helmholtz machine (QAHM), and we propose some generalisations for future work.

Advantages of the QAHM:

- A classical recognition network is used to perform approximate inference. There is no need to sample from a quantum device for each data vector 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 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 optimises the wrong cost function [74]. Solutions found in the literature [76, 78] require full knowledge of the model's parameters which is not available under the grey-box approach employed here.
- The recognition network has to be expressive enough to closely approximate the true posterior. As pointed out in the original work on Helmholtz machines [74], factorised 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) [76]. 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.

Potential generalisations:

- The mappings in Eqs. (2.36) and (2.37), used here to translate information from and to quantum hardware, can be relaxed into trainable stochastic functions. In this case, variables $\boldsymbol{z}$ in the recognition network and $\boldsymbol{u}^2$ 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 $\boldsymbol{u}'$ of the previous layer, is described by the hyperbolic tangent function $\mathbb{E}[u_i|\boldsymbol{u}'] = \tanh\Big(c_i + \sum_j C_{ij} u'_j\Big)$. When $C_{ij} \gg 1$ and $c_i = 0$, this function implements a majority vote of the variables in the previous layer. The copy mapping can be thought of as a majority vote over a single qubit in the previous layer. Hence, by learning all the parameters c_i and C_{ij} one obtains a generalised version of the QAHM implemented here. While this generalisation requires fitting additional parameters, it has the potential to discover better minor-embeddings than those found via heuristics.

- The QAHM allows to use quantum devices in both the recognition and the generator networks (see Fig. 2.2 (a)). The motivation for using the quantum device only in the generator network is to drastically reduce the number of calls to the device. It is an open question whether using the quantum device in the recognition network can enhance the generative model.

Chapter 3

Gate-based generative models

3.1 The framework

Developments in material science, hardware manufacturing, and disciplines such as error-correction and compilation, have brought us one step closer to the era of large-scale, fault-tolerant, universal quantum computation. However, the process is incremental and may take years; existing gate-based quantum hardware implements few physical qubits and can perform short sequences of gates before sources of noise take over. In such a setting, much anticipated algorithms such as Shor's are way out of reach. Yet it has been argued that noisy intermediate-scale quantum (NISQ) devices may find useful applications and commercialisation in the next few years [117, 1]. As prototypes of quantum computers are made available to researchers for experimentation, algorithmic development is indeed adapting to the pace at which quantum hardware is developed.

Parameterized quantum circuits (PQCs) offer a concrete way to implement algorithms and demonstrate quantum supremacy in the NISQ era. PQCs are typically composed of *fixed* gates, e.g., controlled NOTs, and *adjustable* gates, e.g., qubit rotations. Even at low circuit depth, some classes of PQCs are capable of generating highly non-trivial outputs. For example, under well-believed complexity-theoretic assumptions, the class of PQCs called instantaneous quantum polynomial-time (IQP) cannot be efficiently simulated by classical resources (see Lund *et al.* [38] and Harrow and Montanaro [39] for accessible reviews of quantum supremacy proposals). The demonstration of quantum supremacy is an important milestone in the development of quantum computers. In practice, however, we would like to obtain a quantum advantage over classical computing while attacking real-world problems.

The main approach taken by the community consists of formalising problems of interest as variational optimisation problems and use a combination of quantum and classical hardware to find approximate solutions. The intuition is that by outsourcing parts of the algorithm to classical hardware, we significantly reduce the burden on the quantum hardware. In particular, we reduce the required coherence time, circuit depth, and number of qubits, therefore allowing the NISQ hardware to focus entirely on the computationally hard part of the problem.

This hybrid algorithmic approach turned out to be successful in attacking scaled-down problems in chemistry, combinatorial optimisation and machine learning. For example, the variational quantum eigensolver (VQE) [118] has been used for searching the ground state of the electronic Hamiltonian of molecules [119, 120]. Similarly, the quantum approximate optimisation algorithm (QAOA) [121] has been used to find approximate solutions of classical Ising models [122] and clustering problems formulated as MaxCut [123].

The high-level approach is illustrated in Figure 3.1 and is made of three main components: the human, the classical computer, and the quantum computer. The human interprets the problem information and selects an initial model to represent it. The available data is pre-processed on a classical computer to determine a set of parameters for the PQC. The quantum hardware prepares a quantum state as prescribed by the PQC and performs measurements. Measurement outcomes are post-processed by the classical computer to generate a forecast. To improve the forecast, the classical computer implements a learning algorithm that updates the set of parameters. The overall algorithm is run in a closed loop between the classical and quantum devices which comprise the hybrid system. The human supervises the process and uses forecasts towards the goal.

To the best of our knowledge, the earliest hybrid systems were proposed for the task of learning quantum algorithms. In 2008, Bang *et al.* [124] described a method where a classical computer controls the unitary operation implemented by a quantum device. Each execution of the quantum device is deemed as either a 'success' or 'failure', and the classical learning algorithm adjusts the unitary operation towards its target. Starting from a dataset of input-output pairs their simulated system learned an equivalent of Deutsch's algorithm for finding whether a function is constant or balanced. In the same year, Gammelmark and Mølmer [125] described

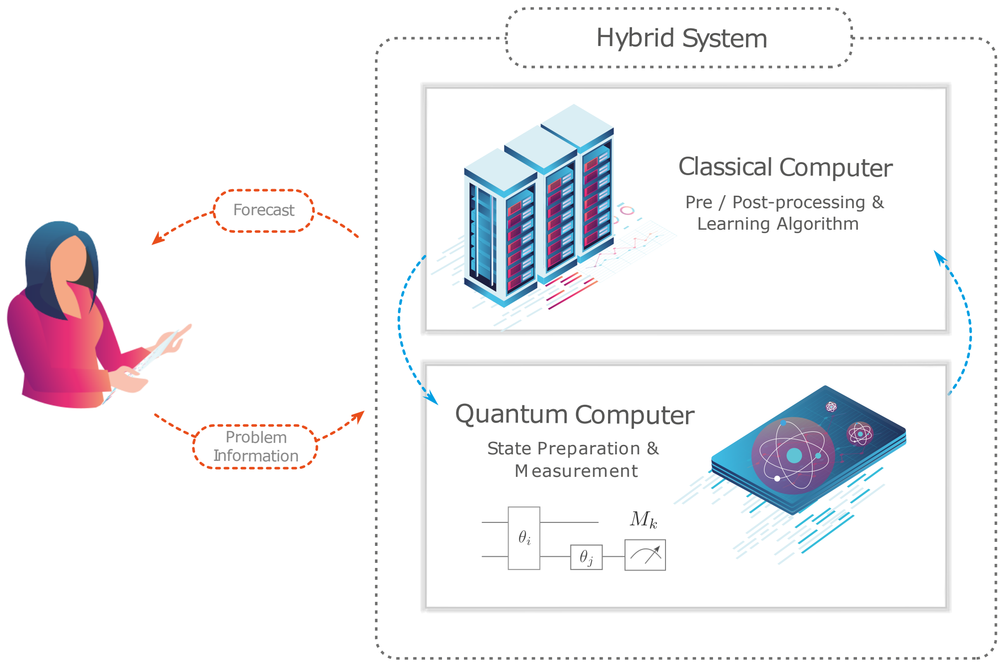


Figure 3.1: High-level depiction of hybrid algorithms used for machine learning. The role of the human is to use the problem information to setup the model, assess the learning progress, and use the forecasts. Within the hybrid system the quantum computer prepares quantum states according to a set of parameters. Using the measurement outcomes, the classical learning algorithm adjusts the set of parameters in order to minimise an objective function. The parameters, now defining a new quantum circuit, are fed back to the quantum hardware in a closed loop.

a more general approach where the parameters of the quantum system are quantized as well. In their simulations they successfully learned Grover's and Shor's algorithms.

These early proposals attacked problems that are well known within the quantum computing community, but much less known among machine learning researchers. More recently though, the hybrid approach based on PQCs has been shown to perform well on machine learning tasks such as classification, regression, and generative modeling, and has been applied to both classical and quantum data of small scale. In Appendix 5.3 we report the machine learning experimental demonstrations that have been carried out to date. This success is in part due to some similarities between PQCs and celebrated classical models such as kernel methods and artificial neural networks [126, 127].

Let us now briefly recall what unsupervised generative modeling is about. It is a machine learning task where the goal is to model an unknown probability distribution and generate artificial data accordingly. More formally, the task is to learn a model distribution $q_{\boldsymbol{\theta}}$ that is close to a target distribution p. The closeness is defined in terms of a divergence D on the statistical manifold, and learning consists of minimising this divergence. For a parameterized model

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} D(p, q_{\boldsymbol{\theta}}). \tag{3.1}$$

Since the target probability distribution is unknown, it is approximated using a dataset $\mathcal{D} = \{\boldsymbol{v}^{(i)}\}_{i=1}^{N}$ which we have access to and whose data vectors $\boldsymbol{v}^{(i)}$ are distributed according to the target distribution p. As an example, each data vector could be a natural image taken from the Internet.

The probabilistic nature of quantum mechanics suggests that a model distribution can be encoded in the wave function of a quantum system [128, 129]. Our proposal, the *quantum circuit Born machine* (QCBM), uses a PQC to prepare such a wave function. This model will be the focus of Section 3.2, but let us briefly see how it works.

We assume our computer to be a closed quantum system[1]. With n qubits, the state is described by a unit vector in a complex inner product vector space $\mathbb{C}^{2^n}$. The computation always starts with an easy to prepare state of the computational basis, for example the product state $|0\rangle^{\otimes n}$. A unitary operator $U_{\boldsymbol{\theta}}$ parameterized by a vector $\boldsymbol{\theta}$ is applied to the initial state producing a new state $U_{\boldsymbol{\theta}}|0\rangle^{\otimes n}$. The value of an observable quantity can be measured and physical observables are described by Hermitian operators. Let $M = \sum_i \lambda_i P_i$ be the Hermitian operator of interest, where λ_i is the i-th eigenvalue and P_i is the projector on the corresponding eigenspace. The *Born rule* states that the outcome of the measurement corresponds to one of the eigenvalues and follows probability distribution $p(\lambda_i) = \mathrm{tr}\Big(P_i U_{\boldsymbol{\theta}} |0\rangle\langle 0| U_{\boldsymbol{\theta}}^\dagger\Big)$. Plugging this in the definition of expectation values we obtain

$$\langle M \rangle = \sum_i \lambda_i p(\lambda_i) = \mathrm{tr}\Big(M U_{\boldsymbol{\theta}} |0\rangle\langle 0| U_{\boldsymbol{\theta}}^\dagger\Big). \tag{3.2}$$

[1]While this is not strictly true in real hardware, machine learning has the potential to cope with some types of noise and deviations from the ideal system. Similar arguments were made in Chapter 2 for quantum annealers. The difference is that there we exploited the open character of the quantum system, here we don't.

which can be estimated from repeated measurements[2] on copies of the state $U_{\boldsymbol{\theta}} |0\rangle^{\otimes n}$.

Now, let us focus on the set of expectation values $\{\langle M_{\boldsymbol{v}} \rangle\}_{\boldsymbol{v}}$ where $M_{\boldsymbol{v}} = |\boldsymbol{v}\rangle\langle\boldsymbol{v}|$ are projectors for the bitstrings. Clearly, these define a generative model for the bitstrings

$$q_{\boldsymbol{\theta}}(\boldsymbol{v}) = \operatorname{tr}\Big(M_{\boldsymbol{v}} U_{\boldsymbol{\theta}} |0\rangle\langle 0| U_{\boldsymbol{\theta}}^{\dagger}\Big). \tag{3.3}$$

Here, learning consists of matching the statistics generated by the circuit and those found in the dataset, carrying out the optimisation problem in Eq. (3.1).

Previous implementations of Born machines [130, 129, 131, 132] often relied on the construction of tensor networks and their efficient manipulation through specialised classical hardware such as graphical-processing units. Our work differs in that Born's rule is naturally implemented by a quantum circuit and executed on a NISQ hardware.

From the application point of view, we suggest the use of QCBMs as a benchmarking tool for hybrid systems. Quantum volume [133, 122] has been proposed as an architecture-neutral metric based on the task of approximating specific classes of random circuits. This is very general and it is indeed useful for estimating the computational power of a quantum computer. We propose the *qBAS score*, a complementary metric based on the generative modeling performance on a canonical synthetic dataset which is easy to generate, visualise, and validate for sizes up to hundreds of qubits. Implementing a low-depth circuit that can uniformly sample such data is 'hard' in the sense that it requires large amounts of entanglement. Any miscalibration or environmental noise will therefore affect this single performance number. The score enables comparison between different devices or across different generations of the same device. As the score depends also on classical resources, hyper-parameters, and design choices, we expect it to be a good choice for the assessment of the hybrid system as a whole.

Now, it is important to realise that one does not have access to the wave function generated by the circuit. Thus, the characterisation of $q_{\boldsymbol{\theta}}$ in Eq. (3.3) may be intractable for all but the smallest circuits. QCBMs belong to the class of *im-*

[2]The number of repetitions required for the estimation is determined by the desired precision as well as by the variance $\operatorname{Var}(M) = \langle M^2 \rangle - \langle M \rangle^2$. We won't discuss estimation methods here.

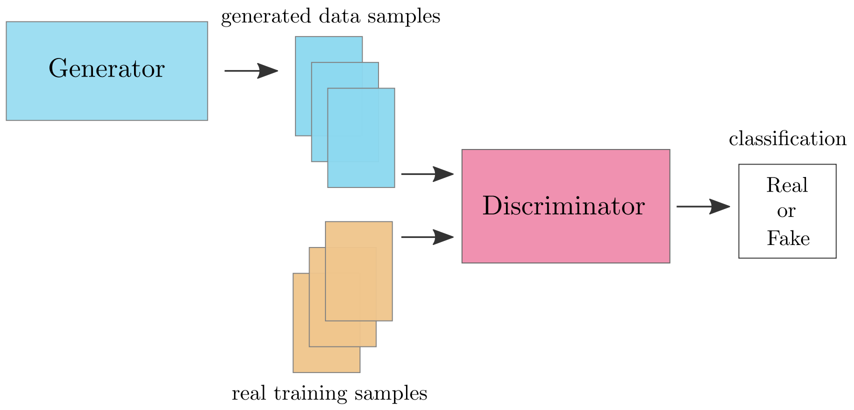


Figure 3.2: Illustration of the adversarial method for generative modeling. The generator produces artificial samples and the discriminator tries to distinguish between the artificial and the real samples. The network is trained until the artificial samples are indistinguishable from the training samples. The target data, the generator, and the discriminator can all be made quantum or classical.

plicit models, models where it is easy to obtain a sample $\boldsymbol{v} \sim q_{\boldsymbol{\theta}}$, but it is hard to estimate the likelihood of a sample $q_{\boldsymbol{\theta}}(\boldsymbol{v})$. The machine learning community has become increasingly interested in implicit models because of their generality, expressive power, and success in practice [134]. However, this poses a challenge to the design of suitable loss functions.

Differentiable loss functions are often hard to design when one does not have access to the likelihood. Non-differentiable loss functions are often hard to optimise; one may resort to gradient-free methods, but these are likely to struggle as the number of parameters becomes large. Employing *adversarial* methods from deep learning [116] we can potentially overcome these limitations. Figure 3.2 (a) shows the intuition; the adversarial method introduces a discriminative model whose task is to distinguish between true data coming from the dataset and artificial data coming from the generative model. This creates a 'game' where the two players, i.e., the models, compete. The intuition is that if a generator is able to confuse a perfect discriminator, then it means it can generate realistic artificial examples, hence solving the generative problem. The advantage is that both models are trained at the same time, with the discriminator providing a differentiable loss function for the generator.

Recently, Lloyd and Weedbrook [135] put forward the theoretical framework of the *quantum generative adversarial network* (QGAN) and suggested variants where target data, generator and discriminator are either classical or quantum. In Section 3.3 we introduce QGANs for the case where all components are quantum mechanical, hence enabling the generative modeling of quantum data. In this case, the discriminator aims at implementing the measurement for optimal distinguishability between target and generator states. In turn, the generator minimises the distinguishability[3].

We simulate this 'game' *in silico* by coupling two PQCs and optimising them in tandem. We analyse how the depths of generator and discriminator impact the generative performance and we design a heuristic for stopping the learning algorithm, which is a non-trivial problem even in classical adversarial methods.

3.2 The quantum circuit Born machine

3.2.1 Model definition and learning algorithm

In this Section, we describe the QABM model and the learning framework in general.

Let us consider a dataset $\mathcal{D} = \left\{\mathbf{x}^{(d)}\right\}_{d=1}^{D}$ which is a collection of D independent and identically distributed random vectors. The underlying probabilities are unknown and the objective is to learn a model for such distribution. For simplicity, we restrict our attention to N-dimensional binary vectors $\mathbf{x}^{(d)} \in \{-1, +1\}^N$, e.g., black-and-white images. This gives an intuitive one-to-one mapping between observation vectors and the computational basis of an N-qubit quantum system, that is $\mathbf{x} \leftrightarrow |\mathbf{x}\rangle = |x_1 x_2 \cdots x_N\rangle$. Note that standard binary encodings can be used to implement integer, categorical, and approximate continuous variables.

A quantum circuit model with fixed depth and gate layout, parameterized by a vector $\boldsymbol{\theta}$, prepares a wave function $|\psi(\boldsymbol{\theta})\rangle$ from which probabilities are obtained according to Born's rule $P_{\boldsymbol{\theta}}(\mathbf{x}) = |\langle \mathbf{x}|\psi(\boldsymbol{\theta})\rangle|^2$. Following a standard approach from generative machine learning [24], we can minimise the Kullback-Leibler (KL) divergence [137] $D_{KL}[P_{\mathcal{D}}|P_{\boldsymbol{\theta}}]$ from the circuit distribution in the computational basis $P_{\boldsymbol{\theta}}$ to the target distribution $P_{\mathcal{D}}$. Minimisation of this quantity is directly related to the minimisation of a well known cost function: the negative log-likelihood $\mathcal{C}(\boldsymbol{\theta}) = -\frac{1}{D}\sum_{d=1}^{D} \ln\Big(P_{\boldsymbol{\theta}}(\mathbf{x}^{(d)})\Big)$. However, there is a caveat; as probabilities are estimated from frequencies of a finite number of measurements, low-amplitude states

[3]The discrimination of quantum states was among the first problems ever considered in quantum information theory, more precisely, by Helstrom [136] in 1969. The novelty of the adversarial method is in using the discriminator's performance to provide a learning signal for the generator.

could lead to incorrect assignments. For example, an estimate $P_{\boldsymbol{\theta}}(\mathbf{x}^{(d)}) = 0$ for some $\mathbf{x}^{(d)}$ in the dataset would trigger the logarithm of zero and lead to infinite cost. To avoid singularities in the cost function, we use a simple variant

$$\mathcal{C}_{nll}(\boldsymbol{\theta}) = -\frac{1}{D}\sum_{d=1}^{D} \ln\Big(\max(\epsilon, P_{\boldsymbol{\theta}}(\mathbf{x}^{(d)}))\Big), \tag{3.4}$$

where $\epsilon > 0$ is a hyper-parameter to be set before the learning process begins[4]. Note that the number of measurements needed to obtain an unbiased estimate of the relevant probabilities may not scale favourably with the number of qubits N. For this reason, in Section 3.2.3.4 we explore alternative cost functions.

After estimating the cost, we update the parameter vector $\boldsymbol{\theta}$ to further minimise the cost. This can in principle be done by any suitable classical optimiser. Here we use a gradient-free algorithm called particle swarm optimisation (PSO) [139, 140] as previously done in the VQE [141]. The algorithm iterates for a fixed number of steps, or until a local minimum is reached and the cost does not decrease.

We choose the layout of the QCBM to be of the following form. Let us consider circuits parameterized by single-qubit rotations $\{\theta_i^{(l,k)}\}$ and two-qubit entangling rotations $\{\theta_{ij}^{(l)}\}$. The subscripts denote qubits involved in the operation, l denotes the layer number and $k \in \{1,2,3\}$ denotes the rotation identifier. The latter is needed as we decompose an arbitrary single-qubit rotation into three simpler rotations (in the next Section we discuss some potential simplifications depending on the set of gates available in hardware). Inspired by the gates readily available in trapped ion quantum computers, we use alternating layers of arbitrary single-qubit gates (odd layers) and Mølmer-Sørensen XX entangling gates [142, 143, 144, 145] (even layers) as our model. All parameters are initialised at random.

It is important to note that in our model the number of parameters is fixed and is independent of the size D of the dataset. This means we can obtain a good approximation to the target distribution only if the model is flexible enough to capture its complexity. Increasing the number of layers or changing the topology of the entangling layer alter this flexibility, potentially improving the quality of the ap-

[4]This hyper-parameter can be interpreted as the pseudocount for a Laplace smoothing, with the only difference being that we do not renormalise the probabilities. Laplace developed the smoothing technique when estimating the chance that the sun will rise tomorrow [138].

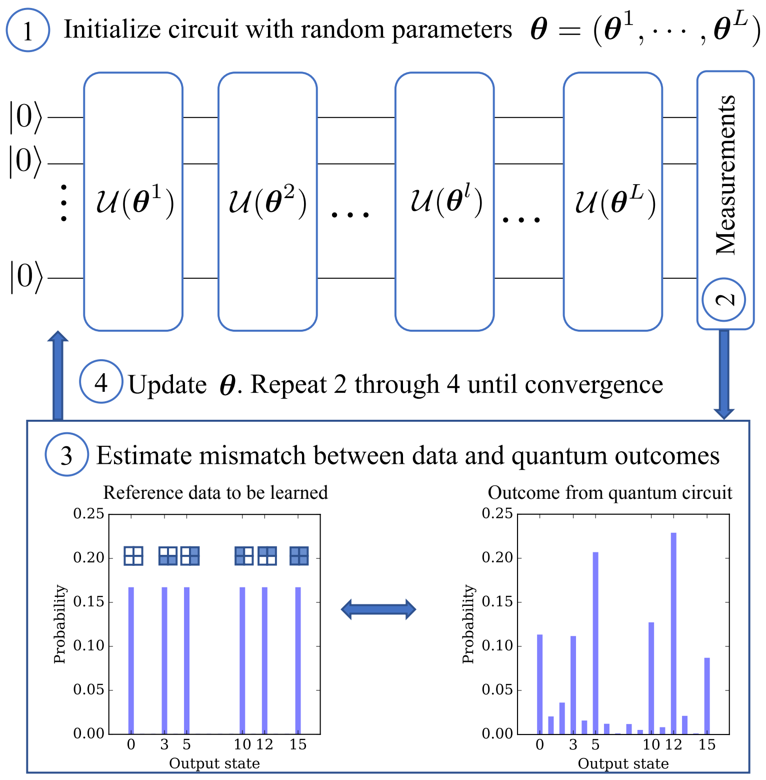

1 Initialize circuit with random parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}^1, \cdots, \boldsymbol{\theta}^L)$
$|0\rangle$
$|0\rangle$
$|0\rangle$
$\mathcal{U}(\boldsymbol{\theta}^1)$
$\mathcal{U}(\boldsymbol{\theta}^2)$
$\mathcal{U}(\boldsymbol{\theta}^l)$
$\mathcal{U}(\boldsymbol{\theta}^L)$
Measurements
2
4 Update $\boldsymbol{\theta}$. Repeat 2 through 4 until convergence
3 Estimate mismatch between data and quantum outcomes
Reference data to be learned
Outcome from quantum circuit
Probability
Output state
0.25
0.20
0.15
0.10
0.05
0.00
0 3 5 10 12 15

Figure 3.3: Data-driven quantum circuit learning. The training data is interpreted as representative of an unknown probability distribution. The generative task is to model such a distribution. The 2^N amplitudes of an N-qubit quantum state prepared by the computer are used to capture the correlations observed in the dataset. Learning consists of updating the parameters $\boldsymbol{\theta}$ of a quantum circuit Born machine (QCBM) to minimise the mismatch between the data and the measurement outcomes.

proximation. However, we anticipate that such flexible models are more challenging to optimise because of their larger number of parameters. Principled ways to choose the circuit layout and to regularise its parameters could significantly help in such a case.

