Quantum Enhancements for Deep Reinforcement Learning in Large Spaces

Sofiene Jerbi,[1,*] Lea M. Trenkwalder,[1] Hendrik Poulsen Nautrup,[1] Hans J. Briegel,[1,2]
and Vedran Dunjko[3]

[1]*Institute for Theoretical Physics, University of Innsbruck, Technikerstr. 21a, Innsbruck A-6020, Austria*
[2]*Department of Philosophy, University of Konstanz, Fach 17, Konstanz 78457, Germany*
[3]*LIACS and LION, Leiden University, Niels Bohrweg 1 Leiden, CA 2333, Netherlands*

Quantum algorithms have been successfully applied to provide computational speed ups to various machine-learning tasks and methods. A notable exception to this has been deep reinforcement learning (RL). Deep RL combines the power of deep neural networks with reinforcement learning, and has provided some of the most impressive recent artificial-intelligence (AI) results including the famous AlphaGo system—yet, no possibilities for quantum advantages have been identified to date. In this work, we show how quantum computers can enhance the performance of deep RL, especially where the action spaces are large. Specifically, we introduce so-called deep energy-based models, inspired by statistical physics, which we show outperform standard deep RL machinery in learning performance. These models are computationally more demanding, but this can be ameliorated by quantum techniques. Specifically, we provide quantum algorithms, some of which can be run on near-term quantum computers, that can be used to speed up deep energy-based RL. This result opens up a new playing field for quantum enhancements in machine learning and AI.

I. INTRODUCTION

Over the course of the last years, machine learning (ML) has emerged not just as a promising domain of application for quantum computing [1–5], but is often listed as one of its potential "killer applications" [6]. In the contexts of classification and big data problems, genuine quantum effects have been exploited to speed up computational bottlenecks stemming from either genuinely hard computational tasks [7,8] or from data volume and dimensionality [9–11]. More recently, new types of machine-learning algorithms for classification and generative modeling have been put forward to maximize the potential of noisy intermediate-scale quantum (NISQ) [12] devices. In this flavor of quantum ML, both quantum generalizations of classical methods [13], and genuinely quantum models [14] have also been proposed.

Beyond being a fruitful domain of application for existing quantum algorithms, the varied machine-learning settings have also stimulated the developments of new quantum algorithms [8,15,16].

However, not all areas of machine learning have been equally amenable to quantum enhancements, with the area of deep reinforcement learning (RL) being a critical exception. Deep RL combines the capacities of deep neural networks (NNs), with the interactive learning framework of reinforcement learning (see Fig. 1) and it underlies some of the most famous results in modern AI, such as the AlphaGo [17,18] system. It is becoming the go-to method in the hardest problems including automated driving [19], drug design [20], but also in machine-assisted research [21,22], quantum control [23–25], quantum sensing [26], quantum-algorithm design [27], and quantum error correction [28–30]. Nonetheless, despite its obvious importance, near-celebrity status, and numerous challenges which are still to be tackled [31], deep RL to date has not been successfully quantum enhanced.

In this work, we demonstrate quantum enhancements for a class of deep reinforcement learning, relying on ideas inspired by statistical physics and by combining them with known and new quantum algorithms. In more detail, we specify classes of neural-network-based RL algorithms, which achieve better learning performances than conventional methods when the state and action spaces of the learning algorithm are large. This is the case in many applications, e.g., with continuous degrees of freedom.

_____
*sofiene.jerbi@uibk.ac.at

We achieve this by utilizing computationally demanding underlying learning models, which, as we show, can enjoy significant quantum enhancements, in terms of speed ups and in terms of natural quantum generalizations. Our models are based on the ideas introduced by LeCun in Ref. [32], and the quantum algorithms involve accelerated sampling from hard distributions and variational circuit ideas.

Contributions. Our main results are threefold.

1. We introduce deep energy reinforcement learning, and demonstrate, using numerical experiments, its superior learning performance over conventional methods in large-action-space environments. Specifically, it attains near-optimal results, whereas the standard methods using models with similar parameter numbers fail.

2. We provide a general quantum-enhanced learning algorithm for deep energy-based RL, which utilizes quantum-sampling subroutines, and other quantum-algorithmic methods to speed up the learning process. For these purposes, we also provide the currently most efficient method for the quantum Gibbs sampling problem, which has further applications in, e.g., training quantum Boltzmann machines [13,33].

3. We specify quantum generalizations of classical energy-based models, which can be trained using our algorithm, and which can have further advantages over conventional methods by their capacity to capture more complex action-state distributions.

Our setting shares similarities with approaches previously proposed in generative machine learning, which utilize quantum algorithms to speed up sampling [34,35], or represent hard distributions using quantum circuits [36–39]. It thus also has the potential of establishing provable quantum-sampling separations [40,41]. However, in RL, one is not simply interested in representing or sampling from a *fixed* target distribution. Instead, agents should be able to represent an entire sequence of distributions (i.e., policies). These should generate useful interactions with the environment and slowly evolve to an optimal behavior, as we explain in more detail shortly. These specificities of the RL scenario make it a richer setting, which calls for new application-specific algorithms and has potential quantum advantages not present in bare generative modeling tasks.

The paper is organized as follows: Sec. II serves as an introduction to the well-known problem of large state and action spaces in reinforcement learning and the (in)adequacy of different methods to deal with this issue. Section III presents a step-by-step construction of the deep energy-based networks we consider. We evaluate their performance on benchmarking tasks and compare them to
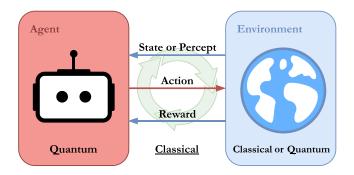

FIG. 1. Quantum-enhanced reinforcement learning. Agent and environment maintain a cyclic interaction consisting of *states* (or percepts) that the agent uses to deliberate on its *actions* according to its policy, and perceives feedback on its behavior in the form of *rewards*. In the quantum-enhanced framework, the agent is granted access to a, generally universal, quantum computer to enhance its deliberation process.

standard NN architectures used in deep RL in Sec. IV. Section V introduces a hybrid quantum-classical algorithm to train energy-based models for RL and describes the quantum subroutines it relies on. Finally, in Sec. VI, we discuss our results and present an outlook.

In Secs. II and III, we take a detailed didactic approach to the construction of our principal energy-based model: the deep energy-based network (DEBN). In this process, we draw interesting connections between the standard feedforward NNs used in deep RL [deep Q networks (DQNs)] and a relatively unexplored energy-based approach relying on restricted Boltzmann machines. The purpose of these sections is to better situate our model with respect to the fields it relates to and to setup a comprehensive framework for deep energy-based RL that allows for a straightforward generalization to quantum models. In Sec. V, we provide such generalizations. We refer the reader who may already be familiar with energy-based models and (deep) RL directly to Sec. III B for a short introduction to our DEBNs, before proceeding to the main contributions of our paper in Secs. IV and V.

II. BACKGROUND AND PRELIMINARIES

In this section, we introduce the well-studied case of RL environments with large state and action spaces and discuss modern approaches [42,43] that attempt to deal with such scenarios by contrasting them with traditional RL methods [44,45].

A. Tabular reinforcement learning methods

RL is a framework for the design of learning agents in interactive environments (see Fig. 1). We refer to Ref. [44] for a description of the RL formalism, which we do not require in full detail here. The goal of an agent in RL is to learn a policy $\pi(\mathbf{a}|\mathbf{s})$ that specifies its actions $\mathbf{a}$ given

perceived states **s** such that this policy maximizes its future rewards in a given task environment. In this work, we focus on two particular methods for the design of RL agents.

(a) Methods deriving from dynamic programming focus on a so-called *action-value function* $Q(\mathbf{s}, \mathbf{a})$ to link policy and expected rewards. These *value-based methods* (VBMs) [44] include, for instance, Q-learning and SARSA.

(b) Projective simulation (PS) [45] is a physics-inspired model for agency [46] that relies on weighted connections in a directed graph to reflect experienced rewards and derive a policy. These weights are referred to as h values. In its simplest (i.e., two-layered) form, the underlying weighted graph contains only directed connections from states to actions, resulting in state-action h values $h(\mathbf{s}, \mathbf{a})$.

To unify the treatment of these two approaches, we define the notion of *merit functions M*, of which both auxiliary functions Q and h are instances. These are real-valued functions defined on the state-action space that assign a certain value of merit to all actions in any given state. A standard procedure to learn merit functions and their associated policies is the so-called *tabular* method: scalar values $\hat{M}(\mathbf{s}, \mathbf{a})$ are stored for each state-action pair and are updated through interaction with the environment. This interaction is governed by the normalization of the merit values into a policy, e.g., by using a Gibbs-Boltzmann distribution, also called "softmax" distribution in ML:

$$\pi(\mathbf{a}|\mathbf{s}) = \frac{e^{\beta \hat{M}(\mathbf{s}, \mathbf{a})}}{\sum_{\mathbf{a}'} e^{\beta \hat{M}(\mathbf{s}, \mathbf{a}')}}, \quad (1)$$

where $\beta > 0$ is an inverse temperature hyperparameter. VBMs and PS each define an update rule that allows the values $\hat{M}(\mathbf{s}, \mathbf{a})$ associated with *experienced* state-action pairs to be iteratively updated such that rewarded actions become more likely to be taken at the next encounter with this state. For instance, in the case of Q-learning (a VBM):

$$\hat{Q}[\mathbf{s}^{(t)}, \mathbf{a}^{(t)}] \leftarrow (1 - \alpha)\hat{Q}[\mathbf{s}^{(t)}, \mathbf{a}^{(t)}]$$
$$+ \alpha\{r^{(t+1)} + \gamma \max_{\mathbf{a}} \hat{Q}[\mathbf{s}^{(t+1)}, \mathbf{a}]\}, \quad (2)$$

where $\mathbf{a}^{(t)}$ is the action performed by the agent in the state $\mathbf{s}^{(t)}$ at timestep t, which lead it to the state $\mathbf{s}^{(t+1)}$ and associated reward $r^{(t+1)}$; α is a learning rate taking value in $[0, 1]$ and used to deal with statistical fluctuations.

The fundamental problem with tabular approaches is their lack of generalization capabilities: there is no mechanism enabling an agent to generalize learned values to unobserved states and actions. Since merit-value updates act on single state-action pairs at a time, learning with tabular methods becomes intractable for environments with

large (potentially continuous) state and action spaces. However, this is a rather common aspect of practical RL tasks. Large and continuous state spaces appear, for instance, in vision problems, where camera images constitute the state space. Large action spaces are common in many control problems of large processes, including quantum control and quantum experiments [21,23,24,29, 30,47–54], where the continuous degrees of control are commonly discretized into actions to efficiently search for their optimal values. While generalization mechanisms do not overcome the intractable lower bound complexities of learning in large spaces [55], practical RL environments allow certain generalization heuristics to learn much faster, as we explain shortly.

B. Quantum approaches to reinforcement learning

To date, quantum enhancements for RL have mostly been considered in the context of tabular methods or when the environments allow genuine quantum interaction. A review of recent advances can be found in Ref. [56].

In the cases where quantum interaction with the environments is possible, quantum enhancements can be proven in terms of the sample complexity (i.e., interaction steps) needed to learn a good policy. For instance, quadratic speed ups in sample complexity have been shown using methods derived from Grover search [57] and phase estimation [58] [55,59–61]. Proof-of-concept environments for which exponential separations in sample complexity have also been showcased [62].

With respect to enhancements of the algorithmic part of the learning agent, speed ups have only been suggested for special models [63,64] and by building on the system of D-wave [65,66]. While the work of Ref. [63] provides a theoretical guarantee of quadratic speed up (as it relies on quantum-walk algorithms similar to the ones we use in Sec. V C), the works that rely on the D-wave system do not provide any guarantee of speed up.

In our work we construct a more general framework, centered around the generalization capacities of the learning model underlying reinforcement learning. For that, we focus on the most successful approach for generalization in RL, based on neural networks.

C. Generalization with neural networks

A common way to provide RL agents with generalization mechanisms is through *function approximation* methods, closely related to regression problems in supervised learning. Such approaches introduce a parametrized representation $M^{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a})$ of the merit function, replacing an otherwise sparsely filled merit table by a function that is defined on the entire state-action space. Updates of the merit function act no longer on individual values $\hat{M}(\mathbf{s}, \mathbf{a})$ but on a set of parameters $\boldsymbol{\theta}$, allowing one to
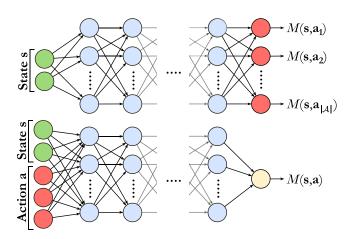
FIG. 2. Sketch of comparison between the layouts of DQN-like networks [42] and the DEBN we consider. The DQN-like model (top) estimates the merit function for all actions simultaneously while the DEBN (bottom) estimates the merit function for a single state-action pair. The first hidden layers may be convolutional.

indirectly update the values $M^{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a})$ of many—including unobserved—state-action pairs simultaneously. Using the terminology of supervised learning, $\{M^{\boldsymbol{\theta}}\}_{\boldsymbol{\theta}}$ constitutes the set of all possible hypothesis functions and learning consists in updating the parameters $\boldsymbol{\theta}$ as to find the setting that best approximates the true merit function M. Due to their success in supervised and unsupervised learning, neural networks, parametrized by their weights, have gained popularity as function approximators in RL. As we clarify next, while generalization over the state space is shared by all function approximation methods based on neural networks, generalization over the action space, i.e., learning a useful similarity measure between actions, is less common.

When using feedforward neural networks as function approximators, one can choose between two different approaches to represent the merit function (see Fig. 2). The first representation (i) takes as input a state $\mathbf{s}$ and outputs *all values* $\left[M^{\boldsymbol{\theta}}(\mathbf{s}, \boldsymbol{a_1}), \ldots, M^{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}_{|\mathcal{A}|})\right]$ associated to that state; while the second (ii) takes as input a single state-action pair $(\mathbf{s}, \mathbf{a})$ and outputs its corresponding merit value $M^{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a})$. Each of these representations has its advantages in different contexts, that we clarify below.

In their seminal work [42], Mnih *et al.* introduce a so-called DQN of type (i) that quickly became the standard in value-based deep RL [67]. DQNs spiked interest notably due to their success in vision problems, e.g., playing Atari games, as their so-called convolutional hidden layers made them perfectly suited to deal with the high-dimensional space of the images they receive as input. Importantly for our later analyses, utilizing a network of type (i) makes it particularly efficient to evaluate the Boltzmann policy of the agent [Eq. (1)], since it requires only a single evaluation

of the neural network followed by the application of a single additional softmax layer that normalizes all the merit values of its output.

However, this representation suffers from shortcomings when it comes to tasks with large action spaces, as we elaborate. During the first stages of learning, it is important for the agent to keep an explorative behavior to not get stuck in a possibly suboptimal policy. For this reason, the function approximator should be able to capture several of the local maxima of the merit function, which may originate from complex state-action correlations. In other words, the representation should be *multimodal*. However, it is reasonable to expect a neural network that can represent only merit values that are linear combinations of (nonlinear) state features to be unlikely to learn highly multimodal merit functions. This is why several works have chosen to avoid DQNs in environments with large action spaces [68,69].

We refer to representations of type (ii) as *deep energy-based networks* due to their interesting connection to so-called energy-based models, as explored in machine learning [32]. Inspired by statistical physics, energy-based models represent a target function using the energy function of a statistical system (that can be in one of combinatorially many configurations, each having a certain energy). When normalized, this energy function gives rise to a probability distribution over all possible configurations. An example of an energy-based model is the Boltzmann machine, introduced by Hinton and Sejnowski [70], which is inspired by the Ising model for spin systems [71]. Restricted Boltzmann machines (RBMs), i.e., Boltzmann machines with a further constrained set of energy functions, were the first energy-based models reportedly used in RL with the work of Sallans and Hinton [43], followed by various extensions [64,68,72–74]. Interestingly, although RBMs are commonly associated to a class of so-called stochastic recurrent neural networks, it turns out that they can be expressed as feedforward neural networks of type (ii) [75] with a single hidden layer. The DEBNs we consider here are then straightforwardly obtained by adding additional hidden layers to this feedforward neural network [76]. Energy-based models are also known to be good at representing multimodal distributions [68,77]. Hence, since DEBNs are able to encode deep and multimodal representations, they are well suited for tasks with both large state and action spaces.

A major drawback of DEBNs is the high cost of sampling from the policy (1): $|\mathcal{A}|$ evaluations of the network are needed to evaluate the policy exactly. *Approximate sampling* methods based on random walks, such as Gibbs sampling, can speed up the sampling process. However, Long *et al.* [78] showed that approximate sampling is still an NP-hard problem in the case of RBMs, an argument that can be trivially generalized to DEBNs.

In this work, we experimentally explore these intuitively motivated advantages of deep energy-based models in complex RL environments and show how quantum algorithms can help mitigate the computational bottleneck that they introduce.