As summarised in Figure 3.3, the learning algorithm iteratively adjusts all the parameters to minimise the value of the cost function. At any iteration the cost is approximated using both samples from the dataset and measurement outcomes from the quantum computer. We refer to this approach as the *data-driven quantum circuit learning* (DDQCL).

3.2.2 Implementation

In the trapped ion computer at University of Maryland we can perform arbitrary single-qubit rotations and Mølmer-Sørensen XX entangling gates involving any two qubits [146]. We used only these gates hence avoiding the need of further compilation. For the simulations *in silico* we implemented the same constraints dictated by the trapped ion experimental setting, but we assumed perfect gate fidelities and error-free measurements. This was implemented using the `QuTiP2` [147] Python library.

In the trapped ion setting, the use of R_z single-qubit rotations is very convenient because it is implemented as a change of frame of reference. In other words, these are 'virtual' gates that have no cost in terms of execution time and fidelity. Therefore, we allow for arbitrary single-qubit operations relying on the decomposition $U_i^{(l)} = R_z(\theta_i^{(l,3)}) R_x(\theta_i^{(l,2)}) R_z(\theta_i^{(l,1)})$, where l is the layer number, i is the qubit index, and $\theta_i^{(l,k)} \in [-\pi, +\pi]$ are Euler angles. The rotations are then expressed as exponentials of Pauli operators $R_z(\theta_i^{(l,\cdot)}) = \exp\Big(-\frac{i}{2}\theta_i^{(l,\cdot)}\sigma_i^z\Big)$ and $R_x(\theta_i^{(l,\cdot)}) = \exp\Big(-\frac{i}{2}\theta_i^{(l,\cdot)}\sigma_i^x\Big)$.

We execute circuits always starting from the $|0\cdots0\rangle$ state. Since the first set of R_z rotations has no effect, we do not include it. When an odd number of layers is used, a similar exception occurs in the last layer. There, the last set of R_z rotations would only add a phase that becomes irrelevant when taking the amplitude squared required for the Born machine. In other words, we can slightly reduce the number of parameters without changing the expressive power of the circuit. Every other layer of arbitrary single-qubit operations would in general require $3N$ parameters, where N is the number of qubits. By using an alternative decomposition, namely $U = R_x R_z R_x$, we could apply commutation rules with XX gates and obtain a reduction to $2N$ parameters in all odd layers. We decided *not* to do this because there is no effective reduction of the number of parameters for experiments up to $L = 5$ layers considered here.

For the entangling gates we use the notation $U_{ij}^{(l)} = XX(\theta_{ij}^{(l)})$, which in exponential form reads as $XX(\theta_{ij}^{(l)}) = \exp\Big(-\frac{i}{2}\theta_{ij}^{(l)}\sigma_i^x\sigma_j^x\Big)$. Recalling that states that differ by a global phase are indistinguishable, a direct computation shows that the adjustable parameters can be taken as $\theta_{ij}^{(l)} \in [-\pi, +\pi]$. Also, there is no need to choose an order for these gates within an entangling layer as they commute with one another.

The number of parameters per entangling layer depends on the chosen connectivity topology. The top right Panel of Figure 3.7 shows a graphical representation of these topologies for the case of $N = 4$ qubits: *all* is a complete graph with

$N(N-1)/2$ parameters, *chain* is a one-dimensional nearest neighbour graph with $N-1$ parameters, and *star* is a star-shaped graph with $N-1$ parameters.

Once the number of layers and topology of entangling gates is fixed, the quantum circuits described above provide a template. By adjusting its parameters we explore a (small) subset of the unitaries that are in principle allowed in the Hilbert space. The variational approach aims at finding the optimal parameters by minimising a cost function. Here we employ the cost function in Eq. (3.4) with $\epsilon = 10^{-8}$.

We use a global-best particle swarm optimisation algorithm [140] implemented in the `PySwarms` [148] Python library. The 'position' of a particle corresponds to a candidate solution, that is, a point $\boldsymbol{\theta}$ in parameter space defining a quantum circuit. The 'velocity' of a particle is a vector determining how to update the position. Both position and velocity are initialised at random and change at each iteration according to the swarm dynamics. There are three hyper-parameters controlling the swarm dynamics: a cognition coefficient c_1, a social coefficient c_2 and an inertia coefficient w. After testing a grid of values, we found that a constant value of 0.5 for all three hyper-parameters works well for our purpose. To avoid large jumps in parameter space, we further restricted position updates in each dimension to a maximum magnitude of π.

We used a number of particles which is twice the number of parameters to optimise. That is, more complex QCBMs are optimised by a larger number of particles. Finally to estimate the cost function for each particle, we always used the outcomes of 1000 measurements in the computational basis and 1000 data vectors sampled exactly from the target distribution.

3.2.3 Results

3.2.3.1 GHZ state preparation

To test the capabilities of DDQCL, we started with the preparation of Greenberger-Horne-Zeilinger (GHZ) states, also known as 'cat states' [149]. Besides their importance in quantum information, the choice is motivated by their simple description and by the availability of many studies about their preparation (see, e.g., Refs. [150, 37, 151]).

From the DDQCL perspective, we explored whether it is possible to learn any of the known recipes for GHZ state preparation starting only from classical data. The

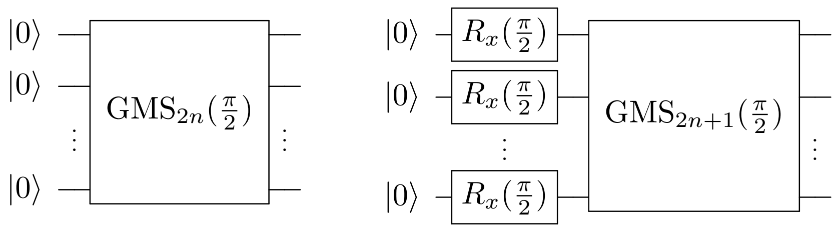


Figure 3.4: Data-driven preparation of GHZ-like states. The left (right) Panel shows a recipe obtained for even (odd) number of qubits. DDQCL brought all parameters very close to $\frac{\pi}{2}$ even though it was not constrained to do so. A human expert rounded the parameters to some precision discovering these circuits. R_x stands for the single-qubit rotation about the x axis. GMS stands for a global Mølmer-Sørensen gate acting on all the $N = 2n$ ($N = 2n+1$) qubits and is equivalent to the application of local XX gates to all $N(N-1)/2$ pairs of qubits.

target data consisted of samples from a distribution corresponding to the two desired computational basis states: $P_0 = 0.5$ for the state $|0\dots0\rangle$ and $P_1 = 0.5$ for the state $|1\dots1\rangle$. This distribution, which we call *the zero-temperature ferromagnet*, could be easily prepared as a mixed state. However, our study relies on pure states prepared by the QCBM; then, the only way to reproduce the zero-temperature ferromagnet is by implementing a unitary transformation that prepares a GHZ-like state. By GHZ-like state we mean a state that differs from the GHZ state only by a relative phase.

Using a layer of single-qubit rotations followed by an entangling layer with the all-to-all topology, DDQCL was indeed able to yield many degenerate preparations of GHZ-like states. In particular, we ran particle swarm optimisation on 25 random initialisations for 3, 4, 5 and 6-qubit instances. A human expert inspected the best set of parameters found for each size and, after rounding the parameters to some precision, spotted a clear pattern. Instances of 3 and 5 qubits yielded a recipe, while instances of 4 and 6 qubit yielded a different recipe. The recipes so obtained are summarised in Figure 3.4 and were verified for larger number of qubits, both even and odd. It turns out that DDQCL successfully reproduced a known recipe for trapped ion quantum computers [150] which, to the best of our knowledge, corresponds to the most compact and efficient GHZ state preparation using XX gates. Another commonly used recipe consists of cascading entangling gates and alternations of single-qubit rotations [151]. DDQCL produced approximate recipes of this kind in some tests when using an entangling layer with chain topology.

Note that in DDQCL all the parameters are learned independently and not constrained to be the same. Yet the learning algorithm unveiled that for this dataset all parameters need to converge to the same value, as shown in Figure 3.4. This is not necessarily the case for other datasets. We also note that the simulations assumed noiseless hardware, making the analysis of parameters easier for the human expert. It would be much more difficult to analyse parameters found with noisy hardware, as DDQCL can learn to compensate certain types of noise, e.g., systematic parameter offsets, in non-trivial ways. The upside is that learning can be successful even in the presence of such noise.

Finally, one can obtain more general solutions to this problem by allowing DDQCL to prepare mixed states. For example, consider a circuit acting on both the main qubit register and an additional ancilla register. By tracing out the ancilla register, e.g., ignoring it during measurement, the main register can implement a mixed state and can be trained to simulate a zero-temperature ferromagnet. Another example which does not resort to an ancilla register is to use decoherence as a mechanism to prepare mixed states that explain the data.

3.2.3.2 Coherent thermal states

Thermal states play an important role in statistical physics, quantum information, and machine learning. Using DDQCL, we trained QCBMs with a number of layers $L \in \{1, 2, 3\}$ and all-to-all topology to prepare thermal states starting from data. A thermal dataset in N dimensions is obtained by exact sampling realisations of $\boldsymbol{x} \in \{-1, +1\}^N$ from the Boltzmann distribution

$$P(\boldsymbol{x}) = Z^{-1} \exp\Big(T^{-1}\Big(\sum_{ij} J_{ij} x_i x_j + \sum_i h_i x_i\Big)\Big), \tag{3.5}$$

where Z is the normalisation constant, T is the temperature, and J_{ij} and h_i are random coefficients. We sampled these coefficients from a Gaussian distribution with zero mean and $\sqrt{N}$ standard deviation as in the Sherrington-Kirkpatrick model [87].

By decreasing the temperature T of the target distribution, we can increase the difficulty of the learning task. This will be evident later from the numerical results, but let us first informally justify the claim with examples. When the temperature is high, say infinite, the Boltzmann distribution corresponds to the uniform distribution over all possible binary vectors. This can be easily modeled by a single-layer QCBM (e.g., a layer of Hadamard gates). When the temperature is low, say zero, the distribution is uniform over the degenerate ground states. Depending on the degeneracy, the QCBM model may require large quantities of entanglement and therefore the learning of many layers. For the Sherrington-Kirkpatrick model we also know that in the limit of large system size a phase transition is expected at $T_{\mathrm{c}} \approx 1$. Although this is not true for small-sized systems, we take this value as a reference temperature and consider temperature $T \in \{2T_{\mathrm{c}}, T_{\mathrm{c}}, T_{\mathrm{c}}/1.5\}$.

For each setting of the temperature and circuit depth, we generated 25 synthetic instances, obtained the corresponding thermal datasets, and executed DDQCL. In order to plot a learning curve indicating the performance of the algorithm, we stored the KL divergence of the QCBM from the target at each iteration. Results on the 25 instances allowed us to use bootstrapping techniques and obtain error bars for the learning curve. In particular, from the 25 learning curves, we sampled $10,000$ sets of size 25 with replacement and computed the median learning curve for each set. From the distribution of $10,000$ medians, we computed the median and the error bars from the 5-th and 95-th percentiles as the lower and upper limits, respectively, accounting for a 90% confidence interval.

Figure 3.5 shows the bootstrapped median and the 90% confidence interval for the KL divergence on instances of $N = 5$ qubits. Deeper QCBMs, such as $L = 3$ (purple pentagons), consistently outperformed $L = 2$ (red circles) and $L = 1$ (yellow triangles). This became more evident as we went from the easy learning task with $T = 2T_{\mathrm{c}}$ in Panel (a) to the hard learning task with $T = T_{\mathrm{c}}/1.5$ in Panel (c).

To assess how well DDQCL performs on the generative task, we compared it to the inverse Bethe approximation [101] (see also Eqs. (3.21) and (3.22) in Ref. [152]). This is a classical closed-form approach widely used in statistical physics to infer the parameters of an Ising model when given a dataset of observations. As shown in Figure 3.5, the inverse Bethe approximation (green bar) performed extremely well in the easy task (a), matched DDQCL with $L = 3$ in the intermediate task (b), and underperformed on the difficult task (c). The latter observation comes from the fact that the median performance of the inverse Bethe approximation has very large confidence intervals. Results for instances of larger size were consistent; in Figure 3.6

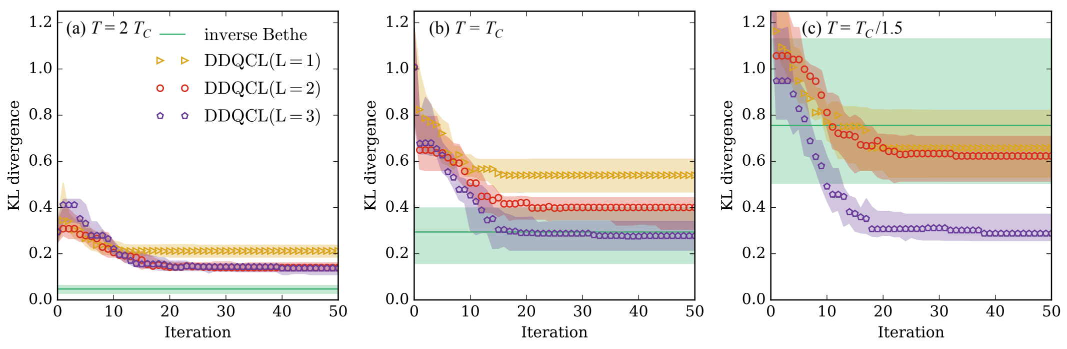


Figure 3.5: Data-driven preparation of coherent thermal states for $N = 5$ qubits. We report the bootstrapped median KL divergence and 90% confidence interval of 25 instances. (a) With $T > T_{\text{c}}$, the learning task is easy and low-depth QCBMs such as $L = 1$ (yellow triangles) and $L = 2$ (red circles) perform very well. (b) With $T \approx T_{\text{c}}$, a gap in performance between QCBMs of different depth becomes evident. (c) With $T < T_{\text{c}}$, the learning task is hard and the deeper QCBM $L = 3$ (purple pentagons) outperforms. The inverse Bethe approximation (green band) produces a classical model which is excellent for the easy task in (a), matches the best quantum model in (b), and underperforms on the hard task in (c).

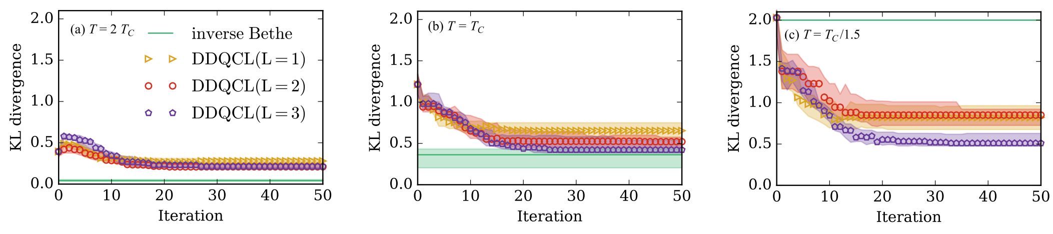


Figure 3.6: Data-driven preparation of coherent thermal states for $N = 6$ qubits. The results are consistent with those shown in Figure 3.5. For the low-temperature case in (c), the inverse Bethe approximation converged only in 7 out of 25 instances. No median value was extracted in this case and we reported a KL divergence of 2.0 as a reference.

we show the results for $N = 6$ qubits.

We emphasise that this result is not a form of quantum supremacy as the two methods are fundamentally different. DDQCL prepares a quantum state without the assumption of an underlying Boltzmann distribution, while the inverse Bethe approximation infers the parameters with such assumption. The error in the inverse Bethe approximation is expected to go to zero with system size, and only above the reference temperature T_c. Thus, it is not surprising that this classical method underperformed in Figures 3.5 (c) and 3.6 (c).

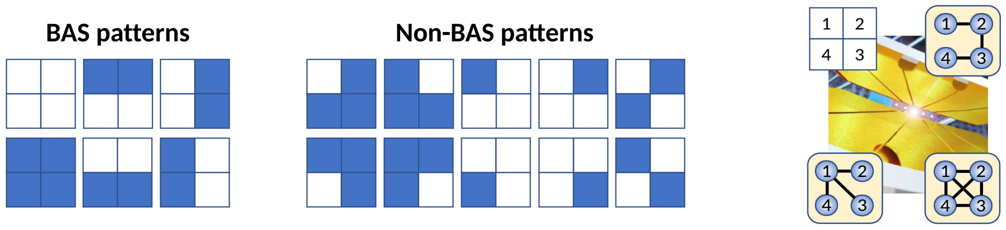


Figure 3.7: BAS dataset and mappings of pixels to qubits. The left Panel shows patterns that belong to BAS(2,2) while the central Panel shows undesired patterns. The right Panel shows a possible mapping of the 4 pixels to $N = 4$ qubits and some of the qubit-to-qubit connectivity topologies that can be natively implemented by trapped ion quantum computers: *chain*, *star*, and *all*.

3.2.3.3 The qBAS score and its application to benchmarking

Bars-and-Stripes (BAS) [99] is a synthetic dataset of images that has been widely used to study unsupervised generative models. BAS(n,m) consists of $n \times m$ pixel pictures obtained by setting each row (or column) to either black (-1) or white ($+1$), at random. Such images can be efficiently produced and visualised.

The total number of images is obtained as follows. First we count the number of single stripes, double stripes, etc., that can fit into the n rows. This number is the sum of binomial coefficients $\sum_{k=0}^{n}\binom{n}{k} = 2^n$. The same expression holds for the number of bars that can be placed in the m columns, that is 2^m. Note that empty (all-white) and full (all-black) patterns are counted in both the bars and the stripes. Therefore, we obtain the total count for the BAS patterns by subtracting the two extra patterns from this double count

$$N_{\mathrm{BAS}(n,m)} = 2^n + 2^m - 2. \tag{3.6}$$

The probability distribution so generated is $1/N_{\mathrm{BAS}(n,m)}$ for each pattern belonging to BAS(n,m), and zero for any other pattern. The top left Panel of Figure 3.7 shows patterns belonging to BAS$(2,2)$, while the top central Panel shows the remaining patterns.

Now, we would like to design a task-specific figure of merit to assess the performance of the components of the hybrid system. It shall take into account quantum resources such as the circuit depth, the gate fidelities, and any other architectural aspects, such as the qubit-to-qubit connectivity and the native set of single- and two-qubit gates. Moreover, it shall take into account classical resources such as the choice of cost function, the optimiser, and the hyper-parameters. As we show next, DDQCL can train a circuit to prepare a quantum state that encodes the BAS probability distribution; in turns, this can be used to compute a figure of merit which we call *the qBAS(n,m) score.*

The qBAS(n,m) score is an instantiation of the F_1 score widely used in the context of information retrieval. The F_1 score is defined as the harmonic mean of the precision p and the recall r, i.e., $\mathrm{F}_1 = 2pr/(p+r)$. The precision p indicates the ability to retrieve patterns which belong to the dataset [5]. In our context this is the number of measurement outcomes that belong to the BAS(n,m) dataset, divided by the total number of measurements N_{reads}. The recall r is the capacity of the model to retrieve the whole set patterns belonging to the dataset. In our case, if we denote the number of distinct measured patterns as $d(N_{\mathrm{reads}})$, then $r = d(N_{\mathrm{reads}})/N_{\mathrm{BAS}(n,m)}$. To score high ($\mathrm{F}_1 \approx 1.0$), both high precision ($p \approx 1.0$) and high recall ($r \approx 1.0$) are required.

The F_1 score is a useful measure for the quality of information retrieval and classification algorithms, but for our purposes it has a caveat: the dependence of r on the total number of measurements. As an example, consider a model that generates only BAS patterns, i.e., its precision is 1.0, but with highly heterogeneous distribution. If some of the BAS patterns have infinitesimally small probability, we can still push the recall to 1.0 by taking a large number of measurements $N_{\mathrm{reads}} \to \infty$. This is not desirable since our purpose is to evaluate circuits on the task of uniformly sampling all the patterns from BAS(n,m).

To define a unique score which is sensitive to deviations, let us first assume a model that perfectly matches the target distribution $P_{\mathrm{BAS}(n,m)} = 1/N_{\mathrm{BAS}(n,m)}$. Under this assumption the expected number of samples needed to obtain a value of $r = 1.0$ can be estimated using the famous *coupon collector's problem*. Then, we can set N_{reads} to be equal to the expected number of samples that need to be drawn to collect all the $N_{\mathrm{BAS}(n,m)}$ patterns ('coupons'). That is, $N_{\mathrm{reads}} = N_{\mathrm{BAS}(n,m)} H_{N_{\mathrm{BAS}(n,m)}}$, where H_k is the k-th harmonic number. In Table 3.1 we provide pre-computed values of N_{reads} for different values of n and m up to 100 qubits. Clearly, the number of readouts required to determine qBAS(n,m) are within experimental capabilities

[5]The meaning and usage of precision in the field of information retrieval differs from the definition of precision in other branches of science and statistics.

(n,m)	N_{qubits}	$N_{\text{BAS}(n,m)}$	N_{reads}
(2,2)	4	6	15
(2,3)	6	10	30
(3,3)	9	14	46
(4,4)	16	30	120
(7,7)	49	254	1554
(8,8)	64	510	3475
(10,10)	100	2046	16780

Table 3.1: Experimental requirements for the qBAS(n,m) score on quantum computers with up to 100 qubits. As described in the main text, N_{reads} is the number of readouts required for every estimation of the qBAS score.

of current NISQ devices.

For statistical robustness, we recommend as a good practice to perform R repetitions of the N_{reads} measurements leading to R independent estimates of the recall (each denoted as r_i). For the precision p, instead, all the samples should be used to robustly estimate this quantity. Using this value of p one can compute R independent values of the qBAS(n,m) score from each of the r_i. These are subsequently bootstrapped to obtain a more robust average for the final reported value of qBAS(n,m).

We note that a more general performance indicator is the KL divergence $D_{KL}[P_{\text{BAS}(n,m)}|P_{\boldsymbol{\theta}}]$. However, this would scale worse than the qBAS(n,m) score. As $n \times m$ becomes large, it is expected that the KL divergence is frequently undefined. This is true when measurements yield distributions such that $P_{\boldsymbol{\theta}}(\boldsymbol{x}^{(d)}) = 0$ for any of the $\boldsymbol{x}^{(d)}$ in BAS(n,m). In all these cases, the qBAS score can still be computed and the number of measurements N_{reads} remains relatively small to be practical for intermediate size $n \times m$[6].

But why do we expect the BAS distribution to be a good target for assessing the hybrid system? For the purposes of benchmarking and measuring the power of NISQ devices with DDQCL, it is insufficient to have a classically easy-to-generate target dataset; we also require such a dataset to be challenging to encode in a quantum state. Because of the importance of entanglement in quantum information processing, we considered the entanglement entropy averaged over all two-qubit

[6]The number of patterns in BAS(n,m) is dominated by $\max(2^n, 2^m)$. For a processor with q qubits we can choose $n = m = \lfloor\sqrt{q}\rfloor$. Approximating the harmonic number as $H_k = \ln k + \gamma$, where $\gamma \approx 0.577$ is the Euler-Mascheroni constant, we obtain $N_{\text{reads}} \sim \mathcal{O}(2^{\lfloor\sqrt{q}\rfloor})$. For a system of say $q = 400$ qubits we can estimate the score using a number of measurements of the order of a million.

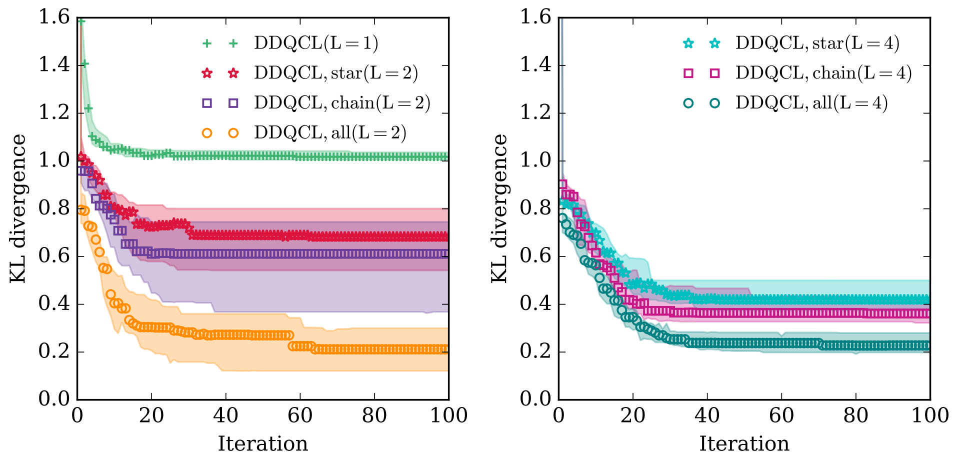


Figure 3.8: Data-driven preparation of BAS(2, 2). We report the bootstrapped median KL divergence and 90% confidence interval. The left Panel shows results for low-depth QCBMs with different topologies. The non-entangling QCBM $L = 1$ (green crosses) severely underperforms. A significant improvement is obtained with $L = 2$ where, however, the choice of topology becomes a key factor: chain (purple squares) performs slightly better than star (red stars) even though they have the same number of parameters; all-to-all (orange circles) has more expressive power and significantly outperforms all the others. The right Panel extends this analysis to deeper QCBMs with $L = 4$. All the topologies achieve a lower median KL divergence and a smaller confidence intervals.

subsets [153] as a proxy measure of a specific quantum state's usefulness for benchmarking purposes.

We start by noting that the four-qubit GHZ state, whose rich entangled nature makes it ideal for studying decoherence and decay of quantum information [150, 151], has entanglement entropy $S_{GHZ} = 1$. Now consider states that encode BAS(2, 2) in the computational basis. In Appendix 5.4 we show that the minimum value of entanglement entropy that any such state must have is $S_{BAS(2,2)} \approx 1.25163$. Furthermore, we show that the maximum value that a quantum representation of BAS(2, 2) can have is $S_{BAS(2,2)} \approx 1.79248$, which happens to be the maximum entanglement entropy known for any four-qubit state [153].

Let us now see the qBAS(2, 2) score in action. We decided to use it to compare the entangling topologies sketched in the top right Panel of Figure 3.7, and to compare circuits with different number of layers. The process consists of two steps; first, DDQCL is used to encode BAS(2, 2) in the wave function of a quantum state. Second, the best circuits, i.e., those achieving the lowest value for the cost function,

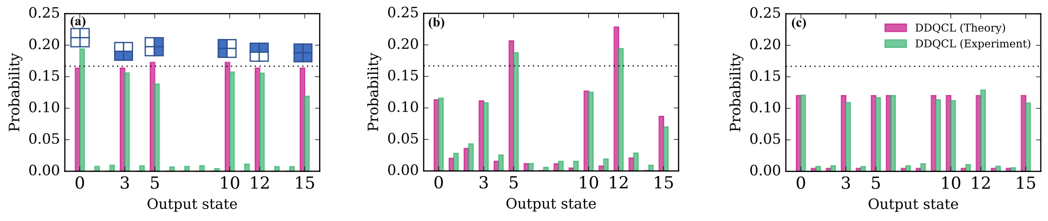


Figure 3.9: Comparison between simulated circuits and experimental implementations in the trapped ion quantum computer hosted by University of Maryland. These histograms are for the best circuits for BAS(2,2) obtained by DDQCL under three different setting: (a) all-to-all with $L = 2$ layers, (b) star with $L = 4$, and (c) star with $L = 2$. The theoretical state obtained in (a) is close to optimal and has a remarkable entanglement entropy averaged over all two-qubit subsets of $S_{BAS(2,2)} = 1.69989$. Circuit diagrams for (a-c) are shown in Appendix 5.5.

are compared using the qBAS(2,2) score.

Figure 3.8 shows the bootstrapped median of the KL divergence and 90% confidence interval over 25 random initialisations of DDQCL *in silico*. Note that the bootstrapping was performed as in Section 3.2.3.2. The all-to-all topology (orange circles) always outperforms sparse topologies (red stars and purple squares). However, deeper QCBMs do not always provide significant improvements, as it is the case for all-to-all $L = 4$ (dark green circles) versus all-to-all $L = 2$ (orange circles). A possible explanation is that, when going from two to four layers, we approximately double the number of parameters, and particle swarm optimisation struggles to find enhanced local optima. Another plausible explanation is that for this small dataset, the all-to-all QCBMs with $L = 2$ are already close to optimal performance (this can be seen from the histogram in Figure 3.9 (a)).

Figure 3.9 shows histograms from *in silico* simulations (pink bars) and from experiments in the trapped ion quantum computer (green bars). These histograms are for the best circuits obtained by DDQCL under three different setting: (a) all-to-all with $L = 2$ layers, (b) star with $L = 4$, and (c) star with $L = 2$. Circuit diagrams for (a-c) are shown in Appendix 5.5. Note that the theoretical state obtained in Panel (a) is close to optimal and attains a remarkable entanglement entropy of $S_{BAS(2,2)} = 1.69989$; this is further evidence that DDQCL is capable of handling quantum states that are rich in entanglement. Now, although visually the experiments seem to match the theory, it is difficult to quantify the quality of the solutions and benchmark the experimental results against the theoretical ones. That is where the qBAS score comes into play.

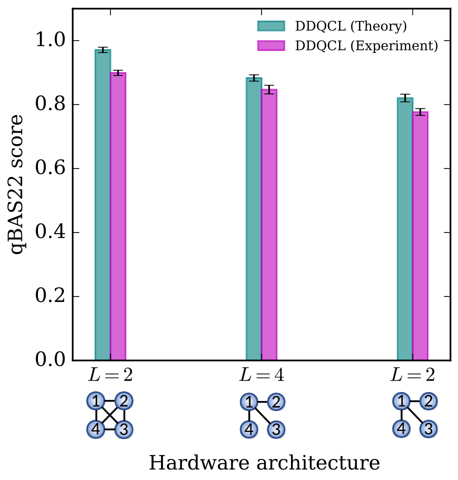


Figure 3.10: qBAS(2, 2) score for three different circuit settings. We report the bootstrapped mean and 95% confidence interval for simulations (green bars) and experiments on the trapped ion quantum computer hosted by University of Maryland (pink bars).

In order to calculate a robust qBAS score we used the following bootstrapping technique. The score was computed 25 times from batches of $N_{\text{reads}} = 15$ samples. From the 25 repetitions, we sampled 10,000 sets of size 25 with replacement and computed the mean for each. From the distribution of 10,000 means, we computed the mean and the error bars from two standard deviations, accounting for a 95% confidence interval. Figure 3.10 shows the bootstrapped mean qBAS(2, 2) score and 95% confidence interval for simulations (green bars) and experiments on the trapped ion quantum computer (pink bars). Clearly, the score is sensitive to the depth of the circuit as shown by the performance improvement of $L = 4$ compared to $L = 2$ in the star topology. Note that the theoretical improvement for using $L = 4$ is larger than that observed experimentally in the trapped ion quantum computer. This is because the quantum computer accumulated errors while executing the deeper circuit. The score is also sensitive to the choice of topology as shown by the drop in performance of star compared to all-to-all when the same number of layers $L = 2$ is used.

Although we compared circuits implemented on the same trapped ion hardware, the score may be used to compare different device generations or even completely different architectures (e.g., superconductor-based versus atomic-based). Similarly, one may use the score to compare classical resources of the hybrid system (e.g., different optimisers).

3.2.3.4 Comparison of cost functions

In realistic machine learning scenarios, we typically do not have access to the target distribution, nor to the output state of QCBM. Hence, we need to compare the two distributions at the level of histograms created from a finite number of samples and measurements. Here we compare three cost functions via simulations.

First, recall that the *clipped negative log-likelihood* is defined as

$$\mathcal{C}_{nll}(\boldsymbol{\theta}) = -\frac{1}{D}\sum_{d=1}^{D} \ln\Big(\max(\epsilon, P_{\boldsymbol{\theta}}(\mathbf{x}^{(d)}))\Big), \tag{3.7}$$

where probabilities are estimated from samples and $\epsilon > 0$ is a small number that avoids an infinite cost when $P_{\boldsymbol{\theta}}(\mathbf{x}^{(d)}) = 0$. We chose to use $\epsilon = 10^{-8}$.

Second, let us define the *earth mover's distance* [154] as

$$\mathcal{C}_{emd}(\boldsymbol{\theta}) = \min_{F} \langle d(\mathbf{x},\mathbf{y})\rangle_F, \tag{3.8}$$

where $F(\mathbf{x},\mathbf{y})$ is a joint probability distribution such that $\sum_{\mathbf{y}} F(\mathbf{x},\mathbf{y}) = P_{\mathcal{D}}(\mathbf{x})$ and $\sum_{\mathbf{x}} F(\mathbf{x},\mathbf{y}) = P_{\boldsymbol{\theta}}(\mathbf{y})$. That is, the marginals correspond to the data and QCBM distributions, respectively. Intuitively, this is the minimum cost of turning one histogram into the other where the metric $d(\mathbf{x},\mathbf{y})$ specifies the cost of transporting a single unit from $\mathbf{x}$ to $\mathbf{y}$. We chose $d(\mathbf{x},\mathbf{y})$ to be the Hamming distance between strings $\mathbf{x} \in \{-1,+1\}^N$ and $\mathbf{y} \in \{-1,+1\}^N$. Since we normalise histograms to sum up to one, the Earth Mover's Distance is equivalent to the 1-st Wasserstein distance [155]. In our simulations, we use the `PyEMD` Python library for fast computation of the earth moving distance [156].

Third, let us define the *moment matching* as

$$\mathcal{C}_{mm}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i}^{N} (\langle x_i\rangle_{P_{\mathcal{D}}} - \langle x_i\rangle_{P_{\boldsymbol{\theta}}})^2 + \frac{2}{N(N-1)}\sum_{i>j}^{N} (\langle x_i x_j\rangle_{P_{\mathcal{D}}} - \langle x_i x_j\rangle_{P_{\boldsymbol{\theta}}})^2, \tag{3.9}$$

where the expectation values $P_{\mathcal{D}}$ and $P_{\boldsymbol{\theta}}$ are taken with respect to data and QCBM distributions, respectively. This cost function can be generalised to include moments beyond the second as well as using different positive exponents for the error.

We compared the cost functions on the task of learning thermal states of size $N = 5$, with $L = 3$ layers and *all* topology. Figure 3.11 shows the bootstrapped

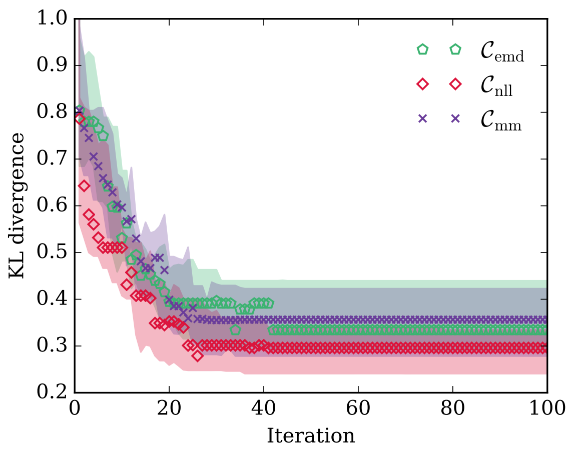


Figure 3.11: Comparison of cost functions. We report the bootstrapped median KL divergence and 90% confidence interval. Both moment matching ($\mathcal{C}_{mm}$) and earth mover's distance ($\mathcal{C}_{emd}$) closely track the clipped negative log-likelihood ($\mathcal{C}_{nll}$).

median KL divergence and 90% confidence interval on 25 realisations for 100 learning iterations. The fact that $\mathcal{C}_{nll}$ (red diamonds) outperforms other cost functions does not come as a surprise; minimisation of the negative log-likelihood is directly related to minimisation of the KL divergence. However, we expect the performance of DDQCL based on this cost function to degrade quickly as the size of the problem increases. In realistic applications, the relevant probabilities in Eq. (3.7), i.e., those associated with the data, are a vanishing fraction of the 2^N probabilities. Moreover, QCBM probabilities need to be estimated from a finite number of measurements.

The earth mover's distance $\mathcal{C}_{emd}$ (green pentagons) performs well, but it suffers from similar scalability issues. Fast algorithms for the computation of this distance may struggle when the number of bins in the histogram increases exponentially as in our case. However, it is reassuring to see that alternative cost functions with no relation to the KL divergence can still produce satisfactory results.

Surprisingly, the moment matching $\mathcal{C}_{mm}$ (purple crosses) closely tracks the other cost functions while retaining computational efficiency. In fact, even though a large number of samples may be needed to obtain low-variance estimates for the moments, only $\mathcal{O}(N^2)$ terms are estimated at each iteration. We expect this cost function to be a good heuristic for DDQCL on large systems.

3.2.4 Discussion

Data is an essential ingredient of any machine learning task. We presented a data-driven quantum circuit learning algorithm for generative modeling of classical data and for benchmarking of hybrid systems.

First, we learned a GHZ state preparation recipe for a trapped ion quantum computer. Minimal intervention of a human expert allowed to generalise the recipe to any number of qubits. This is not an example of compilation, but rather an illustration of how simple probability distributions can guide the synthesis of interesting quantum states. Depending on the circuit structure, the noise in the system, and other factors, the algorithm could lead to a different solution to the same generative tasks. The message here is that 'there is more than one way to skin a cat (state)'.

Second, we trained QCBMs to prepare approximations of thermal states. This illustrates the power of Born machines [128] to approximate Boltzmann machines [77] when the dataset requires thermal-like features.

Finally, tapping into the real power of near-term quantum devices we designed the qBAS score, a task-specific performance metric based on a canonical dataset called Bars-and-Stripes. This dataset is easy to prepare, visualise and validate classically. On the other hand, modeling Bars-and-Stripes requires significant quantum resources in the form of entanglement. Errors in the device and any other sub-optimal setting affect the qBAS score making it an appealing choice for (i) comparing devices across generations, (ii) comparing different architectures, (iii) selecting the best classical optimiser, and (iv) selecting the best hyper-parameters. All these components must be indeed optimised if we aim for a successful implementation of hybrid systems as the number of qubits increases. The qBAS score can be estimated in all NISQ computers available to date.

It is left to future work to demonstrate realistic machine learning using more powerful quantum circuit Born machines (QCBMs). At a finite and fixed low circuit depth, the power of the generative model can be enhanced by including ancilla qubits, in analogy to the role of hidden units in probabilistic graphical models. Layer-wise pre-training of the quantum circuit inspired by deep learning [73, 24] could initialise the larger number of parameters to near-optimal locations in the cost landscape. Finally, the method could be generalised to learn quantum data. In the next Section we import promising ideas from deep learning, namely adversarial methods, and take a step in this direction.

3.3 The quantum generative adversarial network

3.3.1 Model definition

In this Section, we start from information theoretic arguments and derive an adversarial algorithm that learns to generate approximations to quantum data.

Let us consider the problem of generating a pure state ρ_g close to an unknown pure target state ρ_t, where closeness is measured with respect to some distance metric to be chosen[7]. Hereby we use subscripts g and t to label 'generated' and 'target' states, respectively. The unknown target state is provided a finite number of times by a channel. If we were able to learn the state preparation procedure, then we could generate as many 'copies' as we want and use these in a subsequent application. We now describe a game between two players whose outcome is an approximate state preparation for the target state.

Borrowing language from the literature of adversarial machine learning, the two players are called the generator and the discriminator. The task of the generator is to prepare a quantum state and fool the other player into thinking that it is the true target state. Thus, the generator is a unitary transformation G applied to some known initial state, say $|0\rangle$, so that $\rho_g = G\,|0\rangle\langle 0|\,G^\dagger$. We will discuss the generator's strategy later.

The discriminator has the task of distinguishing between the target state and the generated state. It is presented with the mixture $\rho_{mix} = P(t)\rho_t + P(g)\rho_g$, where $P(t)$ and $P(g)$ are prior probabilities summing to one. Note that in practice the discriminator sees one input at a time rather than the mixture of density matrices, but we can treat the uncertainty in the input state using this picture. The discriminator performs a positive operator-valued measurement (POVM) $\{E_b\}$ on the input, so that $\sum_b E_b = I$. According to Born's rule, measurement outcome b is observed with probability $P(b) = \mathrm{tr}[E_b \rho_{mix}]$. The outcome is then fed to a decision rule, a function that estimates which of the two states was provided in input.

A straightforward application of Bayes' theorem suggests that the decision

[7]Our derivations are valid for mixed states in general, hence the use of density operators. However, to simplify the simulations and the discussion we focus here on the specific case of pure states, density operators of rank 1.

rule should select the label for which the posterior probability is maximal, i.e., $\arg\max_{x\in\{g,t\}} P(x|b)$. This rule is called the Bayes' decision function and is optimal in the sense that, given an optimal POVM, any other decision function has a larger probability of error [157]. Recalling that $\max_{x\in\{g,t\}} P(x|b)$ is the probability of the correct decision using the Bayes decision function, we can formulate the probability of error as

$$\begin{aligned} P_{err}(\{E_b\}) &= \sum_b P(b)(1 - \max_{x\in\{g,t\}} P(x|b)) \\ &= \sum_b P(b) \min_{x\in\{g,t\}} P(x|b) \\ &= \sum_b \min_{x\in\{g,t\}} P(x|b)P(b) \\ &= \sum_b \min_{x\in\{g,t\}} P(b|x)P(x) \\ &= \sum_b \min_{x\in\{g,t\}} \mathrm{tr}[E_b \rho_x] P(x). \end{aligned} \tag{3.10}$$

We observe that the choice of POVM plays a key role here, hence, the discriminator should try to find the best possible one. We can write the objective function for the discriminator in variational form as

$$P^*_{err} = \min_{\{E_b\}} P_{err}(\{E_b\}), \tag{3.11}$$

where the minimisation is over all possible POVM elements, and the number of POVM elements is unconstrained.

It was Helstrom who carefully designed a POVM achieving the smallest probability of error when a single sample of ρ_{mix} is provided [136]. He showed that the optimal discriminator comprises two elements, E_0 and E_1, which are diagonal in a basis that diagonalises $\Gamma = P(t)\rho_t - P(g)\rho_g$. When the outcome 0 is observed, the state is labeled as 'target', when the outcome 1 is observed the state is labeled as 'generated'. This would be the discriminator's optimal strategy as it minimises the probability of error in Eq. (3.11). Unfortunately, designing such a measurement would require knowledge of the target state beforehand, contradicting the purpose of the game. Yet we now know that the optimal POVM comprises only two elements.

Using this information, and plugging Eq. (3.10) in Eq. (3.11), we obtain [157]

$$\begin{aligned} P^*_{err} &= \min_{\{E_0,E_1\}} \Big(P(1|t)P(t) + P(0|g)P(g) \Big) \\ &= \min_{\{E_0,E_1\}} \Big(\mathrm{tr}[E_1\rho_t]P(t) + \mathrm{tr}[E_0\rho_g]P(g) \Big) \\ &= \min_{E_0} \Big(\mathrm{tr}[(I-E_0)\rho_t]P(t) + \mathrm{tr}[E_0\rho_g]P(g) \Big) \\ &= \min_{E_0} \Big(-\mathrm{tr}[E_0\rho_t]P(t) + \mathrm{tr}[E_0\rho_g]P(g) \Big) + P(t), \end{aligned} \tag{3.12}$$

where we used $E_1 = I - E_0$ from the definition of POVM.

We now return to the generator and outline its strategy. Assuming the discriminator be optimal, the generator achieves success by maximising the probability of error P^*_{err} with respect to the generated state ρ_g. The result is a zero-sum game similar to that of generative adversarial networks [116] and described by

$$\begin{aligned} &\max_{\rho_g} \min_{E_0} \Big(-\mathrm{tr}[E_0\rho_t]P(t) + \mathrm{tr}[E_0\rho_g]P(g) \Big) \\ =&\min_{\rho_g} \max_{E_0} \Big(\mathrm{tr}[E_0\rho_t]P(t) - \mathrm{tr}[E_0\rho_g]P(g) \Big), \end{aligned} \tag{3.13}$$

where we dropped the constant terms. Now suppose that the game is carried out in turns. On the one hand, the discriminator is after an unknown Helstrom measurement which changes over time as the generator plays. On the other hand, the generator tries to approximate an unknown target state exploiting the learning signal provided by the discriminator.