III. DEEP ENERGY-BASED RL

In this section, we give a short introduction to energy-based models, which we build upon to design our deep energy-based learning agents. We first consider RBM function approximators for RL [43] and present an established connection between RBMs and shallow feedforward networks [75], which allows us to consider a natural extension to a deep architecture—our so-called DEBNs. We show that this function approximation method is compatible with both VBM and PS update rules. This construction thereby also provides an extension of PS to NN function approximation, which we call *deep PS*, in analogy to deep Q-learning.

A. Energy-based RL

1. Energy-based models

Energy-based models (EBMs) are a popular class of probabilistic models that find applications in both supervised and unsupervised learning [32]. They are often used as generative models, that is, to model the empirical probability distribution of observed data vectors $\{\mathbf{v}\}_{\mathbf{v}\in\mathcal{D}}$, e.g., strings of binary variables. EBMs assign a scalar energy value to each configuration of these variables, e.g., all possible binary strings of length $|\mathbf{v}|$, whose structure should reflect dependencies (i.e., correlations) between the variables. This energy function is usually parametrized by a vector $\boldsymbol{\theta}$ that determines the strengths of these dependencies. In the case of a Boltzmann machine, for instance, the energy function takes the form:

$$-E^{\boldsymbol{\theta}}(\mathbf{v}) = \sum_{i<j} w_{ij}\, v_i v_j + \sum_i b_i v_i, \qquad (3)$$

where $\boldsymbol{\theta} = \left[(w_{ij})_{1\leq i<j\leq|\mathbf{v}|},\ (b_i)_{1\leq i\leq|\mathbf{v}|} \right]$. Note that this particular Boltzmann machine is said to be fully visible, since its energy depends only on the visible variables v_i that encode the data (further explained shortly). The topology of the model specifies which variables are assumed (in)dependent. Combined, topology and parametrization constrain the family of available energy functions $\{E^{\boldsymbol{\theta}}\}_{\boldsymbol{\theta}}$ that an EBM can represent.

A probability distribution over the configuration space can be derived from the energy function by a certain normalization over all possible configurations (example below). Normalization assigns a high probability to configurations with low energy and vice versa. This generative probability distribution can be trained, i.e., iteratively

modified, to best fit an (empirical) probability distribution through updates of the parameters $\boldsymbol{\theta}$. These updates effectively select new energy functions from $\{E^{\boldsymbol{\theta}}\}_{\boldsymbol{\theta}}$, and, equivalently, new generative distributions.

In order to capture more complex dependencies between the variables, an additional set of auxiliary variables $\mathbf{h}$, called *hidden* or *latent* variables, can be added to the set of so-called visible variables $\mathbf{v}$. The model has now an energy $E^{\boldsymbol{\theta}}(\mathbf{v}, \mathbf{h})$ that includes terms characterized by the parameters between visible and hidden variables. The probability distribution over the visible variables is then specified by a function $F^{\boldsymbol{\theta}}(\mathbf{v})$ called the *free energy* of the model due to its connection to the equilibrium free energy in statistical physics. It is obtained by tracing out (or marginalizing) the hidden variables from the normalizing distribution. For instance, in the case of a softmax normalization, we have

$$P^{\boldsymbol{\theta}}(\mathbf{v}) = \frac{\sum_{\mathbf{h}} e^{-E^{\boldsymbol{\theta}}(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{v}',\mathbf{h}'} e^{-E^{\boldsymbol{\theta}}(\mathbf{v}',\mathbf{h}')}} = \frac{e^{-F^{\boldsymbol{\theta}}(\mathbf{v})}}{\sum_{\mathbf{v}'} e^{-F^{\boldsymbol{\theta}}(\mathbf{v}')}} \qquad (4)$$

and

$$F^{\boldsymbol{\theta}}(\mathbf{v}) = E^{\boldsymbol{\theta}}[\mathbf{v}, \langle\mathbf{h}\rangle_{P(\mathbf{h}|\mathbf{v})}] + H(\mathbf{h}|\mathbf{v}), \qquad (5)$$

where $\langle\mathbf{h}\rangle_{P(\mathbf{h}|\mathbf{v})}$ is the expectation value of the hidden variables under $P(\mathbf{h}|\mathbf{v})$ [defined similarly to distribution (4)] and $H(\mathbf{h}|\mathbf{v})$ is their Shannon conditional entropy under this same probability distribution. A derivation of Eq. (5) is provided in Appendix B.

2. Energy-based function approximation

Following the idea of Sallans and Hinton [43], we use the free energies of energy-based models as *parametrized* approximations of merit functions (such as the Q function in VBMs or the h values of PS), that is, we take $M^{\boldsymbol{\theta}}$ to be defined by the model's $-F^{\boldsymbol{\theta}}$. We start with a simple energy-based model: a restricted Boltzmann machine, further described in Appendix B. To represent a policy using a RBM, we divide its visible units into state and action units (see Fig. 3). This allows one to define a conditional probability distribution:

$$\pi^{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s}) = \frac{e^{-F^{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})}}{\sum_{\mathbf{a}'} e^{-F^{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a}')}} = \frac{e^{M^{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})}}{\sum_{\mathbf{a}'} e^{M^{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a}')}} \qquad (6)$$

effectively encoding the policy of an agent as prescribed by Eq. (1) (the inverse temperature β can usually be absorbed by the parameters $\boldsymbol{\theta}$).

This approximation of the merit function by the RBM free energy, as opposed to a tabular approach, has the advantage of being defined on the entire state-action space, which allows the agent to act nonrandomly on previously unseen states. Moreover, (implicitly) stored values $M^{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a})$ are not updated independently from one another,
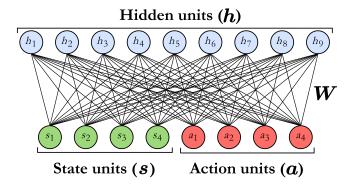
FIG. 3.   Restricted Boltzmann machine used for reinforcement learning. The visible units are divided between state and action units. Their connections to hidden units are weighted by a weight matrix W. Biases are not represented for compactness.

FIG. 4.   Feedforward neural network computing the exact free energy F of a RBM. W and b are the vectors of weights and biases parametrizing the RBM. The activations of the hidden units are softplus functions (see Appendix B). These activations and the input bias are summed up by the output unit.

as updates on $\boldsymbol{\theta}$ act on many values at a time, which allows one to generalize learned merit values to unobserved states and actions.

At first sight, this form of energy-based function approximation may look completely unrelated to an approach relying on feedforward neural networks, such as the DQN of Mnih *et al.*, making their structure hard to compare. Indeed, the expression of the RBM free energy given in Eq. (5) does not seem to be expressible as a neural-network function, i.e., linear combinations of (input) variables interlaid by nonlinear activation functions. But a quick derivation, given in Appendix B, shows that the free energy of a RBM can be evaluated *exactly* by a shallow feedforward neural network taking the form of Fig. 4. We come back to this interesting connection in the next subsection when we consider deep neural networks.

3. Updating the merit function

Updates on a RBM are typically performed through gradient descent applied on the weights $\boldsymbol{\theta}$ parametrizing its (free) energy. When used in an unsupervised learning scenario, the descent direction is given by the gradient of the Kullback-Leibler divergence (also known as relative entropy) between the RBM distribution $P^{\boldsymbol{\theta}}(\mathbf{v})$ and the empirical data distribution $\hat{p}(\mathbf{v})$ [79,80]. In a reinforcement learning scenario, however, we do not have direct access to (samples from) the optimal policy. Instead, the agent receives feedback from the environment in the form of delayed rewards. That is, feedback is given only after long sequences of actions, preventing the direct evaluation of any measure of distance to the optimal policy. For this reason, the update rule of the RBM used in RL is derived from the update rule of the merit function that the model approximates. To explain what we mean, we detail how this is done in the case of Q-learning, which has a tabular
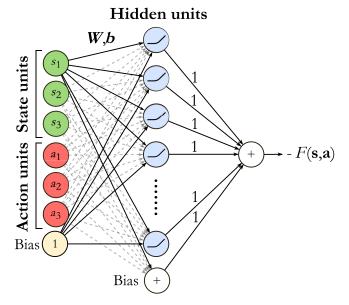
update rule [44] given by

$$-F^{\boldsymbol{\theta}}[\mathbf{s}^{(t)}, \mathbf{a}^{(t)}] \leftarrow -(1-\alpha)F^{\boldsymbol{\theta}}[\mathbf{s}^{(t)}, \mathbf{a}^{(t)}]$$
$$+ \alpha\{r^{(t+1)} - \gamma \max_{\mathbf{a}} F^{\boldsymbol{\theta}}[\mathbf{s}^{(t+1)}, \mathbf{a}]\}, \quad (7)$$

where $-F^{\boldsymbol{\theta}}$ corresponds to a Q-function approximation in this case and $\alpha \in [0, 1]$ is the so-called learning rate. Notice that this expression can be viewed of the form

$$\text{current} \leftarrow (1 - \alpha)\,\text{current} + \alpha\,\text{target}, \quad (8)$$

where the target value is an approximation of the optimal Q value (expressed in its bootstrapped expansion in terms of the current reward and the discounted Q value of the next timestep) that becomes exact upon convergence to the optimal policy.

Stated differently, this update aims at decreasing the so-called *temporal difference* error, i.e., the difference between target and current value:

$$\mathcal{E}^{\boldsymbol{\theta}}_{\text{TD}}(t) = r^{(t+1)} - \gamma \max_{\mathbf{a}} F^{\boldsymbol{\theta}}[\mathbf{s}^{(t+1)}, \mathbf{a}] + F^{\boldsymbol{\theta}}[\mathbf{s}^{(t)}, \mathbf{a}^{(t)}].$$

Analogously, we can derive similar expressions for SARSA:

$$\mathcal{E}^{\boldsymbol{\theta}}_{\text{TD}'}(t) = r^{(t+1)} - \gamma F^{\boldsymbol{\theta}}[\mathbf{s}^{(t+1)}, \mathbf{a}^{(t+1)}] + F^{\boldsymbol{\theta}}[\mathbf{s}^{(t)}, \mathbf{a}^{(t)}]$$

and PS (see Appendix B):

$$\mathcal{E}^{\boldsymbol{\theta}}_{\text{PS}}(t) = \tilde{r}^{(t+1)} + \gamma_{\text{PS}} F^{\boldsymbol{\theta}}[\mathbf{s}^{(t)}, \mathbf{a}^{(t)}]$$

It is then straightforward to derive an update rule for the weights of the RBM [$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \Delta\boldsymbol{\theta}$] by performing gradient descent on the squared error $\left[\mathcal{E}^{\boldsymbol{\theta}}(t)\right]^2$ [81]:

$$
\begin{aligned}
\Delta\boldsymbol{\theta} &= -\frac{\alpha}{2}\nabla_{\boldsymbol{\theta}}\left[\mathcal{E}^{\boldsymbol{\theta}}(t)\right]^2 \\
&= -\alpha\mathcal{E}^{\boldsymbol{\theta}}(t)\nabla_{\boldsymbol{\theta}}\mathcal{E}^{\boldsymbol{\theta}}(t) \\
&\approx -\alpha\mathcal{E}^{\boldsymbol{\theta}}(t)\nabla_{\boldsymbol{\theta}}F^{\boldsymbol{\theta}}[\mathbf{s}^{(t)}, \mathbf{a}^{(t)}].
\end{aligned}
\tag{9}
$$

It turns out that, in the case of RBMs, these update values can be computed efficiently since the gradient of their free energy $\nabla_{\boldsymbol{\theta}}F^{\boldsymbol{\theta}}(\mathbf{v})$ has a tractable expression:

$$
\begin{cases}
\frac{\partial F^{\boldsymbol{\theta}}(\mathbf{v})}{\partial w_{ik}} = -v_i\langle h_k\rangle_{P(\mathbf{h}|\mathbf{v})} \\
\frac{\partial F^{\boldsymbol{\theta}}(\mathbf{v})}{\partial b_i} = -v_i; \qquad \frac{\partial F^{\boldsymbol{\theta}}(\mathbf{v})}{\partial b_k} = -\langle h_k\rangle_{P(\mathbf{h}|\mathbf{v})},
\end{cases}
\tag{10}
$$

where $\langle h_k\rangle_{P(\mathbf{h}|\mathbf{v})} = 1/\left[1 + \exp\left(-\sum_i w_{ik}v_i - b_k\right)\right]$.

Hence, these formulas allow us to adapt RL update rules to train the free energies of RBMs as merit-function approximators.

B. From shallow to deep: the deep energy models for RL

The RBM-based model presented in the previous subsection fulfills one of the goals behind generalization in RL: having a parametrized representation of the merit function that allows us to generalize learned behavior to previously unobserved states and actions. A natural consideration that arises from this approximation method is a possible compromise in representational power: a tabular method can represent arbitrary real-valued functions over the state-action space (assuming it is finite), but there is no guarantee, *a priori*, that the RBM free energy can represent these functions with finitely many hidden units. Fortunately, a result from Le Roux and Bengio [82] shows that RBMs are universal function approximators: a RBM can represent arbitrary probability distributions over its visible units with any given precision, using a finite number of hidden units that depends on this distribution and the desired precision. Unfortunately, these universality results also suggest that certain policies may need a number of hidden units that is linear in the number of state-action pairs to be represented precisely [83]. Similarly, representing arbitrary merit functions using a RBM free energy (Fig. 4) has a comparable cost in the number of hidden units [75]. This leads to a neural network parametrized by a *very large* number of weights, rendering it difficult to train and likely to get stuck on suboptimal policies for general environments.

1. Deep energy-based network

In contrast to shallow neural networks, deep neural networks are strictly more expressive: the Vapnik-Chervonenkis dimension [84], a measure of function approximators' expressive power [85], of neural networks with ReLU activation functions (similar to the softplus function we consider here, see Appendix B) grows linearly with their number of hidden layers when keeping their total number of weights fixed [86]. This added expressive power can give some intuition on how come deep neural networks require only logarithmically many artificial neurons compared to their shallow counterparts to represent a certain general class of functions [87]. Given the greater representational power of deep neural networks, we come back to the connection between the RBM-based approach and feedforward neural networks made in the previous section. To enable the representation of a richer class of merit functions, we extend the free-energy network in Fig. 4 to a deep architecture by duplicating its hidden layer. The resulting neural network represents the free energy (i.e., the energy after tracing out the *stochastic* hidden units) of a so-called deep energy model (DEM) (see Fig. 5), first introduced by Ngiam *et al.* [76]. As pointed out by the authors, this model is different from both deep Boltzmann machines (DBMs) [88] and deep belief networks (DBNs) [89] due to its *directed deterministic* connections between its first layers, followed by a *single* undirected stochastic layer. Conceptually, this model transforms state-action pairs given as input into nonlinear features that are then treated as the visible units of a RBM. This construction preserves the tractability of the free energy inherited from the RBM (as opposed to DBMs and DBNs) while representing high-level abstractions of state-action pairs. Hence, DEBNs constitute models with great representational power without the associated drawback of having a large number of weights to train.

Same as for the RBM, updates of the merit function are performed through gradient descent on the temporal difference error [see Eq. (9)] with respect to the weights of the DEBN. However, due to the deep architecture of the neural network, we have to resort to backpropagation [90] to propagate derivatives of the error through the multiple layers of the DEBN. It can be easily shown that backpropagation of the temporal difference error on the shallow DEBN considered in the previous subsection leads to the same update rule as in Eqs. (9) and (10).

Prior to the DQN of Mnih *et al.* [42], it was widely believed that nonlinear function approximation using (deep) neural networks was not suitable for learning value functions due to its high instability, which is likely to lead to divergence from the optimal value function [91]. This instability originates from the approximation of the target value function [see Eq. (8)] that leads to a nonstationary temporal difference error and from the important
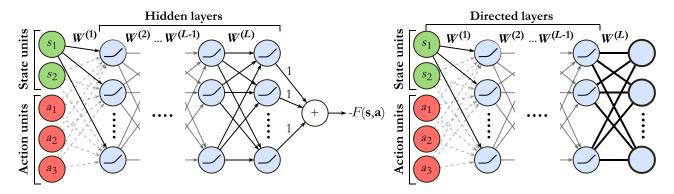
FIG. 5. The DEBN we consider (left) and its corresponding DEM (right). The DEM is obtained by removing the summation unit at the output of its corresponding DEBN and by replacing its last hidden layer with a layer of stochastic units of the same size. Just as in a RBM, these stochastic units can take binary values depending on the input given by their undirected connections. Biases are not represented for compactness.

correlations between the training samples generated by the agent. These two features, arising in RL yet not in supervised learning, both violate the assumptions behind convergence proofs in (un)supervised learning. Mainly two additions enabled stable learning with DQNs: (1) an *experience replay memory* storing (recent) samples of learning experience, used to generate decorrelated training data for the neural network; (2) a *target network*, i.e., a copy of the trained DQN updated at a slower rate, used to supply consistent targets for the temporal difference error during updates of the primary DQN. Naturally, we also rely on these tools to stabilize learning with our DEBNs.

A description of the full learning algorithm is presented in Algorithm 1 of Sec. V.