Note that when $P(t) = P(g) = \frac{1}{2}$, the probability of error in Eq. (3.11) is related to the trace distance between quantum states [44]

$$\begin{aligned} D(\rho_t, \rho_g) &= \frac{1}{2} \|\rho_t - \rho_g\| \\ &= \max_{\{E_b\}} \frac{1}{2} \sum_b |\,\mathrm{tr}[E_b(\rho_t - \rho_g)]|. \end{aligned} \tag{3.14}$$

This is clearer from the variational definition in the second line. Hence, by playing the minimax game above with equal prior probabilities, we are implicitly minimising the trace distance between target and generated state.

Ideally, we would use the trace distance to analyse the learning progress and to design a stopping criterion for the algorithm. For pure states, the trace distance can be estimated using the well-known SWAP test. In practice, implementing a coherent SWAP test on a NISQ computer may be highly non-trivial. We discuss this in Appendix 5.7 where we require a fault-tolerant quantum computer. Furthermore, recall that our derivation is valid also for mixed states; in this case, there is no known simple way to estimate the trace distance and one shall resort to some bound. Since our main interest is in NISQ implementations, and since we need to analyse the learning progress and design a stopping criterion, we put forward a heuristic argument in Section 3.3.2.3.

3.3.2 Implementation

3.3.2.1 Near-term implementation on NISQ computers

We now discuss how the game could be played in practice using noisy quantum computers and no error correction.

First, we assume the ability to efficiently provide the unknown target state as an input. For example, the target state could come from an external channel and be loaded in the quantum computer's register with no significant overhead – the source could be the output of another quantum computer, while the channel could be a quantum Internet [33].

Second, the generator's unitary transformation shall be implemented by a parameterized quantum circuit applied to a known initial state. Note that target and generated states have the same number of qubits and they are never input together, but rather as a mixture with probabilities $P(t)$ and $P(g)$, respectively, i.e., randomly selected with a certain prior probability. Hence they can be prepared in the same quantum register.

Third, resorting to Neumark's dilation theorem [158], the discriminator's POVM shall be realised as a unitary transformation followed by a projective measurement on an extended system. Such extended system consists of the quantum register shared by the target and generated states, plus an ancilla register initialised to a known state. Notice that the number of basis states for the ancillary system needs to match the number of POVM elements. Because here we specifically require two POVM elements, the ancillary system consists of just one ancilla qubit. The unitary transformation on this extended system is also implemented by a parameterized quantum circuit. The measurement is described by projectors on the state space of the ancilla and the two possible outcomes, 0 and 1, are respectively associated with labels 'target' and 'generated'.

Depending on the type of gates, depth, and qubit-to-qubit connectivity, we explore (small) regions of the Hilbert space with the generator circuit, and (small) regions of the cone of positive operators with the discriminator circuit.

Let us see a more concrete example. Assume that the unknown n-qubit target state $\rho_t = |\psi_t\rangle\langle\psi_t|$ can be prepared in the main register $\mathcal{M}$. We construct a generator circuit $G = G_L \cdots G_1$ where each gate is either fixed, e.g., a Controlled NOT, or parameterized. Parameterized gates are often of the form $G_l(\theta_l) = \exp(-i\theta_l H_l/2)$ where θ_l is a real-valued parameters and $H_l \in \{X,Y,Z,I\}^{\otimes n}$ is a tensor product of n Pauli matrices. The generator acts on the initial state $|0\rangle^{\otimes n}$ and can prepare $\rho_g = G|0\rangle\langle 0|G^\dagger$ in the main register $\mathcal{M}$.

Similarly, we construct a discriminator circuit $D = D_K \cdots D_1$ acting non-trivially on both main register $\mathcal{M}$ and ancilla qubit $\mathcal{A}$. Each gates is either fixed or parameterized as $D_k(\phi_k) = \exp(-i\phi_k H_k/2)$, where ϕ_k is real-valued and H_k is a tensor product of $n+1$ Pauli matrices. We measure the ancilla qubit using projectors $E_b = I^{\otimes n} \otimes |b\rangle\langle b|$ with $b \in \{0,1\}$.

Collecting parameters for generator and discriminator into vectors $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, respectively, the minimax game in Eq. (3.13) can be written as $\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}} V(\boldsymbol{\theta},\boldsymbol{\phi})$ with value function

$$\begin{aligned} V(\boldsymbol{\theta},\boldsymbol{\phi}) = & P(t)\,\mathrm{tr}\Big[E_0 D\Big(|\psi_t\rangle\langle\psi_t| \otimes |0\rangle\langle 0|\Big)D^\dagger\Big] - \\ & P(g)\,\mathrm{tr}\Big[E_0 D\Big(G|0\rangle\langle 0|G^\dagger \otimes |0\rangle\langle 0|\Big)D^\dagger\Big]. \end{aligned} \tag{3.15}$$

Each player optimises the value function in turn. This optimisation can in principle be done via different approaches (e.g., gradient-free, first-, second-order methods, etc.) depending on the computational resources available. Here we discuss a simple method of alternated optimisation by gradient descent/ascent

$$\begin{aligned} \boldsymbol{\theta}^{(t+1)} &= \arg\min_{\boldsymbol{\theta}} V(\boldsymbol{\theta},\boldsymbol{\phi}^{(t)}) \\ \boldsymbol{\phi}^{(t+1)} &= \arg\max_{\boldsymbol{\phi}} V(\boldsymbol{\theta}^{(t+1)},\boldsymbol{\phi}), \end{aligned} \tag{3.16}$$

starting from randomly initialised parameters $\boldsymbol{\theta}^{(0)}$ and $\boldsymbol{\phi}^{(0)}$.

To start with, we need to compute the partial derivatives of the value function. The favourable properties of the tensor products of Pauli matrices appearing in our gate definitions allow us to compute the analytical gradient [159, 160, 161] (see Appendix 5.6 for discussion about circuit learning). For the generator, the partial derivatives read

$$\begin{aligned}\frac{\partial V}{\partial \theta_l} = -\frac{P(g)}{2}\Big\{ &\operatorname{tr}\Big[E_0 D\Big(G_{l+}\left|0\right\rangle\!\left\langle 0\right| G_{l+}^{\dagger} \otimes \left|0\right\rangle\!\left\langle 0\right|\Big) D^{\dagger}\Big] - \\ &\operatorname{tr}\Big[E_0 D\Big(G_{l-}\left|0\right\rangle\!\left\langle 0\right| G_{l-}^{\dagger} \otimes \left|0\right\rangle\!\left\langle 0\right|\Big) D^{\dagger}\Big]\Big\},\end{aligned} \tag{3.17}$$

where

$$G_{l\pm} = G_L \cdots G_{l+1} G_l(\theta_l \pm \tfrac{\pi}{2}) G_{l-1} \cdots G_1. \tag{3.18}$$

Note that $G_{l\pm}$ can be interpreted as two new circuits, each one differing from G by an offset of $\pm\frac{\pi}{2}$ to parameter θ_l. Hence, for each parameter l, we are required to execute the circuit compositions DG_{l+} and DG_{l-} on initial state $\left|0\right\rangle^{\otimes n+1}$ and measure the ancilla qubit. Because these auxiliary circuits have the same depth as the original circuit, estimation of the gradient is efficient. Interestingly, up to a scale factor of $\frac{\pi}{2}$, the analytical gradient is equal to the central finite difference approximation carried out at π (see Eq. (5.10) in Appendix 5.6).

Similarly, the partial derivatives for the discriminator read

$$\begin{aligned}\frac{\partial V}{\partial \phi_k} = &\frac{P(t)}{2}\Big\{ \operatorname{tr}\Big[E_0 D_{k+}\Big(\left|\psi_t\right\rangle\!\left\langle \psi_t\right| \otimes \left|0\right\rangle\!\left\langle 0\right|\Big) D_{k+}^{\dagger}\Big] - \\ &\qquad \operatorname{tr}\Big[E_0 D_{k-}\Big(\left|\psi_t\right\rangle\!\left\langle \psi_t\right| \otimes \left|0\right\rangle\!\left\langle 0\right|\Big) D_{k-}^{\dagger}\Big]\Big\} - \\ &\frac{P(g)}{2}\Big\{ \operatorname{tr}\Big[E_0 D_{k+}\Big(G\left|0\right\rangle\!\left\langle 0\right| G^{\dagger} \otimes \left|0\right\rangle\!\left\langle 0\right|\Big) D_{k+}^{\dagger}\Big] - \\ &\qquad \operatorname{tr}\Big[E_0 D_{k-}\Big(G\left|0\right\rangle\!\left\langle 0\right| G^{\dagger} \otimes \left|0\right\rangle\!\left\langle 0\right|\Big) D_{k-}^{\dagger}\Big]\Big\},\end{aligned} \tag{3.19}$$

where

$$D_{k\pm} = D_K \cdots D_{k+1} D_k(\phi_k \pm \tfrac{\pi}{2}) D_{k-1} \cdots D_1. \tag{3.20}$$

In this case, for each parameter k we are required to execute four auxiliary circuit compositions: D_{k+} and D_{k-} on target state $\left|\psi_t\right\rangle \otimes \left|0\right\rangle$, while $D_{k+}G$ and $D_{k-}G$ on initial state $|0\rangle^{\otimes n+1}$.

Finally, all parameters are updated by gradient descent/ascent

$$\begin{aligned} \theta_l^{(t+1)} &\longleftarrow \theta_l^{(t)} - \epsilon \frac{\partial V}{\partial \theta_l}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}, \boldsymbol{\phi}=\boldsymbol{\phi}^{(t)}} \\ \phi_k^{(t+1)} &\longleftarrow \phi_k^{(t)} + \eta \frac{\partial V}{\partial \phi_k}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t+1)}, \boldsymbol{\phi}=\boldsymbol{\phi}^{(t)}}, \end{aligned} \tag{3.21}$$

where ϵ and η are hyper-parameters determining the step sizes. Here we rely on the fine-tuning of these, as opposed to Newton's method which makes use of the Hessian matrix to determine step sizes for all parameters. Other researchers [162] designed circuits to estimate the analytical gradient and the Hessian matrix. Such approach requires the ability to execute complex controlled operations and is expected to require fault-tolerant computers. Our approach is in line with others [160, 163] requiring much simpler circuits which are suitable for NISQ computers.

As we discuss next, accelerated gradient techniques developed by the deep learning community can further improve our method.

3.3.2.2 Optimisation by resilient backpropagation

The problem of minimising the trace distance in Eq. (3.14) directly over the set of density matrices is convex [44]. In our approach, however, we deal with a potentially non-convex problem due to the optimisation of exponentiated parameters and hence the introduction of sine and cosine functions. In other words, we minimise the trace distance 'indirectly'.

A recent paper [164] suggested that the error surface of circuit learning problems is challenging for gradient-based methods due to the existence of barren plateaus. In particular, the region where the gradient is close to zero does not correspond to local minima of interest, but rather to an exponentially large plateau of states that have exponentially small deviations in error from that of the totally mixed state. The derivation of the above result is for a specific class of random circuits; in practice, however, we prefer to deal with highly structured circuits [165, 166].

Here we argue that the existence of plateaus does not necessarily pose a problem for the learning of quantum circuits, provided that the sign of the gradient can be resolved. To validate this claim we refer to the classical literature and argue that similar problems have traditionally occurred also in neural networks and allow for efficient solutions.

The standard gradient-based updates are of the form

$$w_i^{(t+1)} \longleftarrow w_i^{(t)} - \epsilon \frac{\partial}{\partial w_i} E^{(t)}, \tag{3.22}$$

where $w_i^{(t)}$ is the i-th parameter at time t, ϵ is the step size, $E^{(t)}$ is the error function to be minimized and its superscript indicates evaluation at $w = w^{(t)}$. If the step size is too small, the derivatives are also scaled to be too small resulting in a long time to convergence. If the step size is too large, it can lead to oscillatory behaviour of the updates or even to divergence. One of the early approaches to counter this behaviour was the introduction of a 'momentum' term, which takes into account the previous steps when calculating the current update. The learning rules for gradient-descent with momentum (GDM) read

$$\begin{aligned} \Delta_i^{(t)} &\longleftarrow -\epsilon \frac{\partial}{\partial w_i} E^{(t)} + \mu \Delta_i^{(t-1)} \\ w_i^{(t+1)} &\longleftarrow w_i^{(t)} + \Delta_i^{(t)}, \end{aligned} \tag{3.23}$$

where μ is a momentum hyper-parameter. Momentum produces some resilience to plateaus in the error surface, but can lose this resilience when the plateaus are characterised by having very small or zero gradient.

A family of optimisers known as resilient backpropagation algorithms (Rprop) [167] is particularly well suited for problems where the error surface is characterised by large plateaus with small gradient. Rprop algorithms adapt the step size for each parameter based on the agreement between the sign of its current and previous partial derivatives. If the signs of the two derivatives agree, then the step size for that parameter is increased multiplicatively. This allows the optimiser to traverse large areas of small gradient with an increasing speed. If the signs disagree, it means that the last update for that parameter was large enough to jump over a local minima. To fix this, the parameter is reverted to its previous value and the step size is decreased multiplicatively. Rprop is therefore resilient to gradients with very small magnitude as long as the sign of the partial derivatives can be determined.

We use a variant known as iRprop$^-$ [168] which does not revert a parameter to its previous values when the signs of the partial derivatives disagree. Instead, it sets the current partial derivative to zero so that the parameter is not updated, but its step size is still reduced. The hyper-parameters and pseudocode for iRprop$^-$ are described in Algorithm 2.

Algorithm 2 iRprop$^-$ [168]

```
Require: error function $E$, initial parameters $w_i^{(0)}$, initial step size $\Delta_{init}$, minimum
    allowed step size $\Delta_{min}$, maximum allowed step size $\Delta_{max}$, step size decrease
    factor $\eta^-$, and step size increase factor $\eta^+$
Ensure: $\Delta_i^{(-1)} := \Delta_{init}$ and $\frac{\partial}{\partial w_i} E^{(-1)} := 0$ for all $i$
 1: repeat
 2:     for each $i$ do
 3:         if $\frac{\partial}{\partial w_i} E^{(t-1)} \frac{\partial}{\partial w_i} E^{(t)} > 0$ then
 4:             $\Delta_i^{(t)} := \min\{\eta^+ \Delta_i^{(t-1)},\ \Delta_{max}\}$
 5:         else if $\frac{\partial}{\partial w_i} E^{(t-1)} \frac{\partial}{\partial w_i} E^{(t)} < 0$ then
 6:             $\Delta_i^{(t)} := \max\{\eta^- \Delta_i^{(t-1)},\ \Delta_{min}\}$
 7:             $\frac{\partial}{\partial w_i} E^{(t)} := 0$
 8:         else
 9:             $\Delta_i^{(t)} := \Delta_i^{(t-1)}$
10:         end if
11:         ${w_i}^{(t+1)} := {w_i}^{(t)} - \operatorname{sign}\left(\frac{\partial}{\partial w_i} E^{(t)}\right) \Delta_i^{(t)}$
12:     end for
13: until convergence
```

Despite the resilience of Rprop, if the magnitude of the gradient in a given direction is so small that the sign cannot be determined, then the algorithm will not take a step in that direction. Furthermore, the noise coming from the finite number of samples could cause the sign to flip at each iteration. This would quickly make the step size very small and the optimiser could get stuck on a barren plateau.

One possible modification is an ‘explorative’ version of Rprop that explores areas with zero or very small gradient at the beginning of training, but still converges at the end of training. First, any zero or small gradient at the very beginning of training could be replaced by a positive gradient to ensure an initial direction is always defined. Second, one could use large step sizes and decrease them during training to allow for convergence to a minima. Finally, the explorative Rprop could remember the sign of the last suitably large gradient and take a step in that direction whenever the current gradient is zero. This way, when the optimiser encounters a plateau, it would traverse the plateau from the same direction it entered. We leave the investigation of an explorative Rprop algorithm to future work.

3.3.2.3 Heuristic for the stopping criterion

Evaluating the performance of generative models is often intractable and can be done only via application-dependent heuristics [169, 170]. This is also the case for our model as the value function in Eq. (3.15) does not provide information about the generator's performance, unless the discriminator is optimal. Unfortunately, we do not always have access to an optimal discriminator (more on this in Appendix 5.7). We now describe an efficient method that can be used to assess the learning in the quantum setting. In turn, this can be used to define a stopping criterion for the algorithm.

We begin recalling that the discriminator makes use of projective measurements on an ancilla register $\mathcal{A}$ to effectively implement a POVM. Should the ancilla register be maximally entangled with the main register $\mathcal{M}$, its reduced density matrix would correspond to that of a maximally mixed state $\frac{1}{2}I$. Performing projective measurements on the maximally mixed state would then result in uniform random outcomes and decisions.

Ideally, the discriminator would encode all relevant information in the ancilla register and then remove all the correlations with the main register, obtaining a product state $\rho_d = \rho_d^{\mathcal{M}} \otimes \rho_d^{\mathcal{A}}$. Hereby we use subscript d to indicate the state output by the discriminator circuit. This scenario is similar in spirit to the uncomputation technique used in many quantum algorithms [171].

The bipartite entanglement entropy (BEE) is a measure that can be used to quantify how much entanglement there is between two partitions

$$S(\rho_d^{\mathcal{A}}) = -\operatorname{tr}\left[\rho_d^{\mathcal{A}} \ln \rho_d^{\mathcal{A}}\right] = -\operatorname{tr}\left[\rho_d^{\mathcal{M}} \ln \rho_d^{\mathcal{M}}\right] = S(\rho_d^{\mathcal{M}}), \tag{3.24}$$

where $\rho_d^{\mathcal{A}} = \operatorname{tr}_{\mathcal{M}}[\rho_d]$ and $\rho_d^{\mathcal{M}} = \operatorname{tr}_{\mathcal{A}}[\rho_d]$ are reduced density matrices obtained by tracing out one of the partitions, i.e., by ignoring one of the registers. Computing the BEE is intractable in general, but here we can exploit its symmetry and estimate it on the smallest partition, i.e., the ancilla register $\mathcal{A}$. Because this register consists of a single qubit, BEE reduces to

$$S(\rho_d^{\mathcal{A}}) = -\frac{1+\|\boldsymbol{r}\|}{2}\ln\left(\frac{1+\|\boldsymbol{r}\|}{2}\right) - \frac{1-\|\boldsymbol{r}\|}{2}\ln\left(\frac{1-\|\boldsymbol{r}\|}{2}\right), \tag{3.25}$$

where $\boldsymbol{r} \in \mathbb{R}^3$ is the vector of coordinates on the Bloch sphere such that $\rho_d^{\mathcal{A}} = \frac{1}{2}(I + \boldsymbol{\sigma}\cdot\boldsymbol{r})$, $\|\boldsymbol{r}\| \leq 1$, and $\boldsymbol{\sigma} = (\sigma_x, \sigma_y, \sigma_z)$. The three components of the Bloch vector can be estimated using tomography techniques for a single qubit, for which we refer to the excellent review in Ref. [172].

There exist a wide range of methods that can be used depending on the desired accuracy, the prior knowledge, and the available computational resources. In this work we consider the scaled direct inversion (SDI) [172] method, where each entry of the Bloch vector is estimated independently by measuring the corresponding Pauli operator. This is motivated by the fact that $\langle\sigma_i\rangle = \mathrm{tr}\left[\sigma_i\rho_d^{\mathcal{A}}\right] = \boldsymbol{e}_i\boldsymbol{r}$ where $\boldsymbol{e}_i$ is the Cartesian unit vector in the i direction and $i \in \{x, y, z\}$. These measurements can be done in all existing gate-based quantum computers we are aware of by applying a suitable rotation followed by a measurement in the computational basis.

We can write a temporary Bloch vector $\widehat{\boldsymbol{r}}_0 = \left(\widehat{\langle\sigma_x\rangle}, \widehat{\langle\sigma_y\rangle}, \widehat{\langle\sigma_z\rangle}\right)$ where all expectations are estimated from samples. Due to finite sampling error, there is non-zero probability that the vector lies outside the unite sphere, although inside the unit cube. These cases correspond to non-physical states and SDI corrects them by finding the valid state with minimum distance over all Schatten p-distances. As it turns out, this is simply the rescaled vector [172]

$$\widehat{\boldsymbol{r}} = \begin{cases} \widehat{\boldsymbol{r}}_0 & \text{if} \quad \|\widehat{\boldsymbol{r}}_0\| \leq 1 \\ \widehat{\boldsymbol{r}}_0/\|\widehat{\boldsymbol{r}}_0\| & \text{otherwise.} \end{cases} \tag{3.26}$$

This procedure allows us to efficiently estimate the BEE in Eq. (3.25). Equipped with this information, we can now design an heuristic for the stopping criterion.

The reasoning is as follows. Provided that the discriminator circuit is non-trivial, random initialisation of its parameters will likely generate entanglement between main and ancilla registers. In other words, $S(\rho_d^A)$ is expected to be large at the beginning. As the learning algorithm iterates, the discriminator gets more accurate at distinguishing states – this requires the ancilla qubit to depart from the totally mixed state and $S(\rho_d^A)$ to decrease. This is when the learning signal for the generator is stronger, allowing the generated state to get closer to the target. As the two become less and less distinguishable with enough iterations, the discriminator needs to increase correlations between ancilla's bases and relevant factors in the main register. That is, we expect to observe an increase of entanglement between the two registers, hence an increase in $S(\rho_d^A)$. The performance of the discriminator would then saturate as $S(\rho_d^A)$ converges to its upper bound of $\ln(2)$. We propose to detect this convergence and use it as a stopping criterion.

3.3.3 Results

In this Section, we show that the adversarial method can be used to approximate entangled target states.

In realistic scenarios, the target state would come from an external channel or be prepared in the quantum computer's register with no significant overhead. For the simulations we mock this scenario using circuits to prepare the target states. That is, we have $\rho_t = T\left|0\right\rangle\!\left\langle 0\right|T^\dagger$ where T is an unknown circuit. We setup a generator circuit G and a discriminator circuit D, and the composition of these circuits is shown in Figure 3.12, left Panel. We shall stress that neither the generator nor the discriminator are allowed to 'see' the inner workings of T at any time.

Figure 3.12, right Panel, shows the layer that we used as a building block for our circuits. It has $m-1$ general two-qubit gates where m is the number of qubits. Note that each general two-qubit gate can be efficiently implemented with three CNOT gates and 15 parameterized single-qubit rotations as shown in Ref. [173].

We are interested in studying the performance of the algorithm as we change the complexity of the circuits. The complexity of our circuits is determined by the number of layers – we denote this number as $c(\cdot)$ so that, for example, a generator circuit G made of 2 layers has complexity $c(G) = 2$.