IV. PERFORMANCE COMPARISON BETWEEN VARIOUS MODELS

Having introduced how energy-based models can be used for function approximation, we now justify their use in RL. To that end, we study the differences in performance of both standard and energy-based function approximation methods for RL when facing environments with large state and action spaces. In a similar fashion to the analysis of Fu *et al.* [92], we establish several hypotheses regarding the performance of these models with respect to a number of learning aspects appearing in RL. These hypotheses are based on structural considerations along with previous experimental evidence. We then describe and carry out experiments to test our hypotheses.

Let us list our hypotheses, numbered (1) to (4) below.

We start by demonstrating that, similarly to a standard RL scenario based on feedforward NNs, the use of function approximation and especially deep models is also beneficial when considering an energy-based approach.

(1) We expect very simple (shallow) energy-based networks to quickly learn in environments with large

state spaces but simple target merit functions, while tabular methods would fail to learn as fast. This is due to the ability of (even shallow) neural networks to capture the simple correlations of the merit function, enabling it to generalize on the large input space.

(2) We expect shallow neural networks with a restricted number of hidden units to be unable to learn complex merit functions. A separation in learning performance between shallow and deep energy-based networks should then be noticeable when dealing with more complex task environments.

We test hypotheses (1) and (2) using common benchmarking tasks, introduced in Sec. IV A.

In the second part of this section, we investigate the advantage of energy-based function approximators when facing large action spaces. Our hypotheses stem from previous experimental results showing the improved exploration performance of energy-based policies when facing multimodal reward functions in RL [68,69], as well as results in generative modeling showing the ability of energy-based models to produce robust multimodal representations of large image datasets [77].

(3) We postulate that DEBNs are better suited than standard DQNs (see Sec. II C) to learn in RL environments with large action spaces: given the same number of parameters, we expect DEBNs to learn richer representations that allow for better generalization performance when learning merit functions. This should be due to the ability of the DEBN architecture to easily learn complex state-action correlations by having nonlinear hidden layers that act on both states and actions. Since in RL an agent has to rely on its own approximation of the merit function to explore the environment and update itself, we expect the RL learning setting to amplify
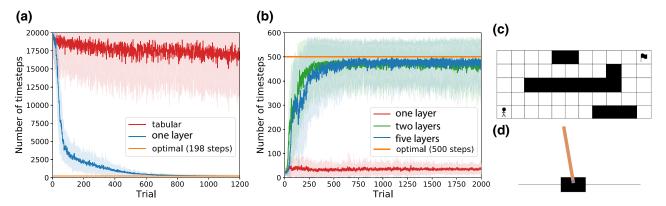
FIG. 6. Numerical evidence of the advantage of (deep) function approximation models in a 100×100 GridWorld task and the CartPole-v1 environment. (a),(b) In both plots, the optimal performance (198 and 500 steps, respectively) is indicated by the orange line. The remaining curves indicate the average performance and standard deviations of 50 agents trained using a PS update rule. (a) In the 100×100 GridWorld task, the performance of tabular agents (red curve) remains close to that of a random walk on the grid while agents with a shallow neural network (blue curve) reach close to optimal performance. (b) In the Cartpole-v1 environment, agents with a shallow neural network (red curve) exhibit close to random behavior while neural networks with at least two hidden layers achieve close to optimal performance. (c) Illustrative depiction of the GridWorld environment. Note that this image is purely illustrative as it does not reflect the actual size and layout of the grid used in the simulations. (d) A game-screen image of the CartPole-v1 environment.

poor generalization performance in DQN-type architectures. We isolate two learning aspects that are specific to a RL scenario and that we believe characterize these amplifying effects:

(a) *Policy sampling.* We refer to the first aspect as the policy-sampling property of the training process. It comes from the fact that an agent's policy influences the experienced data that constitutes its training set. Since this policy is derived from the agent's approximation of the merit function, a model that lacks to generalize properly its learned merit function will then render the training set suboptimal and hence limit the agent's learning performance.

(b) *Reward discounting.* The second aspect comes from the update rules of RL. As opposed to a supervised learning scenario, the target values used to train the merit-function approximators are not fixed but themselves computed using an approximation of the merit function. For instance, for SARSA:

$$\mathcal{E}^{\theta}_{\text{TD}'} = \left[r + \gamma M^{\theta}(\mathbf{s}', \mathbf{a}') \right] - M^{\theta}(\mathbf{s}, \mathbf{a}),$$

and for PS:

$$\mathcal{E}^{\theta}_{\text{PS}} = \widetilde{r} - \gamma_{\text{PS}} M^{\theta}(\mathbf{s}, \mathbf{a}).$$

The trained model has then to cope with a noisy and inaccurate target function [93]. These

properties are a consequence of statistical fluctuations of sampled trajectories and their associated rewards (main source of error in PS), as well as an imperfect approximation of the merit function associated to the current policy (main source of error in SARSA and Q-learning).

To test our hypothesis (3), we design two experiments that isolate each of these learning aspects. In each experiment, we test the generalization performance of DEBN and DQN agents for increasing dimensions of the action space, for which we expect to see a growing separation in performance between the two models. Our experiments have the specificity that the target merit function to be learned by the agents is fixed and clearly defined, which allows us to measure the error of the agents' approximations, that we refer to as generalization error. We also run a control experiment verifying that the learning aspect is indeed responsible for the observed separation.(4) Our last hypothesis posits that these gaps in generalization abilities manifest in a separation in learning performance for a sufficiently complex RL environment, e.g., with a high number of good suboptimal policies. We give evidence of this claim in such a RL environment where both effects highlighted above contribute to the learning performance.

The experiments that test hypotheses (1) and (2) are presented in Sec. IV A, while the experiments that test hypotheses (3) and (4) are presented in Sec. IV B. The Python code used to run these experiments is

openly available in a GitHub repository [94] and the hyperparameters used in these experiments are listed in Appendix F.

A. Shallow versus tabular and deep versus shallow

In this subsection, we numerically demonstrate the importance of function approximation and deep models in RL environments with large and continuous state spaces. The results are presented in Fig. 6.

1. GridWorld simulations

In order to demonstrate the advantage of shallow energy-based networks over tabular methods, we compare their performance in the GridWorld benchmarking task [95] [see Fig. 6(c)]. In this task, the agent navigates through a two-dimensional array of cells by choosing, at each timestep of interaction, one of the four main cardinal directions to move from one cell to the next. The training is divided into trials. At the beginning of each trial, the position of the agent is initialized at a fixed starting cell $(0, 0)$ and a reward is placed in a fixed goal cell (n, n), where n is the grid size. A trial terminates if the agent reaches the goal cell or if the length of the trial exceeds $20\,000$ timesteps. In the former case, the agent receives a reward of 1. In the latter, the agent receives a reward of -1. For the simulations presented here, we choose a 100×100 grid with closed boundary conditions and no obstacles such that the state space is made large while maintaining a simple optimal policy, close to being state independent (going up or right with equal probability, except at the boundaries). This last property brings a very helpful but yet very simple way to generalize the learned merit function and policy on the large state space. In Fig. 6(a), we demonstrate that agents using a shallow neural-network architecture significantly outperform their tabular counterparts in this task. This improved performance attests the generalization capabilities provided by the neural network. The tabular methods are unable to detect and exploit the similarities between different cells.

2. CartPole simulations

We now show, by means of an example, that DEBNs may provide advantages over shallow architectures in more complex task environments. More specifically, we consider the OpenAI Gym CartPole-v1 environment [96], which features a continuous state space. In CartPole, the goal is to balance an inverted pole pendulum mounted on a movable cart by applying to the latter, at each timestep of interaction, a unit of force to the left or right [see Fig. 6(d)]. A trial terminates in one of three situations: (a) the angle between the vertical line above the pivot point and the pole exceeds $12°$, (b) the center of mass of the cart is at a horizontal position outside the range $[-2.4, 2.4]$,

(c) the pendulum is successfully balanced for 500 consecutive timesteps. For every timestep the agent manages to balance the pole, it receives a reward of 1. The state space of this task can be described by four continuous parameters: the position of the cart, the velocity of the cart, the angle between the vertical line above the pivot point and the pole, and the angular velocity of the pole. In Fig. 6(b), we compare the performance of architectures with one, two, and five hidden layers but the same total number of weights. We observe that the network with a single hidden layer does not manage to learn the complex optimal policy of this task. However, already two hidden layers are sufficient to achieve near-optimal performance, demonstrating the importance of *deep* energy-based models in RL tasks.

B. DEBN VERSUS DQN

After considering RL environments with large state spaces, we now focus on environments with large action spaces, for which we expect an advantage of the DEBN architecture over DQN-like networks (see Fig. 2). In order to keep a fair comparison between the two models, we constrain our neural networks in two ways. First, we set their total number of parameters to be equal whenever their performance is compared. Moreover, for these models to be practically relevant, we restrict their total number of parameters to a *reasonable* bound (constant in our experiments, but in practice simply significantly smaller than the state-action size). However, these constraints combined lead to a particular regime where the separation between the two architectures is straightforward: as the size of the action space grows, the restricted parameters of the DQN tend to accumulate in its output layer (which is linear in the number of actions) until the model becomes unusable, while the DEBN architecture is significantly less affected (since its input layer can grow as slowly as logarithmically with the action space). Since this *bottleneck effect* would cause DEBNs to trivially outperform DQNs, we choose our bounds on the number of parameters such that our instances of DQNs stay out of the bottleneck regime.

Let us now introduce in more detail the RL-specific aspects that we expect to cause a separation between the two architectures and the experiments we perform to test this claim.

1. Policy sampling

The first aspect comes from the policy-sampling property of the training process: at each interaction step, an agent samples one action $\mathbf{a}$ from a policy specified by its current approximation $M^\theta(\mathbf{s}, \ldots)$ of the merit function, and updates this approximation according to a loss that is (in general) independent of the values $M^\theta(\mathbf{s}, \mathbf{a}'), M(\mathbf{s}, \mathbf{a}')$ for
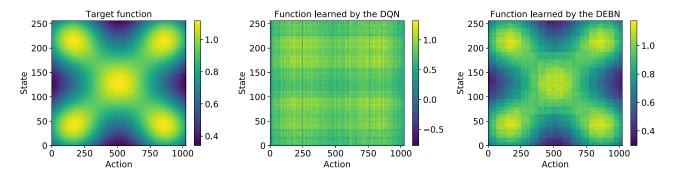
FIG. 7. The policy-sampling experiment. At each learning step, a state is uniformly sampled from the state space (of size 256). Its corresponding binary encoding is perceived by the agent, which samples an action from a Boltzmann policy computed using its current approximation of the merit function (initially random). In the policy-sampling experiment, the target value associated to this sampled state-action pair is compared to the approximation of the neural network, resulting in the loss used to train its weights. In the analog multidimensional regression problem, the entire row of target values associated to that state is compared to the values approximated by the network, each contributing to the loss. We additionally use mini batches of states of size 10 to compute average losses and improve convergence rates. The DEBNs also receive binary encodings of the actions.

different actions $\mathbf{a}'$. As opposed to a supervised (multidimensional) regression task, where, for each sampled state $\mathbf{s}$, the agent could compute its loss on the entire vector $\left[M^\theta(\mathbf{s}, \mathbf{a}_1), \ldots, M^\theta(\mathbf{s}, \mathbf{a}_{|\mathcal{A}|}) \right]$, the policy-sampling aspect can lead to *masking* effects on sampled target values. In particular, when the current approximation incorrectly assigns large values to certain actions, this approximation error skews the sampling distribution in an inopportune way and prevents learning the merit function faithfully on the entire state-action space. Since sampling from the correct data distribution is already an imperative to get good generalization error in regression tasks, it should as well be important in the more general setting of policy sampling. We expect these masking effects to be particularly relevant when the target merit function is multimodal, as this multimodality (in general) adds complexity to the state-action correlations to be learned. Intuitively, since DEBNs take some (compact) representation of individual actions as input, they are able to construct nonlinear features of both states and actions. Hence, this makes DEBNs more likely to learn complex state-action correlations and masks, in a way that is not possible for a network that can only construct state features in its hidden layers. To investigate the magnitude of this effect, we compare the learning performance of these two models in the policy-sampling learning task described in Fig. 7. Interestingly, we observe that the DQNs learn a very noisy approximation of the target function, and even completely fail to learn one of the five modes of the target function. In order to give more quantitative results, we test the generalization performance of the two models on the same task with different levels of discretization of the action space. As we can see in Fig. 8(a), the separation in testing loss appears to increase linearly with the size of the action space. Here, one could argue that this separation is solely

caused by the bottleneck effect, as previously discussed. However, we show that this separation indeed originates from the policy-sampling aspect of training by running a control experiment on the analog supervised learning task, where we indeed observe that both architectures have the same performance up to 256 actions (after which the bottleneck effect might be responsible for part of the separation).

2. Discounted rewards

The second aspect of RL for which we expect a separation comes from the reward discounting introduced by RL update rules. For both projective simulation and value-based methods, merit values do not correspond to the immediate rewards of state-action pairs but also include estimates of future rewards discounted by a discount (or glow) factor [see bootstrapped expression of Eq. (8)]. This is an intrinsic property of RL update rules as they all aim at optimizing long-term rewards. For VBMs, these estimates of future rewards are given by a sample of the immediate reward and the merit value of the next experienced state as approximated by the model; while, for PS, these correspond to the rewards collected during one sampled trajectory in the environment. For *stochastic* policies and/or environments, the sampled terms lead in general to statistical fluctuations in the target merit values on which the agents compute their loss. As for the approximated terms, errors in the approximation of the merit function lead to noise in the target values. We refer to both these effects as fluctuations of the merit values. To study the influence of this RL aspect in isolation from other sources of fluctuations, we consider the following task: an agent is given access to sample rewards experienced by a fixed stochastic policy (also called *off* policy) in a toy environment
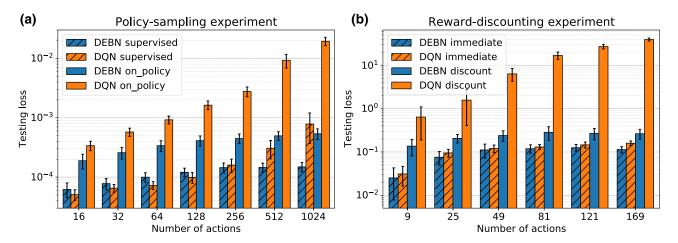
FIG. 8.   Separation in testing loss between DEBN and DQN agents in the policy-sampling and reward-discounting experiments. (a),(b) In both plots, each bar indicates the average testing loss (i.e., generalization error) of 20 agents in their respective tasks. The number of weights of both neural networks is kept constant and equal in each experiment. Hatched bars are associated to the control experiments where the learning setting does not have the tested property. (a) In the policy-sampling experiment, we test the ability of both architectures to faithfully represent a multimodal merit function (depicted in Fig. 7) when training samples are generated by the current approximation of said function. The testing loss corresponds to the average smoothed l_1 loss over all merit values associated to 1000 randomly sampled states after 5000 learning steps. (b) In the reward-discounting experiment, we test the ability of both architectures to cope with the statistical fluctuation on their target merit values brought by a SARSA update rule. The testing loss corresponds to the average l_2 loss over all merit values associated to 100 randomly sampled states after 2000 learning steps (20 000 total sampled transitions and an update period of 10).

where states are randomly sampled at each interaction step, and needs to learn its *discounted* merit function. The constraints of this task effectively cancel the policy-sampling aspect considered in the previous experiment as well as temporal fluctuations induced by the stochastic policy at hand. The setting for which we observe the highest separation between the two models is for an off policy very close to optimal (performing a rewarded action with probability 0.99 and a random action otherwise) and with a discount factor of 0.9. Again, to assess the influence of reward discounting on this separation, we run a control experiment where the discount factor is set to 0. The results of these simulations are presented in Fig. 8(b). Note that we observe this separation despite the use of the learning mechanisms that help counter the statistical fluctuation of target values: notably, the experience replay memory and, more importantly, the target network (see Sec. III B).