All parameters were initialised uniformly at random in $[-\pi, +\pi]$. We chose $P(t) = P(g) = \frac{1}{2}$ so that the discriminator is given target and generated states with equal probability. All expected values required to compute gradients were estimated

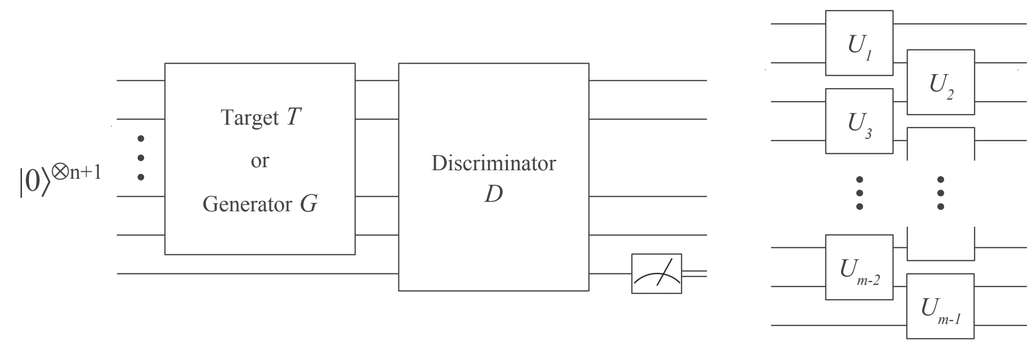


Figure 3.12: The left Panel illustrate a quantum generative adversarial network (QGAN). The target state is prepared by an unknown circuit T, which the generator circuit G learns to approximate. The discriminator circuit D takes n-qubit states in input and learns to label them as ‘target’ or ‘generated’ via the binary outcome of a projective measurement on an ancilla qubit. Neither G or D are allowed to ‘see’ the inner workings of T at any time. The learning signal for the generator comes solely from the probability of error of the discriminator. Right Panel: layer used as a building block for all the circuits in our simulations. For an m-qubit circuit the layer consists of a ladder of $m-1$ general two-qubit gates.

from 100 measurements on the ancilla qubit. Unless stated otherwise, optimisation was performed using iRprop^{-}. We used an initial step size $\Delta_{init} = 1.5\pi \times 10^{-3}$, a minimum allowed step size $\Delta_{min} = \pi \times 10^{-6}$, and a maximum allowed step size $\Delta_{max} = 6\pi \times 10^{-3}$.

Figure 3.13 shows learning curves for simulations on four qubits. The green downward triangles represent mean and one standard deviation of the trace distance between target and generated state, computed on 10 repetitions. In the left Panel, the number of layers are $c(T) = c(G) = 2$ and $c(D) = 1$. We observe that the complexity of the discriminator is not sufficient to provide a learning signal for the generator, and the final approximation is indeed not satisfactory. In the central Panel, $c(T) = c(D) = 2$ and $c(G) = 1$. The generator is less complex than the target state, but it manages to produce a meaningful approximation in average. In the right Panel, $c(T) = c(G) = c(D) = 2$. The complexity of all circuits is optimal, and the generator learns an indistinguishable approximation of the target state.

The trace distance reported here could have been approximately computed using the SWAP test (see Appendix 5.7). However, since we assumed a near-term implementation, we cannot reliably execute the SWAP test. In Section 3.3.2.3 we designed an efficient heuristic to keep track of learning. To test the idea, we per-

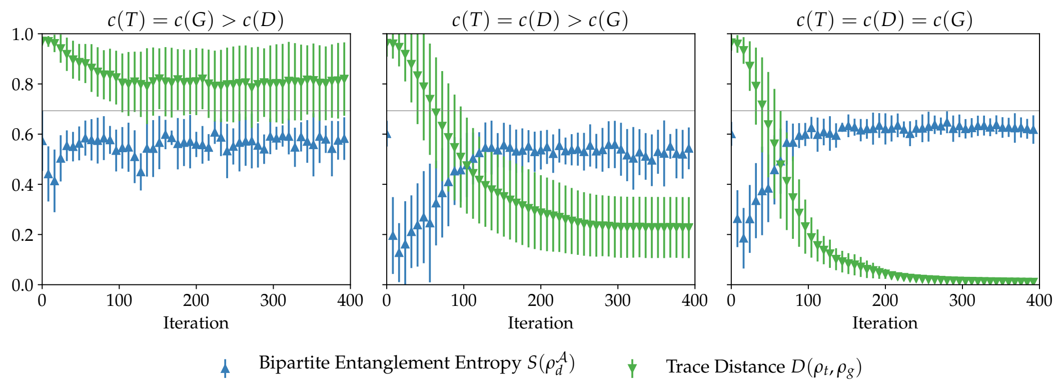


Figure 3.13: Learning curves for simulations on four-qubit target states. We report mean and one standard deviation computed on 10 repetitions. Titles relate the complexities of target $c(T)$, generator $c(G)$, and discriminator $c(D)$ circuits. Green downward triangles indicate the trace distance between target T and generator G, with zero indicating an optimal approximation. In the left Panel, the discriminator D is too simple to provide a learning signal for G. In the central Panel, G is simple but can produce a meaningful approximation of T. In the right Panel, all circuits are complex enough and by iteration 200 we learn an indistinguishable approximation to the target state. The trace distance cannot be always computed. The bipartite entanglement entropy (BEE) of the ancilla qubit (blue upward triangles) can be used as an efficient proxy to assess the learning progress. As learning progresses, the ancilla qubit gets closer to the mixed state where $S(\rho_d^A) = \ln(2) \approx 0.69$ (grey horizontal line). The convergence of BEE can be used as a stopping criterion for the algorithm.

formed additional 100 measurements on the ancilla qubit for each observable σ_x, σ_y, and σ_z. The outcomes were used to estimate the BEE using the SDI method. In Figure 3.13 the blue upwards triangles represent mean and one standard deviation of the BEE, computed on 10 repetitions. The left Panel shows that when the depth of the discriminator circuit is too low, BEE oscillates with no clear pattern. The central and right Panels show that, when using a favourable setting, the initial BEE drops significantly towards zero. This is when the generator begins to learn the target state. Note that, as the algorithm iterates, the ancilla qubit tends towards the maximally mixed state where $S(\rho_d^A) = \ln(2) \approx 0.69$ (grey horizontal line). In this regime, the discriminator predicts the labels with probability equal to the prior $P(t) = P(g) = \frac{1}{2}$.

The convergence of BEE can be used as a stopping criterion for the algorithm. The central and right Panels in Figure 3.13 show that BEE converged after approximately 150 iterations. Stopping the simulation at that point we obtained excellent results in average. We now show some simulated tomographic reconstructions for

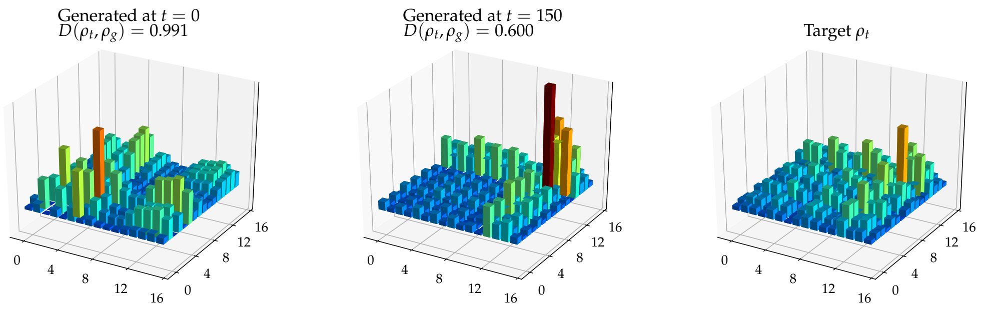


Figure 3.14: Absolute value of tomographic reconstructions for a four-qubit target state. Here we have a target state prepared by a random circuit of $c(T) = 2$ layers, a generator with $c(G) = 1$ and a discriminator with $c(D) = 2$. The right Panel shows the absolute value of the entries of the target density matrix. The initial generated state shown in the left Panel is at trace distance 0.991 from the target. Following our heuristic, we stopped the algorithm at iteration 150 where BEE converged. The final state, shown in the central Panel, is at trace distance 0.6 from the target. The generator managed to capture the main mode of the density matrix, that is, the sharp peak coloured in red.

two cases to support the claim.

First, let us examine the case where the generator is under-parameterized. Figure 3.14, right Panel, shows the absolute value of the entries of the density matrix for a four-qubit target state. The randomly initialised generator produced the state shown in the left Panel which is at 0.991 trace distance from the target. By stopping the algorithm at iteration 150, we generated the state shown in the central Panel whose trace distance is 0.6. The generator managed to capture the main mode of the density matrix, that is, the sharp peak visible on the right.

Second, let us examine the case where the generator is sufficiently parameterized. Figure 3.15, right Panel, shows the target state. The generator initially produced the state shown in the left Panel which is at trace distance 0.951 from the target. By stopping the adversarial algorithm at iteration 150, we generated the state shown in the central Panel whose trace distance is 0.121. Visually, the target and final states are indistinguishable.

But how do the complexities of generator and discriminator affect the outcome? To verify this, we run the adversarial learning on six-qubit target states of $c(T) = 3$ layers, and varied $c(G)$ and $c(D)$. After 600 training iterations, we computed the mean trace distance across five repetitions. As illustrated by the heat-map in Figure 3.16, increasing the complexity always resulted in a better approximation to the target state.

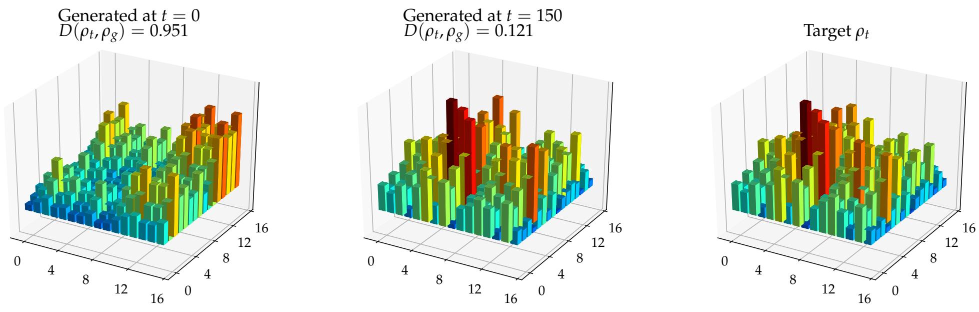
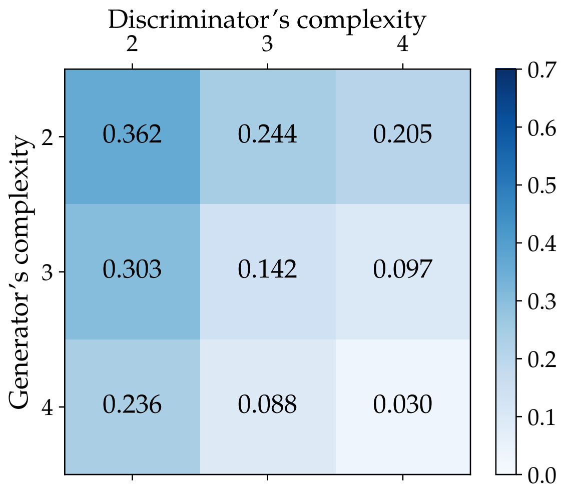


Figure 3.15: Absolute values of tomographic reconstructions for a four-qubit target state. The setting is similar to that of Figure 3.14, but this time the generator is a circuit with $c(G) = 2$ layers. The randomly initialised generator produces the state shown in the left Panel, which is at trace distance 0.951 from the target. Following our heuristic, we stopped the algorithm at iteration 150 where BEE converged. The final state, shown in the central Panel, is at trace distance 0.121 from the target. Visually, the target and final states are indistinguishable.



Generator's complexity \ Discriminator's complexity	2	3	4
2	0.362	0.244	0.205
3	0.303	0.142	0.097
4	0.236	0.088	0.030

Figure 3.16: Quality of the approximation against complexity of circuits for simulations on six-qubit target states. The heat-map shows mean trace distance of five repetitions of the algorithm, computed at iteration 600. All standard deviations were < 0.1 (not shown). The targets were produced by random circuits of $c(T) = 3$ layers. Increasing the complexity of discriminator $c(D) \in \{2, 3, 4\}$ and generator $c(G) \in \{2, 3, 4\}$ resulted in better approximations to the target state in all cases.

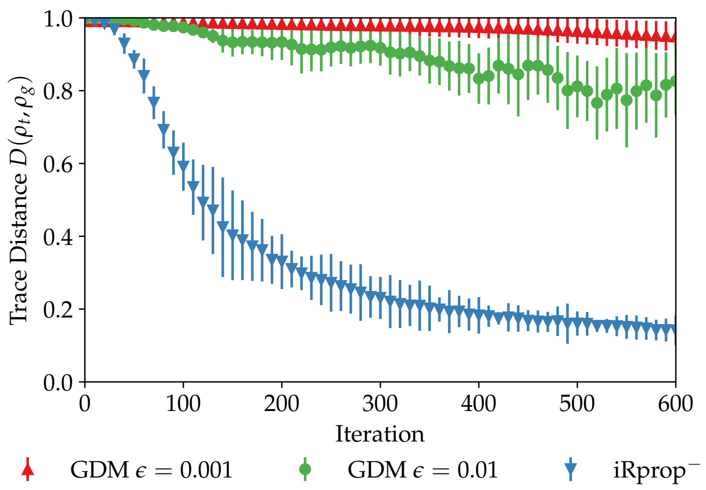


Figure 3.17: Learning curves for different optimisers in simulations on six-qubit target states. The lines represent mean and one standard deviation of the trace distance computed on five repetitions. All circuits had the same number of layers, $c(T) = c(G) = c(D) = 3$. iRprop$^-$ resulted in better performance than gradient descent with momentum (GDM) when using two different step sizes. Increasing the step size further in GDM resulted in unstable performance (not shown).

In our final test, we compared optimisation algorithms on six-qubit target states. We ran GDM and iRprop$^-$ for 600 iterations. Figure 3.17 shows mean and one standard deviation across five repetitions. iRprop$^-$ (blue downward triangles) outperformed GDM both with step size $\epsilon = 0.01$ (green circles) and $\epsilon = 0.001$ (red upward triangles). This is because despite the small magnitude of the gradients when considering targets of six qubits, we were able to estimate their sign and take relevant steps in the correct direction. This is a significant advantage of resilient backpropagation algorithms.

3.3.4 Discussion

In this work we proposed a quantum generative adversarial network (QGAN) that can approximately generate and discriminate pure quantum states. We used information theoretic arguments to formalise the problem as a minimax game. The discriminator circuit maximises the value function in order to better distinguish between the target and generated states. This can be thought of as learning to perform the Helstrom measurement [136]. In turn, the generator circuit minimises the value function in order to deceive the discriminator. This can be thought of as minimising the trace distance of the generated state to the target state. The desired outcome of this game is to obtain the best approximation to the target state for a given generator circuit.

Our near-term implementation has the advantage that it requires few qubits and avoids the SWAP test. A possible long-term implementation discussed in Appendix 5.7 can make use of the actual Helstrom measurement when the target is pure, potentially speeding up the learning process.

Previous work on quantum circuit learning raised the concern of barren plateaus in the error surface [164]. We showed numerically that a class of optimisers called resilient backpropagation [167] achieves high performance for the problem at hand, while gradient descent with momentum performs relatively poorly. These resilient optimisers require only the temporal behaviour of the sign of the gradient, and not the magnitude, to perform an update step. In our simulations of up to seven qubits we were able to correctly ascertain the sign of the gradient frequently enough for the optimiser to converge to a good solution. For regions of the error surface where the sign of the gradient cannot be reliably determined, we suggested an alternative optimisation method that could traverse such regions. We will explore this idea in future work.

In general it is not clear how to assess the model quality in generative adversarial networks, nor how to find a stopping criterion for the algorithm. For example, in the classical setting of computer vision, it is often the case that artificial samples are visually evaluated by humans, e.g., the Turing test, or by a proxy neural network, e.g., the Inception Score [170]. The quantum setting does not allow for these approaches in a straightforward manner. We therefore designed an efficient heuristic based on an estimate of the entanglement entropy of a single qubit, and numerically showed that convergence of this quantity indicates saturation of the performance. We proposed to use this as a stopping criterion.

We tested the quality of the approximation as a function of the complexity of the circuits. Our results indicate that investing more resources in the generator and discriminator circuits leads to noticeable improvements. An interesting avenue for future work is the study of different circuit layouts and parameter initialisations. If prior information about the target state is available, or can be efficiently extracted, one could encode it in the generator circuit by using a suitable ansatz. For example, in Ref. [163] the authors use the Chow-Liu tree to place CNOT gates such that they capture most of the mutual information among variables. Similarly, structured layouts could be used for the discriminator circuit such as hierarchical [165] and universal topologies [166]. These choices could reduce the number of parameters to learn and simplify the error surface.

Other machine learning proposals for state approximation require quantum resources that go far beyond those currently available. For example, the quantum principal component analysis [14] requires universal fault-tolerant hardware in order to implement the necessary SWAP operations. As another example, the gate-based quantum Boltzmann machine [34, 35] requires the preparation of highly non-trivial thermal states. In comparison, our method works well for approximating pure target states and can find application in quantum state tomography on NISQ computers. Clearly, a thorough numerical benchmark is needed to compare the scalability of these methods.

We conclude with an important remark. In this work, we relied on the variational definition of Bayesian probability of error, which assumes the availability of a single copy of the quantum state to discriminate. By assuming the availability of multiple copies, which is in practice the case, one can derive more general adversarial methods based on complex information theoretical quantities. These could be variational definitions of the quantum Chernoff bound [174], the Umegaki relative information, and other measures of distinguishability [157].

3.4 Subsequent developments

Several work has been released after the publication of the material presented in this Chapter. This indicates the importance of the models proposed here and, in general, is an indicator of how fast the field is growing.

In the context of the QCBM model presented in Section 3.2, Zhu *et al.* [175] implemented the whole learning process on four qubits of a trapped ion computer and experimentally demonstrated convergence of the model to the target distribution.

Liu and Wang [163] proposed the use of an alternative cost function that can be optimised using gradient-descent. Such a cost function is the maximum mean discrepancy [176] $D(p, q_{\boldsymbol{\theta}}) = \|\sum_{\boldsymbol{v}} p(\boldsymbol{v})\phi(\boldsymbol{v}) - \sum_{\boldsymbol{v}} q_{\boldsymbol{\theta}}(\boldsymbol{v})\phi(\boldsymbol{v})\|^2$, where ϕ is a function to be evaluated classically, and where expectations are estimated from samples.

Interestingly, their approach allows for gradient-based learning even for discrete data $\boldsymbol{v} \in \{0,1\}^n$, which is often not possible in classical generative models. In their simulations they successfully train QCBMs for the Bars-and-Stripes dataset and for discretised Gaussian distributions. Hamilton *et al.* [177] implemented this schema on the IBM Q20 Tokyo computer, and examined how statistical and hardware noise affect convergence. They found that the generative performance of state-of-the-art hardware is usually significantly worse than that of the numerical simulations.

Leyton-Ortega *et al.* [178] performed a complementary experimental study on the Rigetti 16Q-Aspen computer. They argued that due to the many components involved in hybrid systems (e.g., choice for the entangling layers, optimisers, post-processing strategy, etc.), their performance ultimately depends on the ability to correctly set hyper-parameters. Research on automated hyper-parameter setting will therefore be key to the success of QCBMs.

From the theoretical point of view, Du *et al.* [40] showed that QCBMs have strictly more expressive power than classical models such as deep Boltzmann machines, when only a polynomial number of parameters are allowed. Coyle *et al.* [41] showed that some QCBMs cannot be efficiently simulated by classical means in the worst case, and that this holds for all the circuit families encountered during training.

One key aspect of generative models is their ability to perform inference. This is the ability to 'clamp' some variables to known values and estimate values for other variables by sampling from the conditional probability. For example, inpainting, the process of reconstructing lost portions of images and videos, can be done by inferring missing values from a suitable generative model. Low *et al.* [179] used Grover's algorithm to perform inference on quantum circuits and obtain a quadratic speedup over naïve methods, although the overall complexity remained exponential. Zeng *et al.* [180] equipped the QCBM with this method, although this required amplitude amplification and estimation methods that may be beyond NISQ hardware capabilities. It is an open question how to perform inference on QCBMs in the near term.

In the context of the QGAN model presented in Section 3.3, some authors explored classical-quantum combinations for target, generator and discriminator.

Both Situ *et al.* [181] and Zeng *et al.* [180] used a quantum circuit generator and a classical neural network discriminator. They successfully reproduced the statistics of some discrete target distributions. Romero and Aspuru-Guzik [182] extended this approach to continuous target distributions using a suitable post-processing function.

Zoufal *et al.* [183] proposed a QGAN that approximately encodes 2^n-dimensional data vectors into the wave function of n qubits. While the best known generic method has exponential complexity, their QGAN uses a polynomial number of gates. If both the cost of training and the required precision are kept low, this method has the potential to facilitate algorithms that use this kind of encoding.

Hu *et al.* [184] experimentally demonstrated a QGAN on a single custom superconducting qubit.

In summary, this novel field has not been restricted to theory and simulation – a series of experimental demonstrations on scaled-down problems have been performed. In Appendix 5.3 we summarise the relevant demonstrations. The Reader interested in experimental setups is invited to delve into the references therein. In parallel, the software side has been moving at a fast pace (see Fingerhuth *et al.* [185] for a review of general quantum computing software). There now exist several platforms for hybrid algorithms which are specifically dedicated to ML. This enables experimentation at a much higher rate than previously possible, a scenario reminiscent of the deep learning developments a decade ago.

Chapter 4

Outlook

In this Thesis, we studied four generative models, namely QABM, QAHM, QCBM and QGAN. Towards the end of each respective Section, we analysed the pros and cons of these models and suggested research directions for future work. In this Chapter, we take a step back and give a brief outlook of the field.

Existing implementations of quantum machine learning, either annealing- or gate-based, are not able to compete with state-of-the-art classical algorithms. For example, no demonstration to date was able to challenge the performance of convolutional networks in computer vision. We believe the lack of results on large-scale real problems is one of the reasons why the machine learning community has not shown much interest in quantum computing to date. Physicists, on the other hand, have been increasingly relying on machine learning for models and calculations.

We believe that the hybrid quantum-classical framework is an opportunity to bring physicists and machine learning scientists together, as significant contributions are pressingly needed from both sides. More precisely, hybrid systems provide a framework for the incremental development of hardware and algorithms. In the near term, algorithms shall rely heavily on classical resources. As quantum hardware improves, those classical resources shall gradually be replaced by quantum resources in order to deliver the longed-for quantum *advantage*.

In the meantime, researchers have been progressing with the theory of quantum *supremacy*. For example, it has been shown that sampling from the probability distribution implemented by some classes of non-universal random circuits is a classically intractable task in the average case. As we have pointed out in this Thesis, generative modeling is also an intractable task that requires sampling from complex probability distributions. Is there an interesting link between quantum supremacy and generative modeling that has been overlooked? If so, could generative modeling be a natural application for quantum computers once quantum supremacy has been established?