3. A complete RL environment

In the previous experiments, we study the separations between DEBNs and DQNs in terms of their generalization error when learning merit functions for restricted RL environments. We now give evidence of a separation in learning performance (i.e., average collected rewards) in a complete RL environment. Here, agents learn to solve a toy GridWorld task with a special transition function by directly interacting with it (policy sampling) and are subject to the statistical fluctuations induced by a discount

factor. This task environment has the same description in terms of states and actions as the one used for the discounted-reward experiment, but differs in its transition function such that it adds more structure to the task. The description goes as follows: an agent is situated in a one-dimensional GridWorld with N cells, enumerated 0 to $N - 1$, circular boundary conditions and a goal state at position $\lfloor N/2 \rfloor$. At each interaction step, the agent is at a given position n and can do one of $(\lfloor N/2 \rfloor + 1)^2$ actions. Each action is described by two subactions: moving i steps to the left and j steps to the right for i, j in $0, \ldots, \lfloor N/2 \rfloor$. Call n' the position $n + j - i \bmod \lfloor N/2 \rfloor$. From the actions for which n' corresponds to the goal position $\lfloor N/2 \rfloor$, only the shortest (i.e., smallest $i + j$) is rewarded with a reward of 1. All other actions are not rewarded. Now for the transition function: if $n' = \lfloor N/2 \rfloor$, the agent is moved to the position $n + 1$ (or $n + 2$ if that happens to be the goal position), otherwise it is moved to the position n'. This choice of transition function, while quite artificial, leads to the interesting property that this task allows for several suboptimal policies, e.g., the agent runs into a loop of $l \in \{2, \ldots, N - 2\}$ positions with only one nonrewarded transition, achieving an average reward of $1 - 1/l \geq 0.5$. The only optimal policy on the other hand, runs through a loop of length $N - 1$ and achieves an average reward of 1. This makes it a very challenging task if one seeks the optimal policy, and in part explains why the separation in performance between the two architectures appears small. We do however observe in the results of our experiment
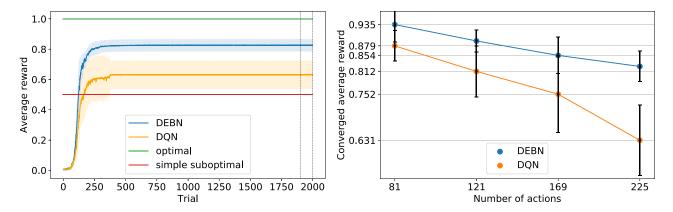
FIG. 9. Separation in average rewards between DEBN and DQN agents in a RL task. Each dot on the right plot indicates the average reward of 20 agents over 100 trials after training for 1900 trials of length 100 interactions. The left plot shows the evolution of this average performance over the course of training for the toy circular-GridWorld task with 225 actions. Very simple suboptimal policies can achieve an average reward of 0.5, represented by the red line. The optimal policy achieves an average reward of 1, indicated by the green line.

(see Fig. 9) an increasing separation in average rewards with respect to the number of actions in the environment. This illustrates how the gaps in generalization performance observed for the RL-specific learning aspects discussed above translate into a separation in learning performance when these aspects act in concert.

4. Comments on the experiments

Due to the simplicity of the chosen tasks, the separations observed in our experiments with DEBNs and DQNs may get less significant in different learning regimes. Similarly to tabular methods being able to learn optimal policies given arbitrarily long learning time, we expect the observed separations to get smaller when looking at longer learning times, or, for instance, when increasing the total number of weights of the networks and hyperparameters such as the update period of the target networks. However, throughout our exploration of relevant regimes, we did not observe any inversion of the performance of both networks, indicating that a large separation might as well be present in broader regimes for more complex environments. We use a PS update rule for the simulations on the benchmarking tasks and a SARSA update rule for the separation results in the reward-discounting and full-RL experiment. We however believe that the same experiments with interchanged update rules exhibit a similar behavior, possibly with different separations. This claim is supported by an extensive numerical study that showed that the tabular versions of PS and VBMs behave similarly in these types of benchmarking tasks [95] and additional numerical simulations that were not included in this paper for conciseness. In order not to penalize the DEBN agents with approximation errors in sampling, we use *exact* sampling (see next section) from the policy of the agents in the experiments above. Note that the quantum speed up presented in the

next section applies to problems with large action spaces and hence can not be numerically demonstrated as classical simulations are intractable.

V. HYBRID DEEP ENERGY-BASED RL WITH QUANTUM ENHANCEMENTS

So far, we have shown that one instance of energy-based models, the DEBNs, can provide a learning advantage over standard DQNs in complex RL environments. However, this learning advantage comes at a cost: the inefficiency of sampling from energy-based policies, needed to act on the environment and train the RL agent. It is easy to see that *exact* sampling is intractable, since it requires computing the merit values $M^\theta(\mathbf{s}, \mathbf{a})$ for all possible actions $\mathbf{a}$ given a state $\mathbf{s}$ to evaluate the expression of Eq. (1), which costs $\mathcal{O}(|\mathcal{A}|)$ feedforward evaluations in the case of DEBNs. Moreover, this hardness of sampling even extends to *approximate* sampling: it is actually an NP-hard problem to generate samples from a distribution that is *provably* close in total variation distance from a general RBM distribution [78]. We address this computational bottleneck by applying quantum algorithms for approximate sampling from Gibbs distributions, that is algorithms that can offer quantum speed ups to *heuristic* sampling methods. We investigate the potential of using both (long-term) fault-tolerant algorithms with provable quadratic speed ups [97–99], as well as near-term approximate optimization approaches [100–102].

We have focused in our simulations on DEBNs rather than other energy-based models due to their interesting properties: they offer deep NN-based representations of the merit function while allowing efficient computation of this function and its gradient (through a forward and backward evaluation of the DEBN). A drawback of DEBNs is that, in order to be evaluated in quantum algorithms,

---

Algorithm 1 Hybrid deep energy-based RL

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize main and target merit function approximators $M^{\boldsymbol{\theta}}$, $\widetilde{M}^{\boldsymbol{\theta}'}$ with the same random parameters
Initialize update rule R in {PS, Q-learning, SARSA}
for episode $= 1, \ldots, E$ **do**
    Initialize state s_1
    for $t = 1, \ldots, T$ **do**
        Sample an action $\boldsymbol{a_t}$ with approximate probability $\pi(\boldsymbol{a_t}|\boldsymbol{s_t}) = e^{\beta M^{\boldsymbol{\theta}}(\boldsymbol{s_t},\boldsymbol{a_t})}/Z_\beta(\boldsymbol{s_t})$ using **Theorem 1**
        Execute action $\boldsymbol{a_t}$ on the environment and observe reward r_t and state $\boldsymbol{s_{t+1}}$
        if R $\in$ {SARSA, Q-learning} **then**
            Store transition $(\boldsymbol{s_t}, \boldsymbol{a_t}, r_t, \boldsymbol{s_{t+1}})$ in $\mathcal{D}$
        end if
        Sample random minibatch $\mathcal{B}$ of transitions $(\boldsymbol{s_j}, \boldsymbol{a_j}, r_j, \boldsymbol{s_{j+1}})$ from $\mathcal{D}$
        for transition $\in \mathcal{B}$ **do**
            Evaluate $M^{\boldsymbol{\theta}}(\boldsymbol{s_j}, \boldsymbol{a_j})$ (and $\widetilde{M}^{\boldsymbol{\theta}'}(\boldsymbol{s_j}, \boldsymbol{a_j})$ for PS) using **Theorem 2**
            if R $\in$ {SARSA, Q-learning} **then**
                Set $\beta' \gg 1$ for Q-learning, otherwise $\beta' = \beta$
                Sample an action a' with approximate probability $\pi(\boldsymbol{a'}|\boldsymbol{s_{j+1}}) = e^{\beta'\widetilde{M}^{\boldsymbol{\theta}'}(\boldsymbol{s_{j+1}},\boldsymbol{a'})}/Z_{\beta'}(\boldsymbol{s_{j+1}})$ using **Theorem 1**
                Evaluate $\widetilde{M}^{\boldsymbol{\theta}'}(\boldsymbol{s_{j+1}}, \boldsymbol{a'})$ using **Theorem 2**
            end if
            Set $y_j = \begin{cases} r_j + \widetilde{M}^{\boldsymbol{\theta}'}(\boldsymbol{s_j}, \boldsymbol{a_j}) - \gamma_{\text{ps}} M^{\boldsymbol{\theta}}(\boldsymbol{s_j}, \boldsymbol{a_j}) & \text{if } R = \text{PS} \\ r_j + \gamma \widetilde{M}^{\boldsymbol{\theta}'}(\boldsymbol{s_{j+1}}, \boldsymbol{a'}) & \text{if } R = \text{SARSA} \\ r_j + \gamma \max_{a'} \widetilde{M}^{\boldsymbol{\theta}'}(\boldsymbol{s_{j+1}}, \boldsymbol{a'}) & \text{if } R = \text{Q-learning} \end{cases}$
            Evaluate $\nabla_{\boldsymbol{\theta}} M^{\boldsymbol{\theta}}(\boldsymbol{s_j}, \boldsymbol{a_j})$ **Theorem 3**
            Accumulate $\Delta\boldsymbol{\theta} += \alpha \left(y_j - M^{\boldsymbol{\theta}}(\boldsymbol{s_j}, \boldsymbol{a_j})\right) \nabla_{\boldsymbol{\theta}} M^{\boldsymbol{\theta}}(\boldsymbol{s_j}, \boldsymbol{a_j})$
        end for
        Perform a gradient descent step $\boldsymbol{\theta} + \Delta\boldsymbol{\theta}$ and reset $\Delta\boldsymbol{\theta} = 0$
        Every C steps, set $\widetilde{M}^{\boldsymbol{\theta}'} = M^{\boldsymbol{\theta}}$
    end for
    if R = PS **then**
        Compute discounted rewards $\widetilde{r_t}$ using the edge-glow mechanism [103]
        Store experienced transitions $(\boldsymbol{s_t}, \boldsymbol{a_t}, \widetilde{r_t})$ in $\mathcal{D}$
    end if
end for

FIG. 10. Pseudocode of the algorithm used to train energy-based agents in RL environments. Possible quantum enhancements are highlighted by Theorems 1, 2, and 3.

one needs to carry out the feedforward computation coherently, which is out of reach of near-term quantum devices. Hence, this encourages us to additionally consider different deep energy-based models such as deep Boltzmann machines as well as their quantum extensions (QBMs) that have the advantage of being easily simulatable. In view of their similar properties, we expect these models to be as robust to the learning aspects that hinder DQN-type models. With DBMs and QBMs however, we face the issue that the evaluation of the merit function (i.e., the negative free energy of the BM after tracing out its hidden units), along with its gradient, is intractable for a large number of hidden layers and units. The reason behind this intractability is similar to the one behind the hardness of sampling from RBMs or DEBN policies: in both cases, one needs to compute the free energy of the model after tracing out mutually dependent (i.e., connected) units, which is a hard task. However, similarly to the task of sampling from RBMs and DEBN policies, approximate sampling algorithms can be used to estimate the free energy of DBMs and its gradient.

In this section, we introduce a hybrid quantum-classical scheme for training deep energy-based models (Algorithm 1 in Fig. 10). We list several classical and quantum models that can be used in this hybrid scheme, and present quantum subroutines to speed up sampling and the evaluation of the approximate merit function and its gradient.

A. Description of the algorithm

In Algorithm 1, we give a general procedure to train RL agents with energy-based merit-function approximators. This procedure applies to both VBM and PS update rules and takes into account common mechanisms to stabilize learning, notably a replay memory and a target network (see Sec. III B). These mechanisms, along with the interaction with the environment, are kept classical. Throughout training though, the agent needs to perform three subroutines that are amenable to quantum speed ups:

1. sampling from a distribution specified by the current approximation of the merit function;

TABLE I.   Applicability of different sampling algorithms to several energy-based models. $\|F^\theta\|$ corresponds to the maximum free energy of the model, while $\|H^\theta\|$ is its maximum clamped energy, i.e., the energy after tracing out the hidden units or simply fixing the input units, respectively. K counts the number of hidden units of the model while A counts its number of action units. β is the inverse temperature of the sampled Gibbs distribution and δ the spectral gap of the Gibbs sampling Markov chain using for sampling (when applicable). The complexities of exact sampling come from basic linear algebra techniques, while the complexities of classical Gibbs sampling are due to lower bound results on mixing times of Markov chains [103] (see also Appendix A). The complexities for the quantum algorithms are proven in Theorem 1.

Sampling →		Classical		Near/Mid-term quantum		Fault-tolerant quantum	
Model ↓		Exact sampling	Gibbs sampling	Quantum annealers	Variational Gibbs state preparation	Quantum simulated annealing	Hamiltonian simulation
Classical	DEBN	$\mathcal{O}(2^A)$	$\widetilde{\mathcal{O}}\left(\frac{1}{\delta}\right)$	✗	✗	$\widetilde{\mathcal{O}}\left(\sqrt{\frac{\beta\|F^\theta\|}{\delta}}\right)$	$\widetilde{\mathcal{O}}\left(\sqrt{2^A}\beta\|F^\theta\|\right)$
	DBM	$\mathcal{O}(2^{AK})$	$\widetilde{\mathcal{O}}\left(\frac{1}{\delta}\right)$	(heuristic)	(heuristic)	$\widetilde{\mathcal{O}}\left(\sqrt{\frac{\beta\|H^\theta\|}{\delta}}\right)$	$\widetilde{\mathcal{O}}\left(\sqrt{2^{AK}}\beta\|H^\theta\|\right)$
Quantum	QBM	$\mathcal{O}(2^{3AK})$	✗	✗	(heuristic)	$\widetilde{\mathcal{O}}\left(\frac{\beta^2\|H^\theta\|^2}{\sqrt{\delta}}\right)$	$\widetilde{\mathcal{O}}\left(\sqrt{2^{AK}}\beta\|H^\theta\|\right)$

2. estimating the approximate merit values associated to experienced states and sampled actions;
3. estimating the gradient of the approximate merit function for experienced state-action pairs.

So far, we have been considering DEBNs, for which subroutines 2 and 3 can be efficiently performed classically and hence do not require quantum speed ups. For different energy-based models however, this can no longer be the case as they generally involve computing expectations values with respect to intractable probability distributions. Nonetheless, we show in Sec. V D that these two subroutines can be performed using the same sampling algorithms of subroutine 1, presented in Sec. V C. To see this, note that the free energy of an energy-based model takes the form

$$F(\mathbf{v}) = \langle E(\mathbf{v}, \mathbf{h}) \rangle_{P(\mathbf{h}|\mathbf{v})} + S(\mathbf{h}|\mathbf{v}),$$

that is the sum of the expected energy and Shannon entropy (or von Neumann entropy in the quantum case) of the model under the conditional Gibbs distribution $P(\mathbf{h}|\mathbf{v})$. Both these terms can be estimated by sampling from this same distribution. As for the gradient of the free energy, we show that for the models we consider, these gradients take the form of expected values of the model parameters under the distribution $P(\mathbf{h}|\mathbf{v})$.

Essentially, at the core of these three subroutines is a sampling problem. As explained in more detail in Sec. V C, classical algorithms that solve this problem should generate samples from a distribution that is close to a target Gibbs distribution of a classical or quantum Hamiltonian, while, in the quantum case, the analog goal is to prepare quantum Gibbs states that encode such distributions. We focus on the complexities of such algorithms to quantify the runtime of our subroutines.

B. Examples of suitable energy-based models

Before discussing our quantum subroutines, we start by describing the energy-based models studied in this section. Having already introduced in Sec. III restricted Boltzmann machines, deep energy models and their associated free-energy DEBNs, we now focus on DBMs and QBMs. As is common in statistical and quantum physics, we switch from energy functions to a description in terms of Hamiltonians, involving here Pauli σ^x and σ^z matrices. Similarly to Sec. III, we define our merit-function approximators $M^\theta(\mathbf{s}, \mathbf{a})$ using the negative free energy $-F^\theta(\mathbf{s}, \mathbf{a})$, i.e., the energy after tracing out hidden units of these models.

1. Deep Boltzmann machines

DBMs [88,104] are generalizations of RBMs with additional hidden layers. They are described by a diagonal Hamiltonian of the form:

$$H^\theta_{\text{DBM}} = -\sum_{i,k_1} w_{i,k_1}\sigma_i^z\sigma_{k_1}^z - \sum_{j,k_1} w_{j,k_1}\sigma_j^z\sigma_{k_1}^z$$
$$- \sum_{\eta,k_\eta,k_{\eta+1}} w_{k_\eta,k_{\eta+1}}\sigma_{k_\eta}^z\sigma_{k_{\eta+1}}^z - \sum_l b_l\sigma_l^z \quad (11)$$

parametrized by a real-valued vector $\boldsymbol{\theta} = (w_{i,k_1}, w_{j,k_1}, w_{k_\eta,k_{\eta+1}}, b_l)_{i,j,k_\eta,l}$ and where we adopt the notation that assigns the indices i, j, l to state, action and all units, respectively, and the indices k_η to hidden units of the ηth hidden layer. Note that general Boltzmann machines with unrestricted connections (UBM) can also be mapped to Hamiltonians of the form H^θ_{DBM}. One way of doing this is by simulating the intralayer connections of a UBM using new connections to additional hidden units (that can be placed in the next hidden layer) [105].

The DBM free energy however does not have a neural-network formulation, and is instead expressed as

$$
\begin{aligned}
F_{\text{DBM}}^{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}) &= -\log\left[\sum_{\mathbf{h}} e^{-H^{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a},\mathbf{h})}\right] \\
&= -\sum_{i,k_1} w_{i,k_1} s_i \langle h_{k_1}\rangle - \sum_{j,k_1} w_{j,k_1} a_j \langle h_{k_1}\rangle \\
&\quad - \sum_{\eta,k_\eta,k_{\eta+1}} w_{k_\eta,k_{\eta+1}} \langle h_{k_\eta} h_{k_{\eta+1}}\rangle - \sum_k b_k \langle h_k\rangle \\
&\quad - \sum_i b_i s_i - \sum_j b_j a_j \\
&\quad + \sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{s},\mathbf{a})\log[P(\mathbf{h}|\mathbf{s},\mathbf{a})], \quad (12)
\end{aligned}
$$

where $\langle h_k\rangle$ is the expected value of the hidden unit k under the conditional distribution $P(\mathbf{h}|\mathbf{s},\mathbf{a})$.

2. Semitransverse quantum Boltzmann machines

QBMs [13,33] are a generalization of Boltzmann machines with additional transverse-field terms σ^x in their Hamiltonians that allow for eigenvectors that are superpositions of computational basis states. The resulting nondiagonal Hamiltonians are of the form:

$$
\begin{aligned}
H_{\text{QBM}}^{\boldsymbol{\theta}} &= -\sum_{i,k_1} w_{i,k_1} \sigma_i^z \sigma_{k_1}^z - \sum_{j,k_1} w_{j,k_1} \sigma_j^z \sigma_{k_1}^z \\
&\quad - \sum_{\eta,k_\eta,k_{\eta+1}} w_{k_\eta,k_{\eta+1}} \sigma_{k_\eta}^z \sigma_{k_{\eta+1}}^z - \sum_l b_l \sigma_l^z \\
&\quad - \sum_k \Gamma_k \sigma_k^x \quad (13)
\end{aligned}
$$

parametrized by a real-valued vector $\boldsymbol{\theta} = (w_{i,k_1}, w_{j,k_1}, w_{k_\eta,k_{\eta+1}}, b_l, \Gamma_k)_{i,j,k_\eta,l,k}$ and where we adopt the notation that assigns the indices k to all hidden units. We choose to apply only the transverse-field terms $\Gamma_k \sigma_k^x$ on hidden units (hence the qualifier *semitransverse*). This choice allows expression of the Hamiltonian in the form $H_{\text{QBM}}^{\boldsymbol{\theta}} = \sum_{\mathbf{v},\mathbf{h_v}} \lambda_{\mathbf{v},\mathbf{h_v}} |\mathbf{v}, \mathbf{h_v}\rangle \langle \mathbf{v}, \mathbf{h_v}|$, where $|\mathbf{v}\rangle = |\mathbf{s}, \mathbf{a}\rangle$ is a given configuration of the state and action units (i.e., a computational basis state) but $|\mathbf{h_v}\rangle$ is allowed to be an arbitrary superposition of computational basis states (see Appendix B). In turn, this leads to a straightforward derivation of $\Lambda_{\mathbf{v}} H_{\text{QBM}}^{\boldsymbol{\theta}} \Lambda_{\mathbf{v}} = \sum_{\mathbf{h_v}} \lambda_{\mathbf{v},\mathbf{h_v}} |\mathbf{v}, \mathbf{h_v}\rangle \langle \mathbf{v}, \mathbf{h_v}|$ for $\Lambda_{\mathbf{v}} = \Lambda_{\mathbf{s},\mathbf{a}} = |\mathbf{s}, \mathbf{a}\rangle \langle \mathbf{s}, \mathbf{a}| \otimes I_{\mathbf{h}}$ a projector on a given configuration $\mathbf{v} = (\mathbf{s}, \mathbf{a})$, which helps evaluate the QBM's free energy and its gradient via sampling (see Sec. V D). These considerations are similar to the ones of Wiebe *et al.* [35] (albeit visible units are the ones allowed to be in superposition in their case), who used QBMs for generative training of quantum states.

This free energy again does not take a neural-network formulation and has the following expression:

$$
F_{\text{QBM}}^{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}) = \text{Tr}[\rho_{\mathbf{s},\mathbf{a}} H_{\text{QBM}}^{\boldsymbol{\theta}}] + \text{Tr}[\rho_{\mathbf{s},\mathbf{a}}\log(\rho_{\mathbf{s},\mathbf{a}})], \quad (14)
$$

where $\rho_{\mathbf{s},\mathbf{a}} = \Lambda_{\mathbf{s},\mathbf{a}} e^{-H_{\text{QBM}}^{\boldsymbol{\theta}}} \Lambda_{\mathbf{s},\mathbf{a}} / \text{Tr}\left[\Lambda_{\mathbf{s},\mathbf{a}} e^{-H_{\text{QBM}}^{\boldsymbol{\theta}}}\right]$ is the QBM Gibbs state after a projective measurement $\Lambda_{\mathbf{s},\mathbf{a}}$.

3. Access models

In the case of RBMs and DEMs, we take advantage of the computability of their free energies to avoid using their full Hamiltonian, and consider instead the Hamiltonians defined by their associated DEBNs.

All the Hamiltonians considered here allow sparse access [106], meaning that we can efficiently (i.e., in polynomial time) construct the two unitary oracles:

$$
\begin{aligned}
\hat{O}_H |j\rangle |k\rangle |z\rangle &= |j\rangle |k\rangle |z \oplus H_{jk}\rangle, \\
\hat{O}_F |j\rangle |l\rangle &= |j\rangle |f(j,l)\rangle.
\end{aligned}
$$

For a d-sparse Hamiltonian H, $\hat{O}_H$ accepts a row index j and column index k and returns H_{jk} in a binary format, while $\hat{O}_F$ accepts a row index j and a number $l \in [d]$ and computes in-place the column index $f(j,l)$ of the lth nonzero element in row j.

While sparse access is sufficient to run all the quantum subroutines described below, its implementation requires resources far beyond those NISQ devices can offer. This is the case of DEBN (and DEM) Hamiltonians for instance, where the implementation of $\hat{O}_H$ requires the computation of the entire neural-network function coherently, which is out of reach of near-term implementations. DBMs and QBMs however also naturally satisfy access in the form of linear combinations of (local) unitaries $H = \sum_{n=1}^N \alpha_n U_n$, where N scales at most quadratically with the number of units of the model, which eases up significantly their implementation.

C. Quantum algorithms for Gibbs state preparation

At the crux of training energy-based models is the ability to sample from Gibbs distributions of the form

$$
p_\beta(\mathbf{x}) = \frac{e^{-\beta H(\mathbf{x})}}{\text{Tr}_{\mathbf{x}}[e^{-\beta H}]} \quad (15)
$$

of a Hamiltonian H specifying our learning model, where we assume $\{\mathbf{x}\}$ to be a subset of the eigenvectors of H, for simplicity [107]. While the energy $H(\mathbf{x})$ of a single configuration can be efficiently computed, the partition function $\text{Tr}_{\mathbf{x}}[e^{-\beta H}] = \sum_{\mathbf{x}} e^{-\beta H(\mathbf{x})}$ involves the energies of combinatorially many configurations, making it hard to approximate in general [78].

Classically, this distribution can be computed explicitly by evaluating the energy of all the configurations $\mathbf{x}$ we want to sample over and normalizing the resulting vector of energies. This is the approach we take in the numerical simulations of Sec. IV. Alternatively, one can use random walk algorithms such as Gibbs sampling to generate samples from an approximation of this distribution (for completeness, we describe these random walk algorithms in Appendix A).

In a quantum setting, one is interested in the similar goal of preparing approximately a state of the form [108]

$$|\psi_\beta\rangle = \frac{1}{\sqrt{\text{Tr}_\mathbf{x}[e^{-\beta H}]}} \sum_\mathbf{x} \sqrt{e^{-\beta H(\mathbf{x})}} |\mathbf{x}\rangle, \qquad (16)$$

called a purified Gibbs state of our model Hamiltonian. This is because measuring this state in the basis $\{\mathbf{x}\}_\mathbf{x}$ effectively returns a sample from the Gibbs distribution, Eq. (15). Interestingly, the exact and Gibbs sampling approaches mentioned above both have their quantum analog, each providing a quadratic speed up in sampling over the classical algorithms (up to overheads in some parameters). However, these quantum algorithms utilize many gates and qubits and will likely require fault-tolerant implementations to be used. To consider approaches that may be more amenable to NISQ constraints, we also study a variational approach to Gibbs state preparation that trades off guarantees of speed up and convergence with less resource-consuming primitives.

We summarize the complexities of these sampling algorithms in the following informal theorem, before providing their short descriptions.

Theorem 1 (Purified Gibbs state preparation (informal)). *Given sparse access to one of the model Hamiltonians listed above and a bound $\|M^\theta\|$ on its free energy or clamped energy* (i.e., *the energy after tracing out the hidden units or simply fixing the input units,* respectively), *we can prepare a purified Gibbs state at inverse temperature β in either*

$$T_{\text{Gibbs}} = \widetilde{\mathcal{O}}\left[\sqrt{\frac{\beta \|M^\theta\|}{\delta}}\right]$$

quantum-walk steps, where δ is a lower bound on the spectral gaps of the Markov chains associated to this quantum walk; or in

$$T'_{\text{Gibbs}} = \widetilde{\mathcal{O}}\left(\sqrt{2^n} \beta \|M^\theta\|\right)$$

gates and queries to the Hamiltonian, where n is the number of action (and hidden) *units of the model.*

Alternatively, *for DBMs and* QBMs, *one can opt for a heuristic variational approach, where the cost of evaluating the objective function is in*

$$\widetilde{\mathcal{O}}\left[\|M^\theta\|/\varepsilon + 1/(p_{\min}^2 \beta \varepsilon)\right]$$

gates and preparations of the parametrized quantum circuit, where ε is the precision of the estimate and $p_{\min}$ an approximation parameter.

A recurring quantity in the subroutines of Theorem 1 is a bound on the model Hamiltonian's norm $\|M^\theta\|$, that one needs to estimate to be able to apply them. In order to do so, one has several possibilities:

(a) Use $\|M^\theta\| = \|\boldsymbol{\theta}\|_1$, the l_1 norm of all the weights that parametrize a chosen model, as this quantity also bounds the Hamiltonian's maximum eigenvalue.

(b) Use the maximum finding algorithm to get an approximate value [99,109] of $\|M^\theta\|$ [even though this algorithm has a complexity in $\widetilde{\mathcal{O}}\left(\sqrt{2^n}\right)$ for $n = A$ or AK].

(c) Assume that the approximated merit function is close to the true function and use a bound on the latter, e.g., $\frac{|R|_{\max}}{1-\gamma}$ in the case of VBMs, where $|R_{\max}|$ is the largest obtainable reward and γ the environment's discount factor.

The proof of Theorem 1 is a combination of known results along with an additional lemma derived from previous works. In the following, we summarize these results, namely fault-tolerant quantum algorithms based on Hamiltonian simulation and quantum simulated annealing to prepare purified Gibbs states, followed by variational approaches to Gibbs state preparation suitable for near-term quantum devices.

1. Gibbs state preparation using simulation

The approach of van Apeldoorn *et al.* [99] consists in using Hamiltonian simulation techniques [110] to implement an approximation of the operation $f(H) = e^{-H}$. Since this operation is not unitary, one needs to rely on so-called block encodings to apply it, that is, construct a unitary $\tilde{U}$ such that $\|(\langle 0| \otimes I)\tilde{U}(|0\rangle \otimes I) - e^{-H}\| \leq \varepsilon$. This construction has a complexity in $\widetilde{O}(\|H\|d)$ for a d-sparse Hamiltonian. One can then apply $\tilde{U}$ on the A register of the maximally entangled state $\sqrt{2^{-n}} \sum_i |i\rangle_A |i\rangle_B$ and use $\widetilde{\mathcal{O}}\left(\sqrt{2^n/\text{Tr}[e^{-H}]}\right)$ rounds of amplitude amplification [111] to amplify the relevant $(|0\rangle \otimes I)$ subspace. This results in a good approximation of the desired purified Gibbs state. To get rid of the $1/\text{Tr}[e^{-H}]$ term in the complexity, the authors additionally apply a linear transformation on H, that we defer, along with a detailed

restatement of their result, to Appendix C. Note that, since this approach is based on Hamiltonian simulation techniques, it is applicable to both classical and quantum (i.e., nondiagonal in the computational basis) Hamiltonians without the need for special considerations.

2. Quantum simulated annealing (QSA)

In the case of quantum simulated annealing, classical and quantum Hamiltonian require a separate treatment, that we describe below. We start by an overview of the classical-Hamiltonian case and then explain how its methods can be generalized to the quantum-Hamiltonian case.

Classical Hamiltonians. The idea behind the QSA approach is to quantize Gibbs sampling random walks to achieve a quadratic speed up in runtime. Random walk algorithms have a runtime complexity that depends on the spectral gap δ of their associated Markov chains (MCs). This runtime, also called mixing time of the MC, is in $\tilde{\mathcal{O}}(\delta^{-1})$ (see Appendix A for more details). An example of such MCs is specified by the Gibbs sampling (or more generally, Metropolis-Hastings) algorithm. Pictorially, these MCs define the transition probabilities of a random walk over the eigenstates of the Hamiltonian that is governed by their respective eigenvalues, such that, after the MC mixing time, eigenstates are sampled according to their Gibbs distribution. A quantum analog of classical random walks are Szegedy-type quantum walks [112,113]. They generalize a MC P to a unitary $W(P)$ called a Szegedy walk operator: by taking P to be a Gibbs sampling MC whose stationary distribution is the Gibbs distribution p_β of Eq. (15), $W(P)$ has the quantum state $|\psi_\beta\rangle$ of Eq. (16) as its eigenvalue-1 eigenvector. Using a combination of phase estimation [58] and amplitude amplification [111], one can transform an initial state $|\phi\rangle$ into the desired state $|\psi_\beta\rangle$ using a total of $\tilde{\mathcal{O}}[(|\langle\psi_\beta|\phi\rangle|\sqrt{\delta})^{-1}]$ applications of $W(P)$ [114]. Direct preparation of $|\psi_\beta\rangle$ from an arbitrary state $|\phi\rangle$, e.g., the uniform superposition $|u\rangle = \sqrt{2^{-n}}\sum_i|i\rangle$, can however be intractable since the overlap $|\langle u|\pi_\beta\rangle|^2$ can be as small as 2^{-n} (or even worse for a different $|\phi\rangle$) [115]. To get around this issue, one can instead use these same tools to prepare several intermediary states $|u\rangle = |\psi_{\beta_0}\rangle, |\psi_{\beta_1}\rangle, \ldots, |\psi_{\beta_l}\rangle$ that follow an *annealing schedule* on the inverse temperature β of the Gibbs distribution. Intuitively, if the overlap between these intermediate states is made large by construction, then the whole transformation should be efficient. Formalizing this idea, Wocjan *et al.* have shown [114] that given a schedule $\beta_0 = 0, \beta_1, \ldots, \beta_l = \beta$ and Gibbs sampling Markov chains $P_{\beta_0}, P_{\beta_1}, \ldots, P_{\beta_l}$ with stationary distributions $p_{\beta_0}, p_{\beta_1}, \ldots, p_{\beta_l}$ that fulfill the slow-varying condition $|\langle\psi_{\beta_i}|\psi_{\beta_{i+1}}\rangle|^2 \geq c$, preparing an ε approximation of $|\psi_\beta\rangle$ has complexity $\mathcal{O}[(l/c)\sqrt{\delta^{-1}}\log^2(l/\varepsilon)]$. Using an *adaptive* annealing schedule, Harrow and Wei [97] were able to provide a construction with length $l =$

$\tilde{\mathcal{O}}\{\sqrt{\log(\text{Tr}[e^{-\beta H}]^{-1})}\} \leq \tilde{\mathcal{O}}(\sqrt{\beta\,\|M^\theta\|})$, leading to a total preparation complexity in $\tilde{\mathcal{O}}(\sqrt{\delta^{-1}\beta\,\|M^\theta\|})$.

Quantum Hamiltonians. In the case of a quantum Hamiltonian (e.g., for QBMs), while we can still construct sparse access to this Hamiltonian, its eigenstates are not computational basis states in general but instead superpositions that are *a priori* unknown. Since random walk algorithms need access to the eigenstates of a problem Hamiltonian in order to navigate its energy landscape (and eventually sample from its associated Gibbs distribution), this prevents us from constructing the necessary Gibbs sampling Markov chains and their associated quantum-walk operators [116]. A way to circumvent this issue was introduced by Temme *et al.* [117], then further generalized to work with quantum-walk operators and QSA by Yung and Aspuru-Guzik [98]. While these approaches use different constructions, both rely at their core on Hamiltonian simulation and quantum phase estimation to get access to the eigenvalues of a quantum Hamiltonian without knowledge of the structure of its eigenvectors. Because we are interested here in quantum-walk approaches, we focus here on the method of Yung and Aspuru-Guzik. This method, as presented, does not use *adaptive* annealing schedules and relies on projective measurements instead of quantum amplitude amplification to adiabatically transform intermediary states, analogously to the approach of Somma *et al.* [118]. For these reasons, the method has a complexity in $\tilde{\mathcal{O}}\left(\beta^2\|M^\theta\|^2/\sqrt{\delta}\right)$, quartically worse in β and $\|M^\theta\|$ than with diagonal Hamiltonians. We prove however, in the following lemma, that the advances in QSA for classical Hamiltonians presented above are also applicable to quantum Hamiltonians, leading to the same complexities.