We do not know. Indeed, it is important to distinguish between supremacy and advantage. Quantum supremacy is expected to be the next important milestone in the history of quantum computing, but such a demonstration may come with no clear practical utility. Quantum advantage is more ambitious as it requires outperforming classical computers on a real-world problem.

Optimisation is a ubiquitous task and is an obvious choice for the demonstration of advantage. Recent work has begun to explore the question of whether existing supremacy proposals lend themselves to improved optimisation algorithms. An affirmative answer would imply practical utility of near-term quantum computers if some key hardware requirements can also be met.

Sampling is another prevalent tasks and is an ideal candidate for such a demonstration. If quantum computers were shown to outperform classical computers on sampling, the advantage would impact leading-edge domains such as Bayesian inference, deep learning, and probabilistic programming. In turn, we would expect a strong impact across science and engineering.

An even more natural task for the demonstration is the *simulation* of quantum mechanical systems, for example those of interest to chemists. For quantum systems that exhibit complex behaviour, a classical model cannot learn to reproduce the statistics unless it uses exponentially scaling resources. Quantum models will deliver a clear advantage for this tasks, provided that we can efficiently handle quantum datasets and learn these models.

A careful analysis of the three domains within the context of existing hardware could significantly bring forward the demonstration of quantum advantage. In this Thesis, we took the sampling route and we experimented with two very different architectures: quantum annealers, which we treated as open quantum systems, and trapped ions, which we treated as closed quantum systems. There exist other interesting architectures, for example photonic computers based on continuous observables. In general there is no consensus on what architecture will be able to deliver the quantum advantage and scale well. Ultimately, we shall consider combining the strengths of each architecture towards solving large-scale real problem.

Chapter 5

Appendices

5.1 Approximating continuous stochastic variables in quantum annealers

Quantum information does not have to be encoded into binary observables, it could also be encoded into continuous observables [186]. There has been work on quantum machine learning that follows the latter direction [187, 188]. However, most available quantum computers do work with qubits nicely resembling the world of classical computation. Here we show how naïve approaches to encoding continuous variables in quantum annealers are likely to fail.

Consider the task of approximating a simple univariate Gaussian distribution. If we were able to do that, we could control the mean μ and the 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 stochastic binary variables. For example, the energy function could include a term $x = \sum_i w_i z_i$, where w_i is a programmable weight in the annealer and z_i is the eigenvalue of the $\hat{Z}_i$ Pauli operator for qubit i. Binary expansions commonly used in classical computers can be though of as a special case where weights increase or decrease exponentially with the precision (i.e., number of qubits used for the encoding). Such an expansion 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. The more general weighted-sum encoding above 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. (2.17) with zero transverse field

$$\begin{aligned} E(\boldsymbol{s}) &= \frac{1}{2\sigma^2}\Big(\sum_i w_i z_i - \mu\Big)^2 \\ &= \frac{1}{2\sigma^2}\Big(\sum_{i\neq j} w_i w_j z_i z_j + \sum_i w_i^2 + \mu^2 - 2\mu \sum_i w_i z_i\Big) \\ &= \sum_{i\neq j} J_{ij} z_i z_j + \sum_i h_i z_i + C \end{aligned} \tag{5.1}$$

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 naïve 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 direct implementation of stochastic continuous variables may be challenging in more general setups that go beyond the univariate Gaussian case.

5.2 Derivation of the bound for Gibbs distributions

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

$$\rho = \sum_i \frac{e^{-E_i}}{\mathcal{Z}} |i\rangle\langle i|, \tag{5.2}$$

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

$$\begin{aligned}
\ln\langle \boldsymbol{u}|\rho|\boldsymbol{u}\rangle &= \ln\langle \boldsymbol{u}|\sum_i \frac{e^{-E_i}}{\mathcal{Z}} |i\rangle\langle i|\boldsymbol{u}\rangle \\
&= \ln\sum_i |\langle i|\boldsymbol{u}\rangle|^2 \frac{e^{-E_i}}{\mathcal{Z}} \\
&\geq \sum_i |\langle i|\boldsymbol{u}\rangle|^2 \ln\frac{e^{-E_i}}{\mathcal{Z}} \\
&= \langle \boldsymbol{u}|\sum_i \ln\frac{e^{-E_i}}{\mathcal{Z}} |i\rangle\langle i|\boldsymbol{u}\rangle \\
&= \langle \boldsymbol{u}|\ln\rho|\boldsymbol{u}\rangle,
\end{aligned} \tag{5.3}$$

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

5.3 Experimental demonstrations to date

In Table 5.1 we provide an overview of parameterized quantum circuits for machine learning that have been demonstrated experimentally on actual quantum hardware.

The tested models were: perceptron, probably approximately correct (PAC), oracle, quantum autoencoder (QAE), quantum approximate optimisation algorithm (QAOA), quantum circuit Born machine (QCBM), quantum kernel estimator (QKE), quantum generative adversarial network (QGAN), and variational quantum model (VQM).

The tested learning algorithms were gradient-based and gradient-free. N/A indicates that a learning algorithm was either not required or not used. For example, in some cases the learning process was simulated classically and the learned model was then deployed on quantum hardware.

Finally, the tested hardware architectures were: superconducting (S), trapped ion (T), and photonic (P).

Reference	**Task**	**Model**	**Learning algorithm**	**Qubits**	**Computer**
Schuld *et al.* [189]	Classification	QKE	N/A	4	IBM Q5 Yorktown (S)
Grant *et al.* [165]	Classification	VQM	N/A	4	IBM Q5 Tenerife (S)
Havlíček *et al.* [190]	Classification	QKE, VQM	Gradient-based	2	IBM Q5 Yorktown (S)
Tacchino *et al.* [191]	Classification	Perceptron	Gradient-based	3	IBM Q5 Tenerife (S)
Benedetti *et al.* [192]	Generative	QCBM	N/A	4	Custom (T)
Hamilton *et al.* [177]	Generative	QCBM	Gradient-based	4	IBM Q20 Tokyo (S)
Zhu *et al.* [175]	Generative	QCBM	Gradient-free	4	Custom (T)
Leyton-Ortega *et al.* [178]	Generative	QCBM	Gradient-based, gradient-free	4	Rigetti 16Q-Aspen (S)
Coyle *et al.* [41]	Generative	QCBM	Gradient-based	4	Rigetti 16Q-Aspen (S)
Hu *et al.* [184]	State learning	QGAN	Gradient-based	1	Custom (S)
Zoufal *et al.* [183]	State learning	QGAN	Gradient-based	3	IBM Q20 Poughkeepsie (S)
Rocchetto *et al.* [37]	State learning	PAC	N/A	6	Custom (P)
Otterbach *et al.* [123]	Clustering	QAOA	Gradient-free	19	Rigetti 19Q-Acorn (S)
Ding *et al.* [193]	Compression	QAE	Gradient-free	3	Rigetti 8Q-Agave (S)
Ristè *et al.* [194]	Parity with noise	Oracle	N/A	5	IBM Q5 Yorktown (S)

Table 5.1: Overview of parameterized quantum circuit models for machine learning that have been demonstrated experimentally on actual quantum hardware.

5.4 Entanglement entropy of BAS$(2,2)$

The measure of entanglement entropy used in this work is the average von Neumann entropy over all 2-qubit subsets [153]. Consider a 4-qubit pure state $\rho = |\psi\rangle\langle\psi|$ and label the four qubits as A, B, C, and D. Then, the entropy can be computed as

$$\begin{aligned} S_\psi = -\frac{1}{3}\Big[&\mathrm{tr}(\rho_{AB}\log_2\rho_{AB}) + \\ &\mathrm{tr}(\rho_{AC}\log_2\rho_{AC}) + \\ &\mathrm{tr}(\rho_{AD}\log_2\rho_{AD})\Big], \end{aligned} \tag{5.4}$$

where ρ_{XY} is the reduced density matrix for the subset XY. As an example, the 4-qubit GHZ has an entanglement entropy of $S_{GHZ} = 1$. Now consider a pure state encoding the uniform probability distribution over the BAS$(2,2)$ dataset in the computational basis

$$\begin{aligned} |BAS(2,2)\rangle = \frac{1}{\sqrt{6}}\Big(&e^{iu_1}|0000\rangle + e^{iu_2}|0011\rangle + e^{iu_3}|0101\rangle + \\ &e^{iu_4}|1010\rangle + e^{iu_5}|1100\rangle + |1111\rangle\Big). \end{aligned} \tag{5.5}$$

A direct computation shows that the entropy of this state is

$$\begin{aligned} S_{BAS(2,2)} = -\tfrac{1}{9}\Big[&\tfrac{2}{\ln(2)}\sqrt{\tfrac{\cos(u_2-u_3-u_4+u_5)+1}{2}}\tanh^{-1}\left(\sqrt{\tfrac{\cos(u_2-u_3-u_4+u_5)+1}{2}}\right) \\ &+\left(\cos\left(\tfrac{u_1-u_3-u_4}{2}\right)+1\right)\log_2\left(\tfrac{2}{3}\cos^2\left(\tfrac{u_1-u_3-u_4}{4}\right)\right) \\ &+\left(\cos\left(\tfrac{u_1-u_2-u_5}{2}\right)+1\right)\log_2\left(\tfrac{2}{3}\cos^2\left(\tfrac{u_1-u_2-u_5}{4}\right)\right) \\ &+\log_2\left(4+2\sqrt{2}\sqrt{\cos(u_2-u_3-u_4+u_5)+1}\right) \\ &+\log_2\left(4-2\sqrt{2}\sqrt{\cos(u_2-u_3-u_4+u_5)+1}\right) \\ &-\left(\cos\left(\tfrac{u_1-u_3-u_4}{2}\right)-1\right)\log_2\left(\tfrac{2}{3}\sin^2\left(\tfrac{u_1-u_3-u_4}{4}\right)\right) \\ &-\left(\cos\left(\tfrac{u_1-u_2-u_5}{2}\right)-1\right)\log_2\left(\tfrac{2}{3}\sin^2\left(\tfrac{u_1-u_2-u_5}{4}\right)\right) \\ &-\log_2(31104)\Big]. \end{aligned} \tag{5.6}$$

Defining new variables $v_1 = u_2 - u_3 - u_4 + u_5$ and $v_2 = u_1 - u_3 - u_4$, the expression above reduces to

$$\begin{aligned}
S_{BAS(2,2)} = -\tfrac{1}{9}\Big[& \tfrac{2}{\ln(2)}\sqrt{\cos^2\left(\tfrac{v_1}{2}\right)}\tanh^{-1}\left(\sqrt{\cos^2\left(\tfrac{v_1}{2}\right)}\right) \\
& + 2\cos^2\left(\tfrac{v_2}{4}\right)\log_2\left(\tfrac{2}{3}\cos^2\left(\tfrac{v_2}{4}\right)\right) \\
& + 2\cos^2\left(\tfrac{v_2-v_1}{4}\right)\log_2\left(\tfrac{2}{3}\cos^2\left(\tfrac{v_2-v_1}{4}\right)\right) \\
& + \log_2\left(4+2\sqrt{2}\sqrt{\cos(v_1)+1}\right) \\
& + \log_2\left(4-2\sqrt{2}\sqrt{\cos(v_1)+1}\right) \\
& + 2\sin^2\left(\tfrac{v_2}{4}\right)\log_2\left(\tfrac{2}{3}\sin^2\left(\tfrac{v_2}{4}\right)\right) \\
& + 2\sin^2\left(\tfrac{v_2-v_1}{4}\right)\log_2\left(\tfrac{2}{3}\sin^2\left(\tfrac{v_2-v_1}{4}\right)\right) \\
& - \log_2(31104)\Big].
\end{aligned} \tag{5.7}$$

In Fig. 5.1 we graphically show the entropy $S_{BAS(2,2)}$ as a function of the new variables v_1 and v_2. Such a function has extrema

$$\begin{aligned}
\min S_{BAS(2,2)} &= \frac{1}{3}\log_2\left(\frac{27}{2}\right) \approx 1.25163, \\
\max S_{BAS(2,2)} &= \frac{1}{2}\log_2(12) \approx 1.79248.
\end{aligned} \tag{5.8}$$

For the minimum value, $v_1 = v_2 = 0$, which can be obtained setting $u_1 = \cdots = u_5 = 0$. For the maximum value, $v_1 = 4\pi/3$ and $v_2 = 2\pi/3$, which can be obtained setting $u_1 = u_2 = u_3 = 0$ and $u_4 = -u_5 = 2\pi/3$. Interestingly, the maximum of $S_{BAS(2,2)}$ happens to coincide with the maximum entanglement entropy known for any 4-qubit state [153].

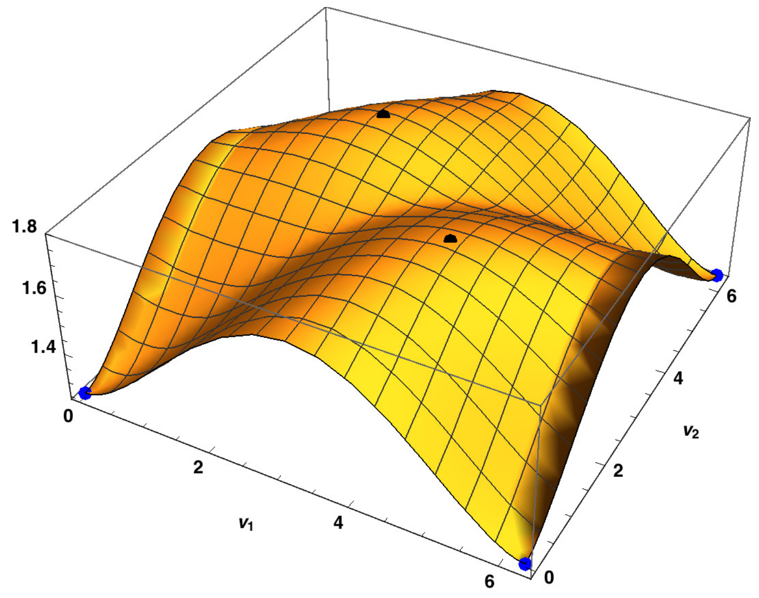


Figure 5.1: Entanglement entropy $S_{BAS(2,2)}$ as a function of variables v_1 and v_2. Points in the domain represent states that encode the BAS(2,2) dataset in the computational basis. The maximum entropy (black dots) attained is 1.79248.

5.5 Best circuits found for BAS(2,2)

In Fig. 5.2 we show circuit diagrams for the analytical solution and for the best circuits found by DDQCL under three different qubit-to-qubit connectivity topologies. The all-to-all circuit with $L = 2$ layers shown in Panel (a) can achieve zero KL divergence for BAS(2,2) by setting $\alpha = \pi^{-1} \arctan\left(2^{-1/2}\right)$ and all single-qubit rotations to zero. DDQCL found an almost optimal solution with $\alpha = 0.2$ and two non-zero R_z rotations. Note that these R_z gates act as the identity on the $|0000\rangle$ state. Panel (b) shows the star circuit with $L = 4$. Panel (c) shows the star circuit with $L = 2$. Circuits (a-c) were simulated *in silico* and executed on trapped ion hardware. The results were then used to generate the histograms in Fig. 3.9 and to compute the qBAS scores in Fig. 3.10.

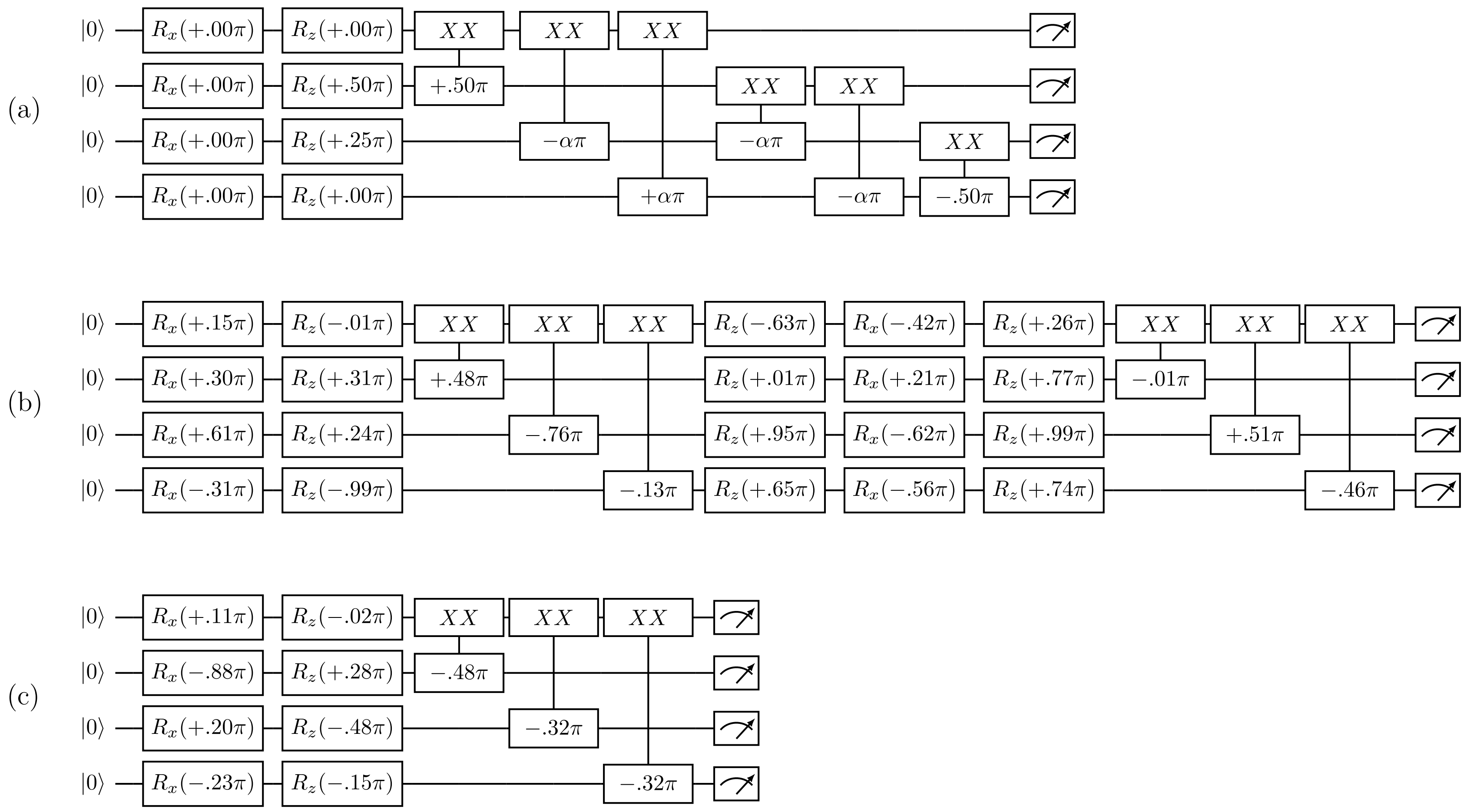

(a)
|0⟩ Rx(+.00π) Rz(+.00π) XX XX XX
|0⟩ Rx(+.00π) Rz(+.50π) +.50π XX XX
|0⟩ Rx(+.00π) Rz(+.25π) −απ −απ XX
|0⟩ Rx(+.00π) Rz(+.00π) +απ −απ −.50π
(b)
|0⟩ Rx(+.15π) Rz(−.01π) XX XX XX Rz(−.63π) Rx(−.42π) Rz(+.26π) XX XX XX
|0⟩ Rx(+.30π) Rz(+.31π) +.48π Rz(+.01π) Rx(+.21π) Rz(+.77π) −.01π
|0⟩ Rx(+.61π) Rz(+.24π) −.76π Rz(+.95π) Rx(−.62π) Rz(+.99π) +.51π
|0⟩ Rx(−.31π) Rz(−.99π) −.13π Rz(+.65π) Rx(−.56π) Rz(+.74π) −.46π
(c)
|0⟩ Rx(+.11π) Rz(−.02π) XX XX XX
|0⟩ Rx(−.88π) Rz(+.28π) −.48π
|0⟩ Rx(+.20π) Rz(−.48π) −.32π
|0⟩ Rx(−.23π) Rz(−.15π) −.32π

Figure 5.2: Circuits found for the BAS(2, 2) under three different qubit-to-qubit connectivity topologies.

5.6 A brief review of circuit learning

Just like classical models, parameterized quantum circuits (PQCs) are trained to perform data-driven tasks. Their learning algorithms can be categorised as either gradient-based or gradient-free. We discuss these two types of algorithms and how they can be applied to optimise the parameters of a circuit $U_{\boldsymbol{\theta}}$.

The task of learning an arbitrary function from data is mathematically expressed as the minimisation of a loss function $L(\boldsymbol{\theta})$, also known as the objective function, with respect to the parameter vector $\boldsymbol{\theta}$. One way to achieve this is by performing an iterative method called gradient descent (GD). In GD, parameters are updated towards the direction of steepest descent of the loss function

$$\boldsymbol{\theta} \longleftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} L, \tag{5.9}$$

where $\nabla_{\boldsymbol{\theta}} L$ is the gradient vector and η is the learning rate – a hyperparameter controlling the size of the update. This procedure is iterated and, assuming suitable conditions, converges to a local minimum of the loss function.

The required partial derivatives can be calculated numerically using a finite difference scheme

$$\frac{\partial L}{\partial \theta_j} \approx \frac{L(\boldsymbol{\theta} + \Delta \boldsymbol{e}_j) - L(\boldsymbol{\theta} - \Delta \boldsymbol{e}_j)}{2\Delta}, \tag{5.10}$$

where Δ is a (small) hyperparameter and $\boldsymbol{e}_j$ is the Cartesian unit vector in the j direction. Note that in order to estimate the gradient vector $\nabla_{\boldsymbol{\theta}} L$, this approach evaluates the loss function twice for each parameter.

Alternatively, Spall's simultaneous perturbation stochastic approximation (SPSA) [195, 90] computes an approximate gradient vector with just two evaluations of the loss function as

$$\frac{\partial L}{\partial \theta_j} \approx \frac{L(\boldsymbol{\theta} + c\boldsymbol{\Delta}) - L(\boldsymbol{\theta} - c\boldsymbol{\Delta})}{2c\Delta_j}, \tag{5.11}$$

where $\boldsymbol{\Delta}$ is a random perturbation vector and c is a (small) hyperparameter.

There are cases when finite difference methods are ill-conditioned and unstable due to truncation and round-off errors. This is one of the reasons why ML relies on the analytical gradient when possible, and it is often calculated with automatic differentiation schemes [196]. The analytical gradient can also be estimated for PQCs, although the equations depend on the choice of parameterization for the gates. For our discussion, we consider circuits $U_{J:1} = U_J \cdots U_1$, where trainable gates are of the from $U_j = \exp\left(-\frac{i}{2}\theta_j P_j\right)$, and where $P_j \in \{I, Z, X, Y\}^{\otimes n}$ is a tensor product of n Pauli matrices. Arguably, this is the most common parameterization found in the literature.

Using this, Li *et al.* [159] proposed a way to efficiently compute analytical gradients in the context of quantum optimal control. Mitarai *et al.* [160] brought this method into the context of supervised learning. Recall that the PQC's output is a set of expectation values $\langle M_k \rangle_{\boldsymbol{\theta}}$. Using the chain rule we can write the derivative $\frac{\partial L}{\partial \theta_j}$ as a function of the derivatives of the expectation values $\frac{\partial \langle M_k \rangle_{\boldsymbol{\theta}}}{\partial \theta_j}$. Each of these quantities can be estimated on quantum hardware using the so called 'parameter shift rule'

$$\frac{\partial \langle M_k \rangle_{\boldsymbol{\theta}}}{\partial \theta_j} = \frac{\langle M_k \rangle_{\boldsymbol{\theta}+\frac{\pi}{2}\boldsymbol{e}_j} - \langle M_k \rangle_{\boldsymbol{\theta}-\frac{\pi}{2}\boldsymbol{e}_j}}{2}, \tag{5.12}$$

where subscripts $\boldsymbol{\theta} \pm \frac{\pi}{2}\boldsymbol{e}_j$ indicate the shifted parameter vector to use for the evaluation (see Schuld *et al.* [161] for a detailed derivation). Note that this estimation can be performed by executing two circuits.

An alternative method can estimate the partial derivative with a single circuit, but at the cost of adding an ancilla qubit. A simple derivation using the gate parameterization introduced above (e.g., see Farhi and Neven [127]) shows that the partial derivative can be written as

$$\frac{\partial \langle M_k \rangle_{\boldsymbol{\theta}}}{\partial \theta_j} = \mathrm{Im}\left(\mathrm{tr}\left(M_k U_{J:j+1} P_j U_{j:1} |0\rangle\langle 0| U_{J:1}^{\dagger}\right)\right). \tag{5.13}$$

This can be thought of as an *indirect measurement* and can be evaluated using the Hadamard test shown in Fig. 5.3. This method can be generalised to compute higher order derivatives, as presented for example by Dallaire-Demers and Killoran [162], and with alternative gate parameterizations, as done for example by Schuld *et al.* [197].

We shall note that despite the apparent simplicity of the circuit in Fig. 5.3, the actual implementation of Hadamard tests may be challenging due to non-trivial controlled gates. Coherence must be guaranteed in order for quantum interference

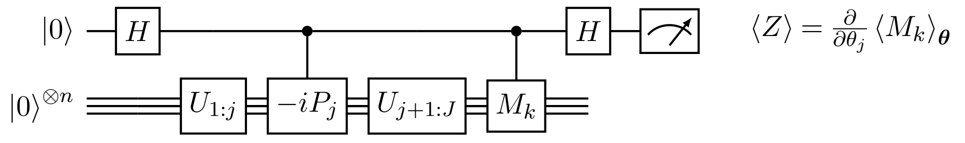

$|0\rangle$
$|0\rangle^{\otimes n}$
H
$U_{1:j}$
$-iP_j$
$U_{j+1:J}$
M_k
H
$\langle Z \rangle = \frac{\partial}{\partial \theta_j} \langle M_k \rangle_{\boldsymbol{\theta}}$

Figure 5.3: The Hadamard test can be used to estimate the partial derivative of an expectation $\langle M_k \rangle_{\boldsymbol{\theta}}$ with respect to the parameter θ_j. Here we show a simple case where gates are of the form $U_j = \exp\left(-\frac{i}{2}\theta_j P_j\right)$ and where both P_j and M_k are tensor products of Pauli matrices. It can be shown that measurements of the Z Pauli observable on the ancilla qubit yield Eq. (5.13), the desired partial derivative. Hadamard tests can be designed to estimate higher order derivatives and to work with different measurements and gate parameterizations.

to produce the desired result. Mitarai and Fujii [198] proposed a method for replacing a class of indirect measurements with direct ones. Instead of an interference circuit one can execute, in some cases, multiple simpler circuits that are suitable for implementations on NISQ computers. The 'parameters shift rule' in Eq. (5.12) is nothing but the direct version of the measurement in Eq. (5.13).

Compared to finite difference and SPSA, the analytical gradient has the advantage of providing an unbiased estimator. Additionally, Harrow and Napp [199] found evidence that training PQCs using the analytical gradient outperforms any finite difference method. This is done by showing that for n qubits and precision ϵ, the query cost of an oracle for convex optimisation in the vicinity of the optimum scales as $\mathcal{O}(\frac{n^2}{\epsilon})$ for the analytical gradient, whereas finite difference needs at least $\Omega(\frac{n^3}{\epsilon^2})$ calls to the oracle. In practice though, it is found that SPSA performs well in small-scale noisy experimental settings (e.g. see Kandala *et al.* [120] and Havlíček *et al.* [190]).

Particular attention should be given to the problems of exploding and vanishing gradients which are well-known to the machine learning community. Classical models, in particular recurrent neural networks, are often constrained to perform unitary operations so that their gradients cannot explode (see Wisdom *et al.* [200] for an example). PQCs naturally implement unitary operations and therefore avoid the exploding gradient problem altogether. On the other hand, McClean *et al.* [164] showed that random circuits of reasonable depth lead to an optimisation landscape with exponentially large plateaus of vanishing gradients with an exponentially decaying variance. This can be understood as a consequence of Levy's lemma [201] which states that a random variable that depends on many independent variables is essentially constant. The learning algorithm is thus unable to estimate the gradient and may perform a random walk in parameter space. While this limits the effectiveness of PQCs initialised at random, the use of highly structured circuits could alleviate the problem (e.g., see Grant *et al.* [202] for a structured initialisation strategy).

We shall stress here that in hybrid systems parameter updates are performed classically. This implies that some of the most successful deep learning algorithms can be readily used for training PQC models. For the case of gradient-based optimisation, heuristics such as stochastic gradient descent [203], resilient backpropagation [167], and adaptive momentum estimation (Adam) [204], have already been applied with success. These were designed to deal with issues of practical importance such as large datasets, large noise in gradient estimates, and the need to find adaptive learning rates in Eq. (5.9). In practice, these choices can reduce the time for successful training from days to hours.

There are cases where gradient-based optimisation may be challenging. For example, in a noisy experimental setting the loss function may be highly non-smooth and not suitable for GD. As another example, the objective function may be itself unknown and therefore should be treated as a black-box. In these cases, circuit learning can be carried out by gradient-free methods. A well-known method of this type is particle swarm optimisation (PSO) [205]. Here the system is initialised with a number of random solutions called particles, each one moving through solution space with a certain velocity. The trajectory of each particle is adjusted according to its own experience and that of other particles so that they converge to a local minima. Another popular method is Bayesian optimisation (BO) [206]. BO uses evaluations of the objective function to construct a model of the function itself. Subsequent evaluations can be chosen either to improve the model or to find a minima.

Zhu *et al.* [175] compared BO and PSO for training a generative model on a trapped ion quantum computer. While BO outperformed PSO in their setting, they found that the large number of parameters challenges both optimisers. They showed that an ideal simulated system is not significantly faster than the experimental system, indicating that the actual bottleneck is the classical optimiser. Leyton-Ortega *et al.* [178] learned a generative model on a superconducting quantum computer and compared the gradient-free methods of zeroth-order optimisation package (ZOOpt) [207] and stochastic hill-climbing (SHC), with GD using Adam. They found that on average ZOOpt achieves the lowest loss on their hardware. They argued that the main optimisation challenge is to overcome the variance of the loss function which is due to random parameter initialisation, hardware noise, and finite number of measurements.

Genetic algorithms [208] are another large class of gradient-free optimisation algorithms. At each step, candidate solutions are evolved using biology-inspired operations such as recombination, mutation, and natural selection. When used to train PQCs, genetic algorithms define a set of allowed gates and the maximum number to be employed. Lamata *et al.* [209] suggested the use of genetic algorithms to train a PQC model for compression using a universal set of single- and two-qubit gates. Ding *et al.* [193] validated the idea experimentally by deploying a pre-trained PQC model on a superconducting computer and found that using a subsequent genetic algorithms improves its fidelity.

To conclude, we note that optimisation algorithms should be tailored for PQCs if we want to achieve better scalability. Very recent work has been approaching circuit learning from this perspective (e.g., see Ostaszewski *et al.* [210] and Nakanishi *et al.* [211]).

5.7 An optimal discriminator circuit

Let us briefly recall the quantum generative adversarial network. We have two circuits, the generator and the discriminator, and a target state. The target state ρ_t is prepared with probability $P(t)$, while the generated state ρ_g is prepared with probability $P(g)$. The discriminator has to successfully distinguish each state or, in other words, he must find the measurement that minimises the probability of labelling error.

Helstrom [136] observed that the optimal POVM that distinguishes two states has the following particular form; let E_0 and E_1 be the POVM elements attaining the minimum in

$$P^*_{err} = \min_{\{E_0,E_1\}} \mathrm{tr}[E_1\rho_t]P(t) + \mathrm{tr}[E_0\rho_g]P(g), \tag{5.14}$$

then both elements are diagonal in a basis that also diagonalizes the Hermitian operator

$$\Gamma = P(t)\rho_t - P(g)\rho_g. \tag{5.15}$$

As pointed out in Ref. [157], in this basis one can construct E_0 by specifying its diagonal elements λ_j according to the rule

$$\begin{aligned} \lambda_j &= 1 \quad \text{when} \quad \gamma_j < 0 \\ \lambda_j &= 0 \quad \text{when} \quad \gamma_j \geq 0, \end{aligned} \tag{5.16}$$

where γ_j are the diagonal elements of Γ. The operator E_1 is then obtained via the relationship $I - E_0$. Hence we can construct the optimal measurement operator if we have access to the operator Γ, and provided that we can diagonalize it.

For pure states $\rho_t = |\psi_t\rangle\langle\psi_t|$ and $\rho_g = |\psi_g\rangle\langle\psi_g|$, we observe that $\mathrm{tr}[\Gamma\rho_g] = P(t)|\langle\psi_g|\psi_t\rangle|^2 - P(g)$ and $\mathrm{tr}[\Gamma\rho_t] = P(t) - P(g)|\langle\psi_g|\psi_t\rangle|^2$. Under the assumption of equal prior probabilities of $\frac{1}{2}$, Eq. (5.14) is minimised when the overlap of the two states is maximised. Since the prior probabilities are hyper-parameters, we can set them to $\frac{1}{2}$ and use the SWAP test [212] to compute the overlap. This procedure effectively implements an optimal discriminator for pure states and provides a strong learning signal to the generator[1].

Figure 5.4 shows the standard circuit for the SWAP test. Note that this bears several disadvantages compared to our adversarial method. In order to perform the SWAP test, we need to access both ρ_t and ρ_g simultaneously. This also requires the use of two registers for a total of $2n+1$ qubits, which is significantly more than the $n+1$ qubits required in the near-term adversarial method.

The standard circuit for the SWAP test requires the ability to perform non-trivial controlled gates in a fault-tolerant way. A potential solution is to find a low-depth circuit for the SWAP test. In Ref. [213] the authors implemented such via supervised circuit learning. As pointed out in their work, this requires (a) an order of 2^{2n} training examples for states of n qubits, and (b) each training example

[1]To the best of our knowledge, if ρ_t and ρ_g are mixed states, there is no simple way to find the Helstrom measurement in Eq. (5.15), nor to compute the error probability in Eq. (5.14). The SWAP test can still be used to obtain useful bounds (see Nielsen and Chuang [44], Chapter 9).

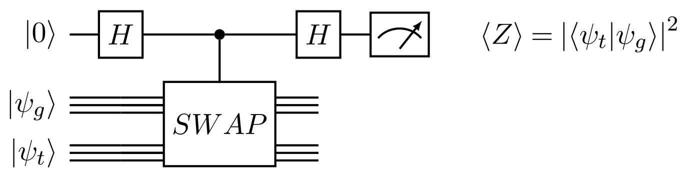


Figure 5.4: The standard circuit for the SWAP test. Measurements of the Z Pauli observable on the ancilla qubit yield the overlap between two pure states in input.

be given by the actual overlap between two states, requiring a circuit which gives the answer to the problem we are trying to solve. We believe that their approach is not suitable for our task.

One could alternatively consider the possibility of implementing a discriminator via distance measurements based on random projections, i.e., Johnson-Lindenstrauss transformations [214]. This would require a reduced amount of resources and could be adapted for the adversarial method. As an example, we could apply a quantum channel to coherently reduce the dimensionality of the input state and then apply the state discrimination procedure in the lower-dimensional space. However, in Ref. [215] the authors proved that such an operation cannot be performed by a quantum channel. One way to think about this is that the Johnson-Lindenstrauss transformation is a projection onto a small random subspace, and that is equivalent to a projective measurement. As the subspace is exponentially smaller than the initial Hilbert space, the probability that this projection preserves the distances is very small.

5.8 Software used

This Thesis was typeset using LaTeX and BibTeX, and was written on Overleaf. Circuits were drawn with `Quantikz` [216] and all plots were made in `matplotlib` [217]. Simulations involved many software packages, most notably `scikit-learn` [218], `PySwarms` [148], `QuTiP2` [147], and `GNU Parallel` [219]. The quantum annealers hosted by NASA Ames Research Center and the trapped ion computer hosted by University of Maryland were accessed via their corresponding Application Programming Interfaces.

Bibliography

[1] M. Mohseni, P. Read, H. Neven *et al.* Commercialize quantum technologies in five years. *Nature*, **543**, 171 (2017).

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105 (2012).

[3] G. Hinton, L. Deng, D. Yu *et al.* Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, **29**, 82 (2012).

[4] J. Levinson, J. Askeland, J. Becker *et al.* Towards fully autonomous driving: Systems and algorithms. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 163–168. IEEE (2011).

[5] A. Esteva, B. Kuprel, R. A. Novoa *et al.* Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, **542**, 115 (2017).

[6] T. Ching, D. S. Himmelstein, B. K. Beaulieu-Jones *et al.* Opportunities and obstacles for deep learning in biology and medicine. *bioRxiv* (2017).

[7] V. Mnih, K. Kavukcuoglu, D. Silver *et al.* Human-level control through deep reinforcement learning. *Nature*, **518**, 529 (2015).

[8] P. Rebentrost, M. Mohseni, and S. Lloyd. Quantum support vector machine for big data classification. *Physical Review Letters*, **113**, 130503 (2014).

[9] G. Wang. Quantum algorithm for linear regression. *Physical Review A*, **96**, 012335 (2017).

[10] N. Wiebe, D. Braun, and S. Lloyd. Quantum algorithm for data fitting. *Physical review letters*, **109**, 050505 (2012).

[11] M. Schuld, I. Sinayskiy, and F. Petruccione. Prediction by linear regression on a quantum computer. *Physical Review A*, **94**, 022342 (2016).

[12] Z. Zhao, J. K. Fitzsimons, and J. F. Fitzsimons. Quantum-assisted gaussian process regression. *Physical Review A*, **99**, 052331 (2019).

[13] S. Lloyd, M. Mohseni, and P. Rebentrost. Quantum algorithms for supervised and unsupervised machine learning. *arXiv:1307.0411* (2013).

[14] S. Lloyd, M. Mohseni, and P. Rebentrost. Quantum principal component analysis. *Nature Physics*, **10**, 631 (2014).

[15] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, **35**, 1798 (2013).

[16] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, **521**, 436 (2015).

[17] The Caltech Archives. Richard Feynman's blackboard at time of his death. `http://archives-dc.library.caltech.edu/islandora/object/ct1%3A483` (1988). Accessed on 2018-05-01.

[18] D. Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, **82**, 273 (1996).

[19] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, **112**, 859 (2017).

[20] Y. Bengio, G. Mesnil, Y. Dauphin *et al.* Better mixing via deep representations. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 552–560 (2013).

[21] T. Kadowaki and H. Nishimori. Quantum annealing in the transverse ising model. *Physical Review E*, **58**, 5355 (1998).

[22] A. B. Finnila, M. A. Gomez, C. Sebenik *et al.* Quantum annealing: A new method for minimizing multidimensional functions. *Chemical Physics Letters*, **219**, 343 (1994).

[23] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006).

[24] I. G. Y. Bengio and A. Courville. Deep learning (2016). MIT Press, URL `http://www.deeplearningbook.org`.

[25] Z. Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, **521**, 452 (2015).

[26] S. Boixo, S. V. Isakov, V. N. Smelyanskiy *et al.* Characterizing quantum supremacy in near-term devices. *Nature Physics*, **14**, 595 (2018).

[27] M. Gu, K. Wiesner, E. Rieper *et al.* Quantum mechanics can reduce the complexity of classical models. *Nature Communications*, **3**, 762 (2012).

[28] M. S. Palsson, M. Gu, J. Ho *et al.* Experimentally modeling stochastic processes with less memory by the use of a quantum processor. *Science Advances*, **3**, e1601302 (2017).

[29] T. J. Elliott and M. Gu. Occam's vorpal quantum razor: Memory reduction when simulating continuous-time stochastic processes with quantum devices. *arXiv preprint arXiv:1704.04231* (2017).

[30] H. J. Kappen. Learning quantum models from quantum or classical data. *arXiv preprint arXiv:1803.11278* (2018).

[31] K. P. Burnham and D. R. Anderson. *Model selection and multimodel inference: a practical information-theoretic approach.* Springer Science & Business Media (2003).

[32] E. Aïmeur, G. Brassard, and S. Gambs. Machine learning in a quantum world. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 431–442. Springer (2006).

[33] H. J. Kimble. The quantum internet. *Nature*, **453**, 1023 (2008).

[34] M. H. Amin, E. Andriyash, J. Rolfe *et al.* Quantum Boltzmann machine. *Physical Review X*, **8**, 021050 (2018).

[35] M. Kieferová and N. Wiebe. Tomography and generative training with quantum Boltzmann machines. *Physical Review A*, **96**, 062327 (2017).

[36] S. Aaronson. The learnability of quantum states. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 463, pages 3089–3114. The Royal Society, The Royal Society London (2007).

[37] A. Rocchetto, S. Aaronson, S. Severini *et al.* Experimental learning of quantum states. *Science Advances*, **5** (2019).

[38] A. Lund, M. J. Bremner, and T. Ralph. Quantum sampling problems, bosonsampling and quantum supremacy. *npj Quantum Information*, **3**, 15 (2017).

[39] A. W. Harrow and A. Montanaro. Quantum computational supremacy. *Nature*, **549**, 203 (2017).

[40] Y. Du, M.-H. Hsieh, T. Liu *et al.* The expressive power of parameterized quantum circuits. *arXiv preprint arXiv:1810.11922* (2018).

[41] B. Coyle, D. Mills, V. Danos *et al.* The born supremacy: Quantum advantage and training of an ising born machine. *arXiv preprint arXiv:1904.02214* (2019).

[42] P. Mehta, M. Bukov, C.-H. Wang *et al.* A high-bias, low-variance introduction to machine learning for physicists. *Physics Reports* (2019).

[43] P. Hauke, H. G. Katzgraber, W. Lechner *et al.* Perspectives of quantum annealing: Methods and implementations. *arXiv preprint arXiv:1903.06559* (2019).

[44] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition.* Cambridge University Press, New York, NY, USA, 10th edition (2011).

[45] R. Salakhutdinov. Learning deep generative models. *Annual Review of Statistics and Its Application*, **2**, 361 (2015).

[46] A. Sinclair and M. Jerrum. Approximate counting, uniform generation and rapidly mixing markov chains. *Inf. Comput.*, **82**, 93 (1989).

[47] A. Frigessi, F. Martinelli, and J. Stander. Computational complexity of Markov chain Monte Carlo methods for finite Markov random fields. *Biometrika*, **84**, 1 (1997).

[48] E. Farhi, J. Goldstone, S. Gutmann *et al.* A quantum adiabatic evolution algorithm applied to random instances of an NP-Complete problem. *Science*, **292**, 472 (2001).

[49] F. Gaitan and L. Clark. Ramsey numbers and adiabatic quantum computing. *Physical Review Letters*, **108**, 010501 (2012).

[50] A. Perdomo-Ortiz, N. Dickson, M. Drew-Brook *et al.* Finding low-energy conformations of lattice protein models by quantum annealing. *Scientific Reports*, **2** (2012).

[51] Z. Bian, F. Chudak, R. Israel *et al.* Discrete optimization using quantum annealing on sparse ising models. *Frontiers in Physics*, **2** (2014).

[52] B. O'Gorman, A. Perdomo-Ortiz, R. Babbush *et al.* Bayesian network structure learning using quantum annealing. *The European Physical Journal Special Topics*, **224**, 163 (2015).

[53] E. G. Rieffel, D. Venturelli, B. O'Gorman *et al.* A case study in programming a quantum annealer for hard operational planning problems. *Quantum Information Processing*, **14**, 1 (2015).

[54] A. Perdomo-Ortiz, J. Fluegemann, S. Narasimhan *et al.* A quantum annealing approach for fault detection and diagnosis of graph-based systems. *Eur. Phys. J. Special Topics*, **224**, 131 (2015).

[55] A. Perdomo-Ortiz, J. Fluegemann, R. Biswas *et al.* A performance estimator for quantum annealers: gauge selection and parameter setting. *arXiv:1503.01083* (2015).

[56] D. Venturelli, S. Mandrà, S. Knysh *et al.* Quantum optimization of fully connected spin glasses. *Physical Review X*, **5**, 031040 (2015).

[57] D. Venturelli, D. J. Marchand, and G. Rojo. Quantum annealing implementation of job-shop scheduling. *arXiv:1506.08479* (2015).

[58] A. Perdomo-Ortiz, A. Feldman, A. Ozaeta *et al.* Readiness of quantum optimization machines for industrial applications. *Physical Review Applied*, **12**, 014004 (2019).

[59] T. Albash, S. Boixo, D. A. Lidar *et al.* Quantum adiabatic markovian master equations. *New Journal of Physics*, **14**, 123016 (2012).

[60] V. N. Smelyanskiy, D. Venturelli, A. Perdomo-Ortiz *et al.* Quantum annealing via environment-mediated quantum diffusion. *Physical Review Letters*, **118**, 066802 (2017).

[61] M. H. Amin. Searching for quantum speedup in quasistatic quantum annealers. *Physical Review A*, **92**, 052323 (2015).

[62] M. Benedetti, J. Realpe-Gómez, R. Biswas *et al.* Estimation of effective temperatures in quantum annealers for sampling applications: A case study with possible applications in deep learning. *Physical Review A*, **94**, 022308 (2016).

[63] J. Raymond, S. Yarkoni, and E. Andriyash. Global warming: Temperature estimation in annealers. *arXiv:1606.00919* (2016).

[64] M. W. Johnson, M. H. S. Amin, S. Gildert *et al.* Quantum annealing with manufactured spins. *Nature*, **473**, 194 (2011).