Lemma 1. *Given sparse access to a quantum Hamiltonian H and a bound $\|M^\theta\|$ on its free energy or clamped energy, we can prepare via adaptive quantum simulated annealing a quantum state ε close to the coherent encoding of its Gibbs distribution at inverse temperature β*

$$\frac{1}{\sqrt{\text{Tr}[e^{-\beta H}]}}\sum_{\varphi}\sqrt{e^{-\beta\langle\varphi|H|\varphi\rangle}}\,|\varphi\rangle\,|\tilde{\varphi}\rangle, \qquad (17)$$

where $|\varphi\rangle$ are eigenstates of H and $|\tilde{\varphi}\rangle$ their complex conjugates; using

$$\tilde{\mathcal{O}}\left(\sqrt{\frac{\beta\|M^\theta\|}{\delta}}\right)$$

quantum-walk steps, where δ is a lower bound on the spectral gaps of the Markov chains associated to this quantum walk.

Proof. The proof of this lemma goes in two steps, each extending the construction of Yung and Aspuru-Guzik [98] to the algorithmic improvements of Wocjan *et al.* [114] and Harrow and Wei [97] respectively:

Step 1. We first note that the generalized Szegedy walk operators defined by Yung and Aspuru-Guzik for quantum Hamiltonians satisfy the relevant properties that are required for the QSA approach of Wocjan *et al.* (Theorem 2 in Ref. [114]) that is based on amplitude amplification instead of projective measurements. These properties are namely having a purified Gibbs state of the form Eq. (17) as eigenvalue-1 eigenstate and a phase gap $\Delta \geq 2\sqrt{\delta}$. Note that, as shown by Yung and Aspuru-Guzik, the eigenbasis of the quantum Hamiltonian is not needed to construct the generalized walk operators and input states of these QSA algorithms. Additionally, by working in the eigenbasis of the quantum Hamiltonian instead of the computational basis, the error analysis in Lemma 3 and Corollary 3 of Ref. [114] to derive the length and overlaps of the annealing sequence is straightforwardly applicable to quantum Hamiltonians, leading to a preparation complexity in $\widetilde{\mathcal{O}}\left(\beta \|M^{\boldsymbol{\theta}}\|/\sqrt{\delta}\right)$.

Step 2. By tracing the construction steps of Harrow and Wei (Theorem 4 of Ref. [97]), we can see that the proof of existence of an annealing (or cooling) schedule, which ensures the slow-varying condition (i.e., large overlaps between intermediary states) in their adiabatic preparation is independent of the Hamiltonian eigenbasis. Hence the proof can be directly expressed in the eigenbasis of quantum Hamiltonians without any other change. Moreover, the tools they use for the adaptive construction of these schedules, namely nondestructive amplitude estimation for binary search of the next inverse temperature β' (Theorems 9 and 10 of Ref. [97]), are also applicable to generalized Szegedy walk operators. Hence, by applying the same methods, we obtain an overall preparation complexity in $\widetilde{\mathcal{O}}\left(\sqrt{\beta \|M^{\boldsymbol{\theta}}\|/\delta}\right)$. ∎

3. Variational Gibbs state preparation

Recent advances in the field of quantum computation have motivated the development of algorithmic approaches that are compatible with near- and midterm devices. One particularly promising approach for the preparation of Gibbs states is based on a variational optimization method. The general idea is to train a parametrized Ansatz $|\psi_{\boldsymbol{\theta}}\rangle = U(\boldsymbol{\theta})|\mathbf{0}\rangle$ such that it approximates the state of Eq. (16), or equivalently, that its density matrix $\rho = |\psi_{\boldsymbol{\theta}}\rangle\langle\psi_{\boldsymbol{\theta}}|$ approximates the Gibbs state $e^{-\beta H}/\text{Tr}[e^{-\beta H}]$. Since this target Gibbs state is the unique state that minimizes the free energy

$$F = \text{Tr}[\rho H] + \frac{1}{\beta}\text{Tr}[\rho \log(\rho)] \qquad (18)$$

with respect to the Hamiltonian H, one can use this free energy as an objective function to be minimized by $|\psi_{\boldsymbol{\theta}}\rangle$. The existing methods in variational Gibbs state preparation [100–102] vary in three aspects.

(a) The choice of Ansatz: Chowdhury *et al.* [100] use an Ansatz derived from Trotterized adiabatic state preparation, while Wu and Hsieh [101] opt for an Ansatz motivated by the quantum approximate optimization algorithm (QAOA), and Wang *et al.* [102] choose a so-called hardware-efficient Ansatz.

(b) The subroutines to evaluate the Ansatz' free energy [Eq. (18)]: the objective function is composed of the average energy of ρ with respect to H and the von Neumann entropy of ρ. The latter is the more term challenging to evaluate. While Wu and Hsieh consider a decomposition into several Renyi entropies, measured using swap tests [119]; Wang *et al.* and Chowdhury *et al.* use a truncation of its Taylor series, where the terms are computed via (high-order) swap tests and amplitude estimation [58], respectively.

(c) The optimization method: here one can opt for a gradient-descent approach, where several evaluations of the objective function and finite-difference formulas can be used to estimate its gradient; or else use gradient-free methods such as Nelder-Mead or Powell's method [120].

The approaches for free-energy evaluation are especially useful for us since they can directly be used for the merit-function evaluation subroutine of Theorem 2. We choose to use the approach of Chowdhury *et al.*, that we restate in Appendix C for completeness. Note however that, depending on the RL application, a different approach might be more suitable, as different approximation methods correspond to different effective models. Further investigations should reveal which methods are more interesting in a RL context.

D. Quantum algorithms for merit function and gradient evaluation

As mentioned above, the neural-network formulation of the RBM and DEM free energies (as DEBNs) allows for efficient exact computation of the modeled merit functions and their gradients. As for DBMs and QBMs, it is easy to see from the formulation of their free energies [Eqs. (12) and (14)] that these should be estimated via sampling from their Gibbs distributions. More specifically, these are the Gibbs distributions of the *clamped* Hamiltonians, where state and action units have been fixed to a given configuration.

Regarding the gradients of these free energies however, the argument becomes slightly more intricate in

the case of QBMs. As derived in Appendix C, derivatives of the QBM free energy take the form $\partial_\theta F(\mathbf{s}, \mathbf{a}) = -\text{Tr}[\Lambda_{\mathbf{s},\mathbf{a}} \partial_\theta e^{-H_{\text{QBM}}}]/\text{Tr}[\Lambda_{\mathbf{s},\mathbf{a}} e^{-H_{\text{QBM}}}]$. Due to the nondiagonal terms in H_{QBM}, we have that H_{QBM} and $\partial_\theta H_{\text{QBM}}$ do not commute, making it impossible to use the identity $\partial_\theta e^{-H} = -\partial_\theta H e^{-H}$ that would apply for instance to DBM Hamiltonians. Nonetheless, due to our restriction to *semitransverse* QBMs and by the use of Duhamel's formula [13], we can show that the gradient of the QBM free energy takes the form of expectation values $\text{Tr}[(\partial_\theta H_{\text{QBM}})\rho_{\mathbf{s},\mathbf{a}}]$ where $\partial_\theta H_{\text{QBM}}$ are simply (products of) Pauli σ^z and σ^x operators appearing in H_{QBM}.

Hence, merit-function and gradient evaluation boil down to Gibbs state preparation again, but with fixed state and action units $\mathbf{s}, \mathbf{a}$. We formalize the subroutines used for these evaluations in the following theorems.

Theorem 2 ((DBM and QBM merit-function evaluation)). *Let $H_{\text{DBM}-\text{QBM}}^{\boldsymbol{\theta}}$ be a Hamiltonian describing a deep Boltzmann machine as specified by Eq.* (11) *or a semitransverse quantum Boltzmann machine as specified by Eq.* (13) *and* ($\mathbf{s}, \mathbf{a}$) *a given assignment of the state and action units. The approximate free energy* $F_{\text{DBM}-\text{QBM}}^{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}) = -M^{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a})$ *associated to this state-action pair can be estimated to precision ε in*

$$\widetilde{\mathcal{O}}\left[T_{\text{Gibbs}} \cdot \left(\frac{\|\boldsymbol{\theta}\|_1}{\varepsilon} + \frac{1}{p_{\min}^2 \varepsilon}\right)\right],$$

where T_{Gibbs} is the complexity of preparing the Gibbs state of the clamped Hamiltonian, $\|\boldsymbol{\theta}\|_1 = \sum_{i,k} |w_{i,k}| s_i + \sum_{j,k} |w_{j,k}| a_j + \sum_{k,k'} |w_{k,k'}|$ $(+\sum_k |\Gamma_k|)$ and $p_{\min}$ is the cutoff used in the estimation of the entropy term of the free energy.

Proof. Using one of the subroutines in Theorem 1, we can prepare a purified Gibbs state of the clamped model Hamiltonian at inverse temperature $\beta = 1$ in time T_{Gibbs}. We then use the subroutines of Theorems 6 and 7 (Appendix C) to evaluate the average energy and von Neumann entropy of this state using $\widetilde{\mathcal{O}}(\|\boldsymbol{\theta}\|_1/\varepsilon)$ and $\widetilde{\mathcal{O}}\left[1/(p_{\min}^2 \varepsilon)\right]$ calls to the Gibbs state preparation unitary, respectively. Note that the success probability of these subroutines can be amplified to $1 - \delta$ using $\log(\delta^{-1})$ repetitions and taking the median of the outcomes, according to the powering lemma [121]. ∎

Theorem 3 (Gradient evaluation of a DBM and QBM merit function). *Let $H_{\text{DBM}-\text{QBM}}^{\boldsymbol{\theta}}$ be a Hamiltonian describing a deep Boltzmann machine as specified by Eq.* (11) *or a semitransverse quantum Boltzmann machine as specified by Eq.* (13) *and* ($\mathbf{s}, \mathbf{a}$) *a given assignment of the state and action units. The gradient of the free energy* $\nabla_{\boldsymbol{\theta}} F_{\text{DBM}-\text{QBM}}^{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}) = -\nabla_{\boldsymbol{\theta}} M^{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a})$ *associated to*

this state-action pair can be estimated to precision ε in

$$\widetilde{\mathcal{O}}\left\{T_{\text{Gibbs}} \cdot \min\left[\frac{K + K_L^2(L-1)}{\varepsilon}, \frac{1}{\varepsilon^2}\right]\right\},$$

where T_{Gibbs} is the complexity of preparing the Gibbs state of the clamped Hamiltonian, K is the number of hidden units in the model and K_L is the number of hidden units in each of the L layers of the model.

Proof. We count the number of components of the gradient to of the order $\mathcal{O}\left[K + K_L^2(L-1)\right]$. Using one of the subroutines in Theorem 1, we can prepare a purified Gibbs state of the clamped model Hamiltonian at inverse temperature $\beta = 1$ in time T_{Gibbs}. From there, we can take two approaches.

(a) Measuring the hidden units of the Gibbs state in the computational basis gives us a vector $\mathbf{h}$ sampled from the distribution $P(\mathbf{h}|\mathbf{s}, \mathbf{a})$. For the QBM, we also need to estimate (in a separate round of measurements) the expected values of Pauli σ_k^x for all hidden units k, which can be done by applying a Hadamard gate on each hidden unit before measuring them in the computational basis. By repeating these measurements $\mathcal{O}\{1/\varepsilon^2 \log[K + K_L^2(L-1)]/\delta\}$ times and averaging the resulting vector samples, we can estimate each component of the gradient to precision ε with probability $1 - \delta$ using Chebyshev's inequality and the powering lemma.

(b) We can use amplitude estimation to compute the expected value of each hidden unit, Hadamard-transformed hidden unit and pair of hidden units. But every single term has to be estimated individually, which amounts to a total number of calls to the Gibbs state preparation unitary in $\mathcal{O}\{[K + K_L^2(L-1)]/\varepsilon \log(\delta^{-1})\}$ to get all estimates to precision ε with probability $1 - \delta$. Note that this approach only applies to Gibbs state preparation subroutines that create a coherent encoding of the Gibbs sate [see Eq. (16) and associated footnote] since we need a coherent preparation of our quantum state to apply amplitude estimation on it [122]. It is hence compatible with the quantum simulated annealing and variational approaches but not the Hamiltonian simulation method.

∎

Note that the clamped model Hamiltonians now act on a subspace of size 2^K for a model with K hidden units. This, as well as a bound $\|H^{\boldsymbol{\theta}}\|$ on the model's clamped energy that is potentially smaller (since action units are fixed), leads to different complexities T_{Gibbs} in Theorem 1 and Table I. Moreover, while the preparation of the clamped

Gibbs state $\rho_{\mathbf{s},\mathbf{a}}$ is straightforward for the Hamiltonian simulation and quantum simulated annealing approaches, we point out that one needs to train again a new Ansatz $|\psi_{\boldsymbol{\theta}}\rangle$ in the variational approach. This is because, in general, one has no way of efficiently clamping state and action units in the Ansatz trained on the full Hamiltonian (postselection on measurements or amplitude amplification are possible but resource consuming). We consider this to be an interesting question that could be explored in future work.

VI. CONCLUSION AND OUTLOOK

Recent years have seen a massive surge of interest in leveraging results from quantum-information theory to enhance machine-learning methods and models. Nonetheless, deep reinforcement learning, as one of the emerging fields in machine learning, remains largely unaffected by this trend. This is in part due to an effort made by the machine-learning community to avoid bottlenecks resulting from application-specific algorithms in favor of general applicability.

In this paper, we consider deep energy-based models specifically designed to cope with large state and action spaces in complex RL environments. We demonstrate that these models can indeed outperform standard RL architectures such as deep Q networks due to their capacity to represent complex correlations between state-action pairs.

The enhanced representational power of deep energy-based models comes at the cost of sampling from an associated probability distribution. Even approximate sampling is a hard problem and therefore, gives rise to a computational bottleneck. In this paper, we address this bottleneck by leveraging results from quantum approximate sampling algorithms in order to enable various models for deep energy-based RL.

Our main result is a hybrid deep energy-based RL algorithm that enables quadratic quantum speed ups with full-scale quantum computers and is applicable to various deep energy-based models. In the interest of deploying near-term quantum devices, we further prove that the same algorithm can be combined with deep Boltzmann machines and quantum Boltzmann machines to provide heuristic quantum speed ups with near-term quantum computers.

In our work, we consider both methods, which require full fault-tolerant devices (that are, according to recent investigations, still out of reach [123,124]), as well as variational methods, which may be more compatible with near-term quantum computers (see Table I). When considering near-term restrictions, we must account for size limits, gate noise, decoherence, and architecture constraints. With respect to size, as we detail in Appendix D, for a deep or quantum Boltzmann machine with n units, one would only require $2n + 3$ qubits to run our variational methods. This means that interesting and useful algorithms could be run with the order of 100 qubits, so well in the NISQ-era range.

Of course, any such implementation will still suffer from noise and decoherence effects, but these can be (partially) mitigated through error-mitigation techniques [125,126]. Errors that remain will still affect the computation, but as has been argued previously, in ML contexts, this may be less detrimental [2,127]. Performance of such near-term implementations remains an open area of research.

The presented framework for quantum energy-based RL does not limit itself to the methods and models presented here. Following current advances in quantum machine learning, we expect genuinely quantum function approximators such as parametrized quantum circuits [14,128–130], quantum generative models [41], or quantum kernels [131–133] to present interesting features (e.g., the ability to represent quantum correlations) that could provide advantages in complex RL environments. Moreover, our study of sampling algorithms has been considering the restrictive setting of isolated (or context-free) sampling distributions. The RL setting however offers an online structure that could be exploited by the sampling algorithms. For instance, inspired by existing quantum algorithms that speed up sampling from sequences of slowly evolving Markov chains [115], one could adapt the Markov chain-based approach of quantum simulated annealing to exploit the property that the agent's policy is slowly evolving as well during training. Additionally, one could adapt other heuristic methods that accelerate approximate sampling such as multicanonical Markov chain Monte Carlo [134] to a reinforcement learning scenario, as to encourage exploration and learning multimodal policies [135]. In sum, our framework opens up a new avenue for connecting a large class of quantum models and methods to be applied in the context of quantum RL.

APPENDIX A: RANDOM AND QUANTUM WALKS

1. Mixing-time random walks

A discrete-time random walk on a graph is described by a Markov chain (MC) that is specified by a transition

matrix P gathering the transition probabilities $[P]_{i,j} = P_{ij}$ from node i to node j of the graph. When the MC is ergodic (i.e., irreducible and aperiodic), it has a unique stationary distribution $\boldsymbol{\pi} = P\boldsymbol{\pi}$ over the nodes of the graph. $\boldsymbol{\pi}$ can be approximated by repeated applications $P^t \boldsymbol{\pi}_0$ of P to any initial distribution $\boldsymbol{\pi}_0$, or equivalently, by applying t steps of the random walk P to any initial distribution $\boldsymbol{\pi}_0$ over the nodes of the graph (see Fig. 11). The minimal number of applications of P needed to obtain an ϵ'-close approximation of $\boldsymbol{\pi}$ (in l_p norm) starting from *any* $\boldsymbol{\pi}_0$ is called the *mixing time* of P:

$$t_{\epsilon'}^{\text{mix}} = \min\{t : \forall \boldsymbol{\pi}_0, \, \|P^t \boldsymbol{\pi}_0 - \boldsymbol{\pi}\|_p \leq \epsilon'\}. \quad (A1)$$

When the MC is moreover reversible, its mixing time can be related to its spectral properties. More especially, we have [103]

$$\left\lfloor \frac{1-\delta}{2\log(2\epsilon')} \frac{1}{\delta} \right\rfloor \leq t_{\epsilon'}^{\text{mix}} \leq \left\lceil \frac{1}{\delta} \log\left(\frac{1}{\epsilon' \pi^*}\right) \right\rceil, \quad (A2)$$

where $\delta = 1 - \max_{i>0} |\lambda_i| \in (0, 1]$ (i.e., the difference between the largest and second largest eigenvalues in absolute value of the matrix P) is the *spectral gap* of P and $\pi^* = \min_i\{\pi_i\}$. We say that the mixing time of a reversible MC P is of the order $\tilde{\mathcal{O}}(\delta^{-1})$ [136].

2. Metropolis-hastings (MH)

The MH algorithm is a general Markov chain Monte Carlo method that allows one to sample from a probability distribution over a configuration space $\mathcal{X}$ given only access to its corresponding density function $f(\mathbf{x})$ without having to evaluate its value for all $\mathbf{x}$. When dealing with the energy-based models listed in Sec. V, $\mathcal{X}$ can either be the space $\mathcal{A}$ of all actions and $f(\mathbf{x}) = -F^{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a})$ for a fixed state $\mathbf{s}$, or $\mathcal{X}$ is the space $\mathcal{A} \times \mathcal{H}$ of all configurations of action and hidden units and $f(x) = -E^{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}, \mathbf{h})$. MH constructs a reversible MC P having the desired distribution as stationary distribution by transforming a candidate transition matrix K into P. This construction starts by introducing an acceptance ratio $R_{\mathbf{x}\mathbf{x}'} = \min\{1, \exp[\beta f(\mathbf{x}')]K_{\mathbf{x}'\mathbf{x}}/\exp[\beta f(\mathbf{x})]K_{\mathbf{x}\mathbf{x}'}\}$ for each

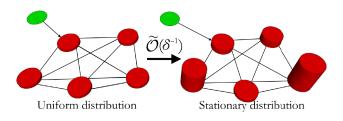Uniform distribution   Stationary distribution

FIG. 11. Mixing-time random walks. Note that the graphs are generally nonsymmetric (that is, the connections are directed), but are represented symmetric for simplicity.

pair $(\mathbf{x}, \mathbf{x}') \in \mathcal{X}$, which is used to define P as follows:

$$P_{\mathbf{x}\mathbf{x}'} = \begin{cases} R_{\mathbf{x}\mathbf{x}'} K_{\mathbf{x}\mathbf{x}'} & \text{if } \mathbf{x}' \neq \mathbf{x} \\ 1 - \sum_{\mathbf{x}'' \neq \mathbf{x}} R_{\mathbf{x}\mathbf{x}''} K_{\mathbf{x}\mathbf{x}''} & \text{if } \mathbf{x}' = \mathbf{x} \end{cases}. \quad (A3)$$

In practice, K is chosen such that the sets $N(\mathbf{x}) = \{\mathbf{x}' : K_{\mathbf{x}\mathbf{x}'} > 0\}$ define a neighborhood structure on $\mathcal{X}$ and $\max_{\mathbf{x}\in\mathcal{X}} N(\mathbf{x}) \ll |\mathcal{X}|$. A possible choice for $N(\mathbf{x})$ and $K(\mathbf{x}, \cdot)$ is, e.g., a Hamming ball around $\mathbf{x}$ and the uniform distribution over it, respectively.

If $K_{\mathbf{x}\mathbf{x}} > 0 \; \forall \mathbf{x} \in \mathcal{X}$ and the graph defined by N is connected, then MH [137] ensures that P is reversible and has as stationary distribution:

$$\pi_\beta(\mathbf{x}) = \frac{e^{\beta f(\mathbf{x})}}{\sum_{\mathbf{a}'} e^{\beta f(\mathbf{x}')}}. \quad (A4)$$

3. Gibbs sampling (GS)

Gibbs sampling [138] is a particular instance of the Metropolis-Hastings algorithm where the neighborhood $N(\mathbf{x})$ is a Hamming ball of radius 1 around $\mathbf{x}$ (including $\mathbf{x}$ itself) and the candidate transition matrix K is given by

$$K_{\mathbf{x}\mathbf{x}'} = \begin{cases} \frac{1}{\log|\mathcal{X}|} \frac{e^{\beta f(\mathbf{x}')}}{e^{\beta f(\mathbf{x})} + e^{\beta f(\mathbf{x}')}} & \text{if } \mathbf{x}' \in N(\mathbf{x}) \\ 0 & \text{if } \mathbf{x}' \notin N(\mathbf{x}) \end{cases}. \quad (A5)$$

This K has the special property that its corresponding acceptance ratio $R_{\mathbf{x}\mathbf{x}'} = 1$ for all $\mathbf{x}' \in N(\mathbf{x})$. In particular, the conditions for reversibility of P are always met.

4. Szegedy walks

Szegedy-type quantum walks [112,113] are a quantum analog of classical random walks. They generalize an MC P to a unitary $W(P)$ called the *Szegedy walk operator*. Here we describe how this unitary can be constructed and explain its interesting properties.

Let $\mathcal{H}$ be the Hilbert space that shares the same basis vectors as the MC $P = [P_{ij}]$. So-called *quantum diffusion operators* U_P and V_P act on $\mathcal{H} \otimes \mathcal{H}$ as

$$\begin{cases} U_P |i\rangle |0\rangle = |i\rangle \sum_j \sqrt{P_{ij}} |j\rangle \\ V_P |0\rangle |i\rangle = \sum_j \sqrt{P_{ij}^*} |j\rangle |i\rangle \end{cases},$$

where P^* is the time-reversed MC [139] associated to P. Its elements are given by $P_{ij}^* = P_{ji}\pi_i/\pi_j$, where $\boldsymbol{\pi} = (\pi_i)$ is the stationary distribution of P. When $P^* = P$, we say that the MC is *reversible*. Although Szegedy walks can be defined if this property is not fulfilled, one would require additional access to P^* to be able to implement V_P. Because this is not usually the case (the expression of P^* involves the stationary distribution of P), the reversibility of P is then crucial for the implementation of V_P.

With the quantum diffusion operators U_P and V_P at hand, we can construct the Szegedy walk operator $W(P)$, which is the reflection over the spaces A and B:

$$\begin{cases} A = \text{span}\{U_P \ket{i} \ket{0} : i = 1, \ldots, N\} \\ B = \text{span}\{V_P \ket{0} \ket{i} : i = 1, \ldots, N\} \end{cases}$$

$$W(P) = \text{ref}(B)\,\text{ref}(A),$$

where

$$\begin{cases} \text{ref}(A) = U_P\,[\mathbb{1}_N \otimes \text{ref}(0)]\,U_P^\dagger \\ \text{ref}(B) = V_P\,[\text{ref}(0) \otimes \mathbb{1}_N]\,V_P^\dagger \end{cases} \quad (A6)$$

and

$$\text{ref}(0) = 2\ket{0}\bra{0} - \mathbb{1}_N.$$

We refer to $A + B$ as the busy subspace and to its orthogonal complement $A^\perp \cap B^\perp$ as the idle subspace. $W(P)$ acts as the identity on the idle subspace, but, on the busy subspace, its unique eigenvector with eigenvalue 1 (or, equivalently, eigenphase 0) is

$$\ket{\pi}\ket{0} = \sum_i \sqrt{\pi_i} \ket{i}\ket{0}.$$

This follows from Refs. [112,113].

As we are interested in the spectral gap δ of P classically, we are interested in the *phase gap* $\Delta = 2\min_{i>0} |\theta_i|$, i.e., the smallest nonzero eigenphase of $W(P)$, in the quantum case. Given that the eigenvalues of P and the eigenphases of $W(P)$ are related by $|\lambda_i| = \cos(\theta_i)\ \forall i$, it can be shown [112,113] that

$$\Delta \geq \sqrt{2\delta},$$

i.e., the phase gap of $W(P)$ is quadratically larger than the spectral gap of P. This is a crucial property for Szegedy quantum walks as a "quantum search" [58,111] for $\ket{\pi}\ket{0}$ on the busy subspace of $W(P)$ is governed by its phase gap as $\tilde{\mathcal{O}}(\Delta^{-1})$. This is quadratically faster than sampling from π classically.

APPENDIX B: BOLTZMANN MACHINES AND ENERGY-BASED MODELS

1. Restricted Boltzmann machines

Boltzmann machines [140] are energy-based generative models, meaning that they can learn probability distributions $\hat{p}(\mathbf{x})$ over a set $\mathcal{X} \subset \{0,1\}^n$, by encoding a parametrized distribution $p^{\boldsymbol{\theta}}(\mathbf{x})$. This probability distribution is specified by an energy function $E^{\boldsymbol{\theta}}(\mathbf{x})$ defined over the entire set $\mathcal{X}$ and normalized by a Boltzmann-Gibbs distribution:

$$P^{\boldsymbol{\theta}}(\mathbf{x}) = \frac{e^{-E^{\boldsymbol{\theta}}(\mathbf{x})}}{\sum_{\mathbf{x}'} e^{-E^{\boldsymbol{\theta}}(\mathbf{x}')}}. \quad (B1)$$

This energy function takes the form of an Ising model spin Hamiltonian, which accounts for pair interactions w_{ik} and individual external magnetic fields b_i $\{[(w_{ik})_{i,k}, (b_i)_i] = \boldsymbol{\theta} \in \mathbb{R}^d\}$ acting between and over spins x_i and x_k. Different configurations of these spins correspond to different binary vectors $\mathbf{x} \in \mathcal{X}$. Commonly used types of BMs are the so-called restricted Boltzmann machines [79,141], characterized by their bipartite structure (see Fig. 3), which restricts the interactions w_{ik} in the Ising model Hamiltonian and energy function. RBMs divide the spins $\{x_i\}_i$ between two subsets $\mathbf{v} = \{v_i\}_i$ and $\mathbf{h} = \{h_k\}_k$, respectively, called visible and hidden units. The visible units represent the data that we want to encode in the model, while the hidden (or latent) units allow more degrees of freedom to be had in the specification of the energy function and allow more complex distributions to be represented over the visible units. The RBM model effectively encodes the probability distribution:

$$P^{\boldsymbol{\theta}}(\mathbf{v}) = \frac{\sum_{\mathbf{h}} e^{-E^{\boldsymbol{\theta}}(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{v}',\mathbf{h}'} e^{-E^{\boldsymbol{\theta}}(\mathbf{v}',\mathbf{h}')}} = \frac{e^{-F^{\boldsymbol{\theta}}(\mathbf{v})}}{\sum_{\mathbf{v}'} e^{-F^{\boldsymbol{\theta}}(\mathbf{v}')}}, \quad (B2)$$

where

$$-E^{\boldsymbol{\theta}}(\mathbf{v}, \mathbf{h}) = \sum_{i,k} w_{ik} v_i h_k + \sum_i b_i v_i + \sum_k b_k h_k \quad (B3)$$

and

$$\begin{aligned} F^{\boldsymbol{\theta}}(\mathbf{v}) &= -\log\left[\sum_{\mathbf{h}'} e^{-E^{\boldsymbol{\theta}}(\mathbf{v},\mathbf{h}')}\right] \\ &= -\log\left[\frac{e^{-E^{\boldsymbol{\theta}}(\mathbf{v},\mathbf{h})}}{P(\mathbf{h}|\mathbf{v})}\right] \quad \forall \mathbf{h} \\ &= E^{\boldsymbol{\theta}}(\mathbf{v}, \mathbf{h}) + \log[P(\mathbf{h}|\mathbf{v})] \quad \forall \mathbf{h} \\ &= \sum_{\mathbf{h}} P(\mathbf{h}|\mathbf{v})\{E^{\boldsymbol{\theta}}(\mathbf{v},\mathbf{h}) + \log[P(\mathbf{h}|\mathbf{v})]\} \\ &= E^{\boldsymbol{\theta}}[\mathbf{v}, \langle \mathbf{h} \rangle_{P(\mathbf{h}|\mathbf{v})}] - H(\mathbf{h}|\mathbf{v}) \end{aligned}$$

is the so-called *free energy* of the RBM (the average energy after tracing out the hidden units).

2. Derivation of the free-energy neural network

We derive below the expression of a RBM free energy as a feedforward neural network. This derivation is inspired by Ref. [75].

Starting from the expression of a RBM probability distribution:

$$P^{\theta}(\mathbf{v}) = \frac{1}{Z^{\theta}} \sum_{\mathbf{h}} \exp\left(\sum_{i,k} w_{ik} v_i h_k + \sum_i b_i v_i + \sum_k b_k h_k\right)$$

$$= \frac{1}{Z^{\theta}} \exp\left(\sum_i b_i v_i\right) \prod_k \sum_{h_k \in \{0,1\}} \exp\left(\sum_i w_{ik} v_i h_k + b_k h_k\right) \quad \# \sum_{\mathbf{x} \in \{0,1\}^2} e^{ax_1 + bx_2} = (1 + e^a)(1 + e^b)$$

$$= \frac{1}{Z^{\theta}} \exp\left(\sum_i b_i v_i\right) \prod_k \left[1 + \exp\left(\sum_i w_{ik} v_i + b_k\right)\right] \quad \# \sum_{\mathbf{x} \in \{0,1\}^2} e^{ax_1 + bx_2} = (1 + e^a)(1 + e^b)$$

$$= \frac{1}{Z^{\theta}} \exp\left(\sum_i b_i v_i\right) \exp\left\{\sum_k \log\left[1 + \exp\left(\sum_i w_{ik} v_i + b_k\right)\right]\right\} \quad \# \prod_k a_k = \exp\left[\sum_k \log(a_k)\right]$$

$$= \frac{1}{Z^{\theta}} \exp\left[\sum_i b_i v_i + \sum_k \mathrm{softplus}\left(\sum_i w_{ik} v_i + b_k\right)\right] \quad \# \mathrm{softplus}(x) = \log[1 + \exp(x)]$$

$$= \frac{1}{Z^{\theta}} \exp\left[-F^{\theta}(\mathbf{v})\right]$$

we end up with

$$-F^{\theta}(\mathbf{v}) = \sum_i b_i v_i + \sum_k \mathrm{softplus}\left(\sum_i w_{ik} v_i + b_k\right),$$
(B4)

which takes the form of the feedforward neural network depicted in Fig. 4. See Fig. 12 for a comparison between the softplus function and the commonly used Rectified Linear Unit (ReLU) activation function for NNs.

3. Eigendecomposition of H_{QBM}

We write the QBM Hamiltonian in the computational basis $(\mathbf{v}, \mathbf{h})$, where we divide the Hilbert space on which it acts into visible and hidden units $\mathcal{V} \otimes \mathcal{H}$. Separating
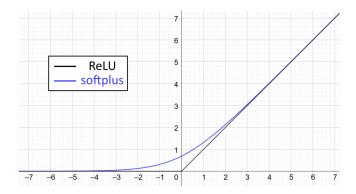

FIG. 12. Comparison between the commonly used ReLU activation function $\max(0,x)$ and the softplus function $x + \log(1 + e^{-x}) = \log(1 + e^x)$ appearing in the expression of the free energy of a RBM.

the σ^z and $\sigma^z \sigma^z$ operators from the σ^x operators in the Hamiltonian of Eq. (13) allows the latter to be expressed as

$$H_{\mathrm{QBM}} = \begin{pmatrix} H_1^{z,zz} & 0 & \cdots & 0 \\ 0 & H_2^{z,zz} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & H_{|\mathcal{V}| \times |\mathcal{H}|}^{z,zz} \end{pmatrix}$$

$$+ I_{\mathbf{v}} \otimes \begin{pmatrix} H_{1,1}^x & \cdots & H_{1,|\mathcal{H}|}^x \\ \vdots & \ddots & \vdots \\ H_{|\mathcal{H}|,1}^x & \cdots & H_{|\mathcal{H}|,|\mathcal{H}|}^x \end{pmatrix}$$

$$= \begin{pmatrix} M_1 & 0 & \cdots & 0 \\ 0 & M_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & M_{|\mathcal{V}|} \end{pmatrix} = \sum_{\mathbf{v}} |\mathbf{v}\rangle \langle \mathbf{v}| \otimes M_{\mathbf{v}},$$

where $\{M_1, \ldots, M_{|\mathcal{V}|}\}$ are nondiagonal matrices acting on the subspaces $\{\mathrm{span}\{\mathbf{v}_1\} \otimes \mathcal{H}, \ldots, \mathrm{span}\{\mathbf{v}_{|\mathcal{V}|}\} \otimes \mathcal{H}\}$, respectively, for $\mathbf{v}_i$ computational basis states of visible units. This block diagonal form implies that the eigenvectors of H_{QBM} are block vectors of the form $|\mathbf{v}, \mathbf{h_v}\rangle$, where $|\mathbf{v}\rangle$ is a computational basis state and $|\mathbf{h_v}\rangle$ can be a superposition of hidden unit configurations.