[65] S. Boixo, V. N. Smelyanskiy, A. Shabani *et al.* Computational multiqubit tunnelling in programmable quantum annealers. *Nature Communications*, **7**, 10327 (2014).

[66] D. Korenkevych, Y. Xue, Z. Bian *et al.* Benchmarking quantum hardware for training of fully visible boltzmann machines. *arXiv preprint arXiv:1611.04528* (2016).

[67] Z. Bian, F. Chudak, W. G. Macready *et al.* The Ising model: teaching an old problem new tricks. *Technical report*, D-Wave Systems (2010).

[68] M. Denil and N. De Freitas. Toward the implementation of a quantum RBM. In *NIPS Deep Learning and Unsupervised Feature Learning Workshop* (2014).

[69] V. Dumoulin, I. J. Goodfellow, A. C. Courville *et al.* On the challenges of physical implementations of RBMs. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 1199–1205 (2014).

[70] Alejandro, B. O'Gorman, J. Fluegemann *et al.* Determination and correction of persistent biases in quantum annealers. *Scientific Reports*, **6**, 18628 (2016).

[71] S. Mandrà, Z. Zhu, and H. G. Katzgraber. Exponentially biased ground-state sampling of quantum annealing machines with transverse-field driving hamiltonians. *Physical Review Letters*, **118**, 070502 (2017).

[72] Y. Bengio *et al.* Learning deep architectures for ai. *Foundations and trend in Machine Learning*, **2**, 1 (2009).

[73] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, **18**, 1527 (2006).

[74] G. E. Hinton, P. Dayan, B. J. Frey *et al.* The wake-sleep algorithm for unsupervised neural networks. *Science*, **268**, 1158 (1995).

[75] P. Dayan, G. E. Hinton, R. M. Neal *et al.* The helmholtz machine. *Neural computation*, **7**, 889 (1995).

[76] J. Bornschein, S. Shabanian, A. Fischer *et al.* Bidirectional helmholtz machines. In *International Conference on Machine Learning*, pages 2511–2519 (2016).

[77] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, **9**, 147 (1985).

[78] R. Salakhutdinov and G. Hinton. Deep boltzmann machines. In *Artificial Intelligence and Statistics*, pages 448–455 (2009).

[79] C. Sun, A. Shrivastava, S. Singh *et al.* Revisiting unreasonable effectiveness of data in deep learning era. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 843–852 (2017).

[80] V. Choi. Minor-embedding in adiabatic quantum computation: I. the parameter setting problem. *Quantum Information Processing*, **7**, 193 (2008).

[81] V. Choi. Minor-embedding in adiabatic quantum computation: Ii. minor-universal graph design. *Quantum Information Processing*, **10**, 343 (2011).

[82] D. Eppstein. Finding large clique minors is hard. *J. Graph Algorithms Appl.*, **13**, 197 (2008).

[83] J. Matoušek and R. Thomas. On the complexity of finding iso-and other morphisms for partial k-trees. *Discrete Mathematics*, **108**, 343 (1992).

[84] J. Cai, W. G. Macready, and A. Roy. A practical heuristic for finding graph minors. *arXiv:1406.2741* (2014).

[85] M. Mezard and A. Montanari. *Information, Physics, and Computation*. Oxford University Press, Inc., New York, NY, USA (2009).

[86] H. Nishimori. *Statistical Physics of Spin Glasses and Information Processing: An Introduction*. International series of monographs on physics. Oxford University Press (2001).

[87] D. Sherrington and S. Kirkpatrick. Solvable model of a spin-glass. *Physical review letters*, **35**, 1792 (1975).

[88] M. A. Carreira-Perpinan and G. E. Hinton. On contrastive divergence learning. In *Aistats*, volume 10, pages 33–40. Citeseer (2005).

[89] G. E. Hinton. A practical guide to training restricted boltzmann machines. In *Neural Networks: Tricks of the Trade (2nd ed.)*, volume 7700 of *Lecture Notes in Computer Science*, pages 599–619. Springer (2012).

[90] J. C. Spall. Adaptive stochastic approximation by the simultaneous perturbation method. *IEEE transactions on automatic control*, **45**, 1839 (2000).

[91] J. C. Spall. *Introduction to Stochastic Search and Optimization*. John Wiley & Sons, Inc., New York, NY, USA, 1 edition (2003).

[92] E. T. Jaynes. Information theory and statistical mechanics. ii. *Physical review*, **108**, 171 (1957).

[93] E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, **106**, 620 (1957).

[94] E. Jaynes and G. Bretthorst. *Probability Theory: The Logic of Science*. Cambridge University Press (2003).

[95] N. Chancellor, S. Szoke, W. Vinci *et al.* Maximum-entropy inference with a programmable annealer. *Scientific reports*, **6** (2016).

[96] G. F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial intelligence*, **42**, 393 (1990).

[97] P. M. Long and R. Servedio. Restricted boltzmann machines are hard to approximately evaluate or simulate. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 703–710. Omnipress (2010).

[98] M. Lichman. UCI machine learning repository (2013). URL `http://archive.ics.uci.edu/ml`.

[99] D. J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY, USA (2002).

[100] M. Mezard, G. Parisi, and M. Virasoro. *Spin Glass Theory and Beyond*. Lecture Notes in Physics Series. World Scientific (1987).

[101] F. Ricci-Tersenghi. The bethe approximation for solving the inverse ising problem: a comparison with other inference methods. *Journal of Statistical Mechanics: Theory and Experiment*, **2012**, P08015 (2012).

[102] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, **220**, 671 (1983).

[103] E. Schneidman, M. J. Berry, R. Segev *et al.* Weak pairwise correlations imply strongly correlated network states in a neural population. *Nature*, **440**, 1007 (2006).

[104] M. Mézard and T. Mora. Constraint satisfaction problems and neural networks: A statistical physics perspective. *Journal of Physiology-Paris*, **103**, 107 (2009). Neuromathematics of Vision.

[105] I. Mastromatteo and M. Marsili. On the criticality of inferred models. *Journal of Statistical Mechanics: Theory and Experiment*, **2011**, P10012 (2011).

[106] C. Klymko, B. Sullivan, and T. Humble. Adiabatic quantum programming: minor embedding with hard faults. *Quantum Information Processing*, pages 1–21 (2013).

[107] W. Lechner, P. Hauke, and P. Zoller. A quantum annealing architecture with all-to-all connectivity from local interactions. *Science advances*, **1**, e1500838 (2015).

[108] J. Job and D. Lidar. Test-driving 1000 qubits. *Quantum Science and Technology*, **3**, 030501 (2018).

[109] H. G. Katzgraber. Viewing vanilla quantum annealing through spin glasses. *Quantum Science and Technology*, **3**, 030505 (2018).

[110] T. F. Rønnow, Z. Wang, J. Job *et al.* Defining and detecting quantum speedup. *Science*, **345**, 420 (2014).

[111] T. E. Potok, C. D. Schuman, S. R. Young *et al.* 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*, pages 47–55. IEEE Press (2016).

[112] J. Bornschein and Y. Bengio. Reweighted wake-sleep. *arXiv preprint arXiv:1406.2751* (2014).

[113] R. Salakhutdinov and H. Larochelle. Efficient learning of deep boltzmann machines. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 693–700 (2010).

[114] L. Younes. On the convergence of markovian stochastic algorithms with rapidly decreasing ergodicity rates. *Stochastics: An International Journal of Probability and Stochastic Processes*, **65**, 177 (1999).

[115] A sub-sampled version of the MNIST dataset (Accessed: August 2017).

[116] I. Goodfellow, J. Pouget-Abadie, M. Mirza *et al.* Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes *et al.*, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc. (2014).

[117] J. Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, **2**, 79 (2018).

[118] A. Peruzzo, J. McClean, P. Shadbolt *et al.* A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, **5** (2014).

[119] P. O'Malley, R. Babbush, I. Kivlichan *et al.* Scalable quantum simulation of molecular energies. *Physical Review X*, **6**, 031007 (2016).

[120] A. Kandala, A. Mezzacapo, K. Temme *et al.* Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, **549**, 242 (2017).

[121] E. Farhi, J. Goldstone, and S. Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028* (2014).

[122] N. Moll, P. Barkoutsos, L. S. Bishop *et al.* Quantum optimization using variational algorithms on near-term quantum devices. *Quantum Science and Technology*, **3**, 030503 (2018).

[123] J. S. Otterbach, R. Manenti, N. Alidoust *et al.* Unsupervised machine learning on a hybrid quantum computer. *arXiv preprint arXiv:1712.05771* (2017).

[124] J. Bang, J. Lim, M. Kim *et al.* Quantum learning machine. *arXiv preprint arXiv:0803.2976* (2008).

[125] S. Gammelmark and K. Mølmer. Quantum learning by measurement and feedback. *New Journal of Physics*, **11**, 033017 (2009).

[126] M. Schuld and N. Killoran. Quantum machine learning in feature hilbert spaces. *Physical Review Letters*, **122**, 040504 (2019).

[127] E. Farhi and H. Neven. Classification with quantum neural networks on near term processors. *arXiv preprint arXiv:1802.06002* (2018).

[128] S. Cheng, J. Chen, and L. Wang. Information perspective to probabilistic modeling: Boltzmann machines versus born machines. *Entropy*, **20**, 583 (2018).

[129] Z.-Y. Han, J. Wang, H. Fan *et al.* Unsupervised generative modeling using matrix product states. *Physical Review X*, **8** (2018).

[130] E. Stoudenmire and D. J. Schwab. Supervised learning with tensor networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg *et al.*, editors, *Advances in Neural Information Processing Systems 29*, pages 4799–4807. Curran Associates, Inc. (2016).

[131] D. Liu, S.-J. Ran, P. Wittek *et al.* Machine learning by unitary tensor network of hierarchical tree structure. *New Journal of Physics*, **21**, 073059 (2019).

[132] X. Gao, Z. Zhang, and L. Duan. A quantum machine learning algorithm based on generative models. *Science Advances*, **4** (2018).

[133] L. S. Bishop, S. Bravyi, A. Cross *et al.* Quantum volume (2017).

[134] I. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160* (2016).

[135] S. Lloyd and C. Weedbrook. Quantum generative adversarial learning. *Physical Review Letters*, **121**, 040502 (2018).

[136] C. W. Helstrom. Quantum detection and estimation theory. *Journal of Statistical Physics*, **1**, 231 (1969).

[137] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, **22**, 79 (1951).

[138] P. S. Laplace. *Essai philosophique sur les probabilités*. Cambridge University Press (1825).

[139] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4 (1995).

[140] Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE*

World Congress on Computational Intelligence (Cat. No.98TH8360), pages 69–73 (1998).

[141] J. I. Colless, V. V. Ramasesh, D. Dahlen *et al.* Computation of molecular spectra on a quantum processor with an error-resilient algorithm. *Physical Review X*, **8**, 011021 (2018).

[142] A. Sørensen and K. Mølmer. Quantum computation with ions in thermal motion. *Physical Review Letters*, **82**, 1971 (1999).

[143] A. Sørensen and K. Mølmer. Entanglement and quantum computation with ions in thermal motion. *Physical Review A*, **62**, 022311 (2000).

[144] J. Benhelm, G. Kirchmair, C. F. Roos *et al.* Towards fault-tolerant quantum computing with trapped ions. *Nature Physics*, **4**, 463 EP (2008).

[145] D. Maslov and Y. Nam. Use of global interactions in efficient quantum circuit constructions. *New Journal of Physics* (2017).

[146] S. Debnath, N. M. Linke, C. Figgatt *et al.* Demonstration of a small programmable quantum computer with atomic qubits. *Nature*, **536**, 63 EP (2016).

[147] J. Johansson, P. Nation, and F. Nori. QuTiP 2: A Python framework for the dynamics of open quantum systems. *Computer Physics Communications*, **184**, 1234 (2013).

[148] L. J. V. Miranda. Pyswarms, a research-toolkit for particle swarm optimization in python (2017). URL `https://zenodo.org/badge/latestdoi/97002861`.

[149] D. M. Greenberger, M. A. Horne, A. Shimony *et al.* Bell's theorem without inequalities. *American Journal of Physics*, **58**, 1131 (1990).

[150] T. Monz, P. Schindler, J. T. Barreiro *et al.* 14-qubit entanglement: Creation and coherence. *Physical Review Letters*, **106**, 130506 (2011).

[151] A. Ozaeta and P. L. McMahon. Decoherence of up to 8-qubit entangled states in a 16-qubit superconducting quantum processor. *Quantum Science and Technology*, **4**, 025015 (2019).

[152] I. Mastromatteo. On the typical properties of inverse problems in statistical mechanics. *arXiv preprint arXiv:1311.0190* (2013).

[153] A. Higuchi and A. W. Sudbery. How entangled can two couples get? *Physics Letters A*, **273**, 213 (2000).

[154] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, **40**, 99 (2000).

[155] E. Levina and P. Bickel. The earth mover's distance is the mallows distance: Some insights from statistics. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 251–256. IEEE (2001).

[156] O. Pele and M. Werman. Fast and robust earth mover's distances. In *2009 IEEE 12th International Conference on Computer Vision*, pages 460–467. IEEE (2009).

[157] C. A. Fuchs. Distinguishability and accessible information in quantum theory. *arXiv preprint quant-ph/9601020* (1996).

[158] M. Neumark. Spectral functions of a symmetric operator. *Izvestiya Rossiiskoi Akademii Nauk. Seriya Matematicheskaya*, **4**, 277 (1940).

[159] J. Li, X. Yang, X. Peng *et al.* Hybrid quantum-classical approach to quantum optimal control. *Physical review letters*, **118**, 150503 (2017).

[160] K. Mitarai, M. Negoro, M. Kitagawa *et al.* Quantum circuit learning. *Physical Review A*, **98**, 032309 (2018).

[161] M. Schuld, V. Bergholm, C. Gogolin *et al.* Evaluating analytic gradients on quantum hardware. *Physical Review A*, **99**, 032331 (2019).

[162] P.-L. Dallaire-Demers and N. Killoran. Quantum generative adversarial networks. *Physical Review A*, **98**, 012324 (2018).

[163] J.-G. Liu and L. Wang. Differentiable learning of quantum circuit born machines. *Physical Review A*, **98**, 062324 (2018).

[164] J. R. McClean, S. Boixo, V. N. Smelyanskiy *et al.* Barren plateaus in quantum neural network training landscapes. *Nature communications*, **9**, 4812 (2018).

[165] E. Grant, M. Benedetti, S. Cao *et al.* Hierarchical quantum classifiers. *npj Quantum Information*, **4**, 65 (2018).

[166] H. Chen, L. Wossnig, S. Severini *et al.* Universal discriminative quantum neural networks. *arXiv preprint arXiv:1805.08654* (2018).

[167] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Neural Networks, 1993., IEEE International Conference on*, pages 586–591. IEEE (1993).

[168] C. Igel and M. Hüsken. Improving the Rprop learning algorithm. In *Proceedings of the second international ICSC symposium on neural computation (NC 2000)*, volume 2000, pages 115–121. Citeseer (2000).

[169] L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. In *International Conference on Learning Representations (ICLR 2016)*, pages 1–10 (2016).

[170] T. Salimans, I. Goodfellow, W. Zaremba *et al.* Improved techniques for training gans. In *Advances in Neural Information Processing Systems 29*, pages 2234–2242 (2016).

[171] C. H. Bennett, E. Bernstein, G. Brassard *et al.* Strengths and weaknesses of quantum computing. *SIAM journal on Computing*, **26**, 1510 (1997).

[172] R. Schmied. Quantum state tomography of a single qubit: comparison of methods. *Journal of Modern Optics*, **63**, 1744 (2016).

[173] V. V. Shende, I. L. Markov, and S. S. Bullock. Minimal universal two-qubit controlled-not-based circuits. *Physical Review A*, **69**, 062321 (2004).

[174] K. M. Audenaert, J. Calsamiglia, R. Muñoz-Tapia *et al.* Discriminating states: The quantum Chernoff bound. *Physical review letters*, **98**, 160501 (2007).

[175] D. Zhu, N. M. Linke, M. Benedetti *et al.* Training of quantum circuits on a hybrid quantum computer. *Science Advances*, **5** (2019).

[176] A. Gretton, K. M. Borgwardt, M. Rasch *et al.* A kernel approach to comparing distributions. In *Proceedings of the 22nd National Conference on Artificial Intelligence - Volume 2*, AAAI'07, pages 1637–1641. AAAI Press, Vancouver, British Columbia, Canada (2007).

[177] K. E. Hamilton, E. F. Dumitrescu, and R. C. Pooser. Generative model benchmarks for superconducting qubits. *Physical Review A*, **99**, 062323 (2019).

[178] V. Leyton-Ortega, A. Perdomo-Ortiz, and O. Perdomo. Robust implementation of generative modeling with parametrized quantum circuits. *arXiv preprint arXiv:1901.08047* (2019).

[179] G. H. Low, T. J. Yoder, and I. L. Chuang. Quantum inference on bayesian networks. *Physical Review A*, **89**, 062315 (2014).

[180] J. Zeng, Y. Wu, J.-G. Liu *et al.* Learning and inference on generative adversarial quantum circuits. *Physical Review A*, **99**, 052306 (2019).

[181] H. Situ, Z. He, L. Li *et al.* Quantum generative adversarial network for generating discrete data. *arXiv preprint arXiv:1807.01235* (2018).

[182] J. Romero and A. Aspuru-Guzik. Variational quantum generators: Generative adversarial quantum machine learning for continuous distributions. *arXiv preprint arXiv:1901.00848* (2019).

[183] C. Zoufal, A. Lucchi, and S. Woerner. Quantum generative adversarial networks for learning and loading random distributions. *arXiv preprint arXiv:1904.00043* (2019).

[184] L. Hu, S.-H. Wu, W. Cai *et al.* Quantum generative adversarial learning in a superconducting quantum circuit. *Science Advances*, **5** (2019).

[185] M. Fingerhuth, T. Babej, and P. Wittek. Open source software in quantum computing. *PLOS ONE*, **13**, 1 (2018).

[186] S. Lloyd and S. L. Braunstein. Quantum computation over continuous variables. *Physical Review Letters*, **82**, 1784 (1999).

[187] H.-K. Lau, R. Pooser, G. Siopsis *et al.* Quantum machine learning over infinite dimensions. *Physical Review Letters*, **118**, 080501 (2017).

[188] S. Das, G. Siopsis, and C. Weedbrook. Continuous-variable quantum gaussian process regression and quantum singular value decomposition of nonsparse low-rank matrices. *Physical Review A*, **97**, 022315 (2018).

[189] M. Schuld, M. Fingerhuth, and F. Petruccione. Implementing a distance-based classifier with a quantum interference circuit. *EPL (Europhysics Letters)*, **119** (2017).

[190] V. Havlíček, A. D. Córcoles, K. Temme *et al.* Supervised learning with quantum-enhanced feature spaces. *Nature*, **567**, 209 (2019).

[191] D. G. Francesco Tacchino, Chiara Macchiavello and D. Bajoni. An artificial neuron implemented on an actual quantum processor. *npj Quantum Information*, **5**, 26 (2019).

[192] M. Benedetti, D. Garcia-Pintos, O. Perdomo *et al.* A generative modeling approach for benchmarking and training shallow quantum circuits. *npj Quantum Information*, **5** (2019).

[193] Y. Ding, L. Lamata, M. Sanz *et al.* Experimental implementation of a quantum autoencoder via quantum adders. *Advanced Quantum Technologies*, page 1800065 (2019).

[194] D. Ristè, M. P. da Silva, C. A. Ryan *et al.* Demonstration of quantum advantage in machine learning. *npj Quantum Information*, **3**, 16 (2017).

[195] J. C. Spall. A one-measurement form of simultaneous perturbation stochastic approximation. *Automatica*, **33**, 109 (1997).

[196] A. G. Baydin, B. A. Pearlmutter, A. A. Radul *et al.* Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, **18**, 1 (2018).

[197] M. Schuld, A. Bocharov, K. Svore *et al.* Circuit-centric quantum classifiers. *arXiv preprint arXiv:1804.00633* (2018).

[198] K. Mitarai and K. Fujii. Methodology for replacing indirect measurements with direct measurements. *Physical Review Research*, **1**, 013006 (2019).

[199] A. Harrow and J. Napp. Low-depth gradient measurements can improve convergence in variational hybrid quantum-classical algorithms. *arXiv preprint arXiv:1901.05374* (2019).

[200] S. Wisdom, T. Powers, J. R. Hershey *et al.* Full-capacity unitary recurrent neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 4887–4895. Curran Associates Inc., USA (2016).

[201] M. Ledoux. *The Concentration of Measure Phenomenon*. Mathematical surveys and monographs. American Mathematical Society (2001).

[202] E. Grant, L. Wossnig, M. Ostaszewski *et al.* An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *arXiv preprint arXiv:1903.05076* (2019).

[203] H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407 (1951).

[204] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[205] R. C. Eberhart and X. Hu. Human tremor analysis using particle swarm optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 3, pages 1927–1930. IEEE (1999).

[206] P. I. Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811* (2018).

[207] Y.-R. Liu, Y.-Q. Hu, H. Qian *et al.* Zoopt: Toolbox for derivative-free optimization. *arXiv preprint arXiv:1801.00329* (2017).

[208] K. Sastry, D. Goldberg, and G. Kendall. Genetic algorithms. In *Search methodologies*, pages 97–125. Springer (2005).

[209] L. Lamata, U. Alvarez-Rodriguez, J. Martín-Guerrero *et al.* Quantum autoencoders via quantum adders with genetic algorithms. *Quantum Science and Technology*, **4** (2018).

[210] M. Ostaszewski, E. Grant, and M. Benedetti. Quantum circuit structure learning. *arXiv preprint arXiv:1905.09692* (2019).

[211] K. M. Nakanishi, K. Fujii, and S. Todo. Sequential minimal optimization for quantum-classical hybrid algorithms. *arXiv preprint arXiv:1903.12166* (2019).

[212] H. Buhrman, R. Cleve, J. Watrous *et al.* Quantum fingerprinting. *Physical Review Letters*, **87**, 167902 (2001).

[213] L. Cincio, Y. Subaşı, A. T. Sornborger *et al.* Learning the quantum algorithm for state overlap. *New Journal of Physics*, **20**, 113022 (2018).

[214] S. Dasgupta and A. Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Structures & Algorithms*, **22**, 60 (2003).

[215] A. W. Harrow, A. Montanaro, and A. J. Short. Limitations on quantum dimensionality reduction. In *International Colloquium on Automata, Languages, and Programming*, pages 86–97. Springer (2011).

[216] A. Kay. Tutorial on the Quantikz Package. *arXiv preprint arXiv:1809.03842* (2018).

[217] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, **9**, 90 (2007).

[218] F. Pedregosa, G. Varoquaux, A. Gramfort *et al.* Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, **12**, 2825 (2011).

[219] O. Tange. GNU Parallel - The Command-Line Power Tool. *;login: The USENIX Magazine*, **36**, 42 (2011).