4. Derivation of the PS error function

We show in Sec. III A how to derive an error function from the Q-learning update rule. Due to its shared origin, deriving an error function from the SARSA update rule can be done similarly. In contrast, h values are not linked to the

Bellmann equation and hence do not represent estimates of the (bounded) return. Instead, they are allowed to accumulate rewards indefinitely (and hence diverge). Diverging h values can still lead to converging policies after normalization but they constitute a problem for the derivation of an error function, as this error can never be minimized to 0, hence making neural networks harder to train. One mechanism in PS that allows us to avoid this divergence is damping, and we show here how it can be translated into a regularization term in the PS error function.

Consider the original update rule of PS [45]:

$$h_{ij} \leftarrow h_{ij} - \gamma_{\text{PS}}(h_{ij} - 1) + g_{ij}\lambda, \qquad (B5)$$

where $\gamma_{\text{PS}} \in [0, 1]$ is an internal damping (or forgetfulness) parameter allowing the agent to adapt to changing environments, λ is the last *positive* reward experienced by the agent, and the edge glow g_{ij} is a varying rescaling factor accounting for delayed rewards.

Let us take this update rule with damping $\gamma_{\text{PS}} = 0$ and edge glow $g_{ij} = 1$:

$$h_{ij} \leftarrow h_{ij} + \lambda. \qquad (B6)$$

This setup cancels the damping mechanism and supposes that edge-glow values $g_{ij} \neq 1$ are absorbed in the reward λ. This update rule gives the following error function:

$$\mathcal{E}_{\text{PS}}[\mathbf{s}^{(t)}, \mathbf{a}^{(t)}] = r^{(t+1)}, \qquad (B7)$$

which is never 0 as long as the given transition is rewarded (which is the case in a static environment) and leads to diverging h values.

To avoid this divergence, we take into account damping by effectively introducing a regularization term:

$$\mathcal{E}_{\text{PS}}[\mathbf{s}^{(t)}, \mathbf{a}^{(t)}] = r^{(t+1)} - \gamma_{\text{PS}}h_{ij} \qquad (B8)$$

such that the error is now 0 when $h_{ij} = r^{(t+1)}/\gamma_{\text{PS}}$, hence effectively bounding the h values. Note that $\mathcal{E}_{\text{PS}}[\mathbf{s}^{(t)}, \mathbf{a}^{(t)}] = h_{ij}^{(t+1)} - h_{ij}^{(t)}$, so this new error corresponds to the update rule:

$$h_{ij} \leftarrow (1 - \gamma_{\text{PS}})h_{ij} + \lambda, \qquad (B9)$$

which is the original PS update rule [Eq. (B5), with the edge glow fixed to 1].

APPENDIX C: DEFERRED THEOREMS AND DERIVATIONS FOR QUANTUM SUBROUTINES

Theorem 4 (Harrow and Wei [97, Theorem 10]). *Assume that we are given a prior distribution $\Pi_0(x)$ and a likelihood function $L(x) \geq 0$, so that we can parametrize the partition function $Z(\beta') = \sum_x \Pi_0(x)e^{-\beta'L(x)}$ at each*

inverse temperature $\beta' \in [0, 1]$. Assume that we can generate the state $|\Pi_0\rangle$ corresponding to the coherent encoding of the prior, and assume that for every inverse temperature β' we have a Markov chain $P_{\beta'}$ with stationary distribution $|\Pi_{\beta'}\rangle$ and spectral gap lower bounded by δ. Then, for any $\varepsilon > 0$, $\eta > 0$, there is a quantum algorithm that, with probability at least $1 - \eta$, produces a state $|\widetilde{\Pi_1}\rangle$ so that $\left\| |\widetilde{\Pi_1}\rangle - |\Pi_1\rangle \right\| \leq \varepsilon$ for $|\Pi_1\rangle$ the coherent encoding of the posterior distribution $\Pi_1(x) = \Pi_0(x)[e^{-L(x)}/Z(1)]$. The algorithm uses

$$\widetilde{\mathcal{O}} \left\{ \sqrt{\mathbb{E}_{\Pi_0}[L(x)]/\delta} \right\}$$

total steps of the quantum-walk operators corresponding to the Markov chains $M_{\beta'}$.

In the case of a DEBN, we use Theorem 4 with $x = \mathbf{a}$, $L(x) = -\beta M_{-}^{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a})$ and $\Pi_0(x) = \frac{1}{|\mathcal{A}|}$. $M_{-}^{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a})$ is obtained from $M^{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a})$ by the transformation $M_{-}^{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}) = M^{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}) - \max\{0, \max_{\mathbf{a}} M^{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a})\}$, such that $-2\max_{\mathbf{a}} |M^{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a})| \leq M_{-}^{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}) \leq 0$, i.e., $0 \leq L(x) \leq 2\beta \max_{\mathbf{a}} \|M^{\boldsymbol{\theta}}\|$. Hence, sampling from the distribution $\pi_\beta(\mathbf{a}|\mathbf{s}) = e^{\beta M_{-}^{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})}/Z_{-,\beta}(\mathbf{s}) = e^{\beta M^{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})}/Z_\beta(\mathbf{s})$ has complexity in $\widetilde{\mathcal{O}}\left\{\sqrt{\mathbb{E}_{\Pi_0}[L(x)]/\delta}\right\} = \widetilde{\mathcal{O}}\left[\sqrt{(1/|\mathcal{A}|)\sum_{\mathbf{a}} L(\mathbf{a})/\delta}\right] \leq \widetilde{\mathcal{O}}\left(\sqrt{\beta\|M^{\boldsymbol{\theta}}\|/\delta}\right)$.

Note: while the value $\|M^{\boldsymbol{\theta}}\| = \max_{\mathbf{a}} |M^{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a})|$ is not needed to implement the quantum-walk operator of P'_β associated to $M_{-}(\mathbf{s}, \mathbf{a})$ (see Appendix E), it is actually needed in a subroutine behind the algorithm of Theorem 4 (see Algorithm 1 in Ref. [97]).

A similar approach can be taken for DBMs, where now $x = (\mathbf{a}, \mathbf{h})$, $L(x) = \beta E_{-}(\mathbf{s}, \mathbf{a}, \mathbf{h})$ and $\Pi_0(x) = 1/|\mathcal{A}|2^K$. The resulting complexity is now in $\widetilde{\mathcal{O}}\left[\sqrt{(1/|\mathcal{A}|2^K)\sum_{\mathbf{a},\mathbf{h}} L(\mathbf{a}, \mathbf{h})/\delta}\right] \leq \widetilde{\mathcal{O}}\left(\sqrt{\beta\|M^{\boldsymbol{\theta}}\|/\delta}\right)$ (for a different spectral gap δ).

Theorem 5 (Gibbs state preparation with sparse access [99, Lemma 44]). *Let $H \in \mathbb{C}^{N \times N}$ be a d-sparse Hamiltonian. Suppose that $I \preceq H$, we are given $C \in \mathbb{R}_+$ such that $\|H\| \leq 2C$, and we know a lower bound $z \leq Z = \text{tr}[e^{-H}]$. If $\epsilon \in (0, 1/3)$, then we can prepare a purified Gibbs state $|\gamma\rangle_{AB}$ such that*

$$\left\| \text{Tr}\|_B \left[|\gamma\rangle \langle\gamma|_{AB}\right] - \frac{e^{-H}}{Z} \right\| \leq \epsilon$$

using

$$\mathcal{O}\left[\sqrt{\frac{N}{z}}Cd\log\left(\frac{C}{\epsilon}\right)\log\left(\frac{1}{\epsilon}\right)\right]$$

queries, and

$$\mathcal{O}\left\{\sqrt{\frac{N}{z}}Cd\log\left(\frac{C}{\epsilon}\right)\log\left(\frac{1}{\epsilon}\right)\left[\log(N)+\log^{5/2}\left(\frac{C}{\epsilon}\right)\right]\right\}$$

gates.

For the DEBN, we have $N = 2^A$, while $N = 2^{AK}$ for the DBM and QBM, where K is the number of hidden units of these models. We enforce the condition $I \preceq H$ by choosing $H = \beta H_{\text{model}} - (\beta\tilde{\lambda}_{\text{min}} - 3/2)I$, where $\tilde{\lambda}_{\text{min}}$ is either bounded by the weights of the model or estimated via a minimum finding subroutine. This choice also gives us natural bounds $z = e^{-2}$ and $C = \beta\|H_{\text{model}}\|$. Since the DEBN and DBM Hamiltonians are diagonal, their sparsity is $d = 1$. As for the QBM Hamiltonian, the σ^x terms each add at most one nonzero entry to each row and column, which leads to a sparsity $d = 1 + K$. Hence the complexity of preparing the purified Gibbs states of the DEBN and DBM and QBM is in $\tilde{\mathcal{O}}\left(\sqrt{2^A}\beta\|H_{\text{model}}\|\right)$ and $\tilde{\mathcal{O}}\left(\sqrt{2^{AK}}\beta\|H_{\text{model}}\|\right)$, respectively.

Theorem 6 (Average energy estimation [100, Theorem 2]). *Let H be a Hamiltonian that can be represented as a linear combination of unitaries $H = \sum_{k=1}^{K}\alpha_k U_k$ with $|\alpha|_1 = \sum_k |\alpha_k|$ for $K \in O(\log N)$, and let U_ψ be a unitary that prepares a state $|\psi\rangle$. The number of applications of the unitaries U_P, U_S and the unitary U_ψ to estimate the average energy $\langle\psi|H|\psi\rangle$ within error ϵ with probability at least 2/3 is in $\mathcal{O}\left(\|\alpha\|_1/\epsilon\right)$.*

Theorem 7 (von Neumann entropy estimation [100, Theorem 1]). *Given access to an oracle U_ρ that prepares a purification $\sum_j \sqrt{p_j}|j\rangle|\psi_j\rangle$ of a density matrix $\rho = \sum_j p_j|\psi_j\rangle\langle\psi_j|$ where $p_j \geq p_{\text{min}}$, the von Neumann entropy of the state ρ can be estimated within error ϵ with probability greater than 2/3 with $\tilde{\mathcal{O}}[1/(p_{\text{min}}^2\epsilon)]$ preparations of the purified density operator.*

1. Derivation of the gradient expression for DBMs

This derivation is relatively easy because we can use the property $\partial_\theta e^{-H} = -\partial_\theta H e^{-H}$ with a diagonal Hamiltonian H. Define $Z_{\mathbf{s,a}} = \sum_{\mathbf{h}} e^{-H(\mathbf{s,a,h})}$, then

$$\begin{aligned}
\frac{\partial F}{\partial\theta}(\mathbf{s,a}) &= -\frac{\partial}{\partial\theta}\log(Z_{\mathbf{s,a}}) \\
&= -\frac{1}{Z_{\mathbf{s,a}}}\frac{\partial Z_{\mathbf{s,a}}}{\partial\theta} \\
&= -\frac{1}{Z_{\mathbf{s,a}}}\sum_{\mathbf{h}}\frac{\partial e^{-H(\mathbf{s,a,h})}}{\partial\theta} \\
&= \frac{1}{Z_{\mathbf{s,a}}}\sum_{\mathbf{h}}\frac{\partial H(\mathbf{s,a,h})}{\partial\theta}e^{-H(\mathbf{s,a,h})}
\end{aligned}$$

$$\begin{aligned}
&= \sum_{\mathbf{h}}P(\mathbf{h}|\mathbf{s,a})\frac{\partial H(\mathbf{s,a,h})}{\partial\theta} \\
&= \langle\partial_\theta H\rangle_{P(\mathbf{h}|\mathbf{s,a})},
\end{aligned}$$

where $\partial_\theta H$ is the σ_l^z or $\sigma_l^z\sigma_{l'}^z$ operator associated to the parameter θ (or equivalently the random variables associated to units l and l' of the DBM), depending if the latter is a bias or weight respectively. $P(\mathbf{h}|\mathbf{s,a})$ is the Gibbs distribution obtained by fixing the operators σ^z associated to state and action units to the configuration $(\mathbf{s,a})$.

2. Derivation of the gradient expression for QBMs

Since the QBM Hamiltonian involves both σ^x and σ^z terms, we have that H and $\partial_\theta H$ do not commute, and hence $\partial_\theta e^{-H} \neq -\partial_\theta H e^{-H}$. This makes the derivation of the gradient a bit more involved.

Define $Z_{\mathbf{s,a}} = \text{Tr}\left[\Lambda_{\mathbf{s,a}}e^{-H}\right]$, then

$$\begin{aligned}
\frac{\partial F}{\partial\theta} &= -\frac{\partial}{\partial\theta}\log(Z_{\mathbf{s,a}}) \\
&= -\frac{1}{Z_{\mathbf{s,a}}}\frac{\partial Z_{\mathbf{s,a}}}{\partial\theta} \\
&= -\frac{\text{Tr}\left[\Lambda_{\mathbf{s,a}}\partial_\theta e^{-H}\right]}{\text{Tr}\left[\Lambda_{\mathbf{s,a}}e^{-H}\right]}.
\end{aligned}$$

Since $\partial_\theta e^{-H} \neq -\partial_\theta H e^{-H}$, we now rely on Duhamel's formula to express

$$\partial_\theta e^{-H} = -\int_0^1 e^{-\tau H}\partial_\theta H e^{(\tau-1)H}d\tau,$$

$$\Rightarrow \Lambda_{\mathbf{s,a}}\partial_\theta e^{-H} = -\int_0^1 \Lambda_{\mathbf{s,a}}e^{-\tau H}\partial_\theta H e^{(\tau-1)H}d\tau.$$

We then use the circular permutation property of the trace to get

$$\text{Tr}\left[\Lambda_{\mathbf{s,a}}\partial_\theta e^{-H}\right] = -\int_0^1 \text{Tr}\left[\partial_\theta H e^{(\tau-1)H}\Lambda_{\mathbf{s,a}}e^{-\tau H}\right]d\tau.$$

Using the eigendecompostion of $H_{\text{QBM}} = \sum_{\mathbf{v,h_v}}\lambda_{\mathbf{v,h_v}}|\mathbf{v,h_v}\rangle\langle\mathbf{v,h_v}|$ (Appendix B), we find

$$\begin{aligned}
e^{(\tau-1)H}\Lambda_{\mathbf{v}}e^{-\tau H} &= \sum_{\mathbf{v',h_v'}}e^{(\tau-1)\lambda_{\mathbf{v',h_v'}}}|\mathbf{v',h_v'}\rangle\langle\mathbf{v',h_v'}| \\
&\quad \Lambda_{\mathbf{v}}\sum_{\mathbf{v',h_v'}}e^{-\tau\lambda_{\mathbf{v',h_v'}}}|\mathbf{v',h_v'}\rangle\langle\mathbf{v',h_v'}| \\
&= \sum_{\mathbf{h_v}}e^{-\lambda_{\mathbf{v,h_v}}}|\mathbf{v,h_v}\rangle\langle\mathbf{v,h_v}| \\
&= \Lambda_{\mathbf{v}}e^{-H}\Lambda_{\mathbf{v}}
\end{aligned}$$

and hence

$$\mathrm{Tr}\left[\Lambda_{\mathbf{s},\mathbf{a}}\partial_{\theta}e^{-H}\right] = -\int_{0}^{1}\mathrm{Tr}\left[\partial_{\theta}H\Lambda_{\mathbf{s},\mathbf{a}}e^{-H}\Lambda_{\mathbf{s},\mathbf{a}}\right]d\tau$$
$$= -\mathrm{Tr}\left[\partial_{\theta}H\Lambda_{\mathbf{s},\mathbf{a}}e^{-H}\Lambda_{\mathbf{s},\mathbf{a}}\right]$$

and

$$\frac{\partial F}{\partial \theta} = \frac{\mathrm{Tr}\left[\partial_{\theta}H\Lambda_{\mathbf{s},\mathbf{a}}e^{-H}\Lambda_{\mathbf{s},\mathbf{a}}\right]}{\mathrm{Tr}\left[\Lambda_{\mathbf{s},\mathbf{a}}e^{-H}\right]}$$
$$= \mathrm{Tr}[(\partial_{\theta}H)\rho_{\mathbf{s},\mathbf{a}}]$$
$$= \langle\partial_{\theta}H\rangle_{\rho_{\mathbf{s},\mathbf{a}}},$$

where now $\partial_{\theta}H$ is the σ_{k}^{x}, σ_{l}^{z} or $\sigma_{l}^{z}\sigma_{l'}^{z}$ operator associated to the parameter θ.

APPENDIX D: EVALUATION OF THE VARIATION APPROACH REQUIREMENTS

The variational approaches for Gibbs state preparation presented in Sec. V 3 involve three main components: the preparation of the parametrized Ansatz state $|\psi_{\theta}\rangle$, the evaluation of its average energy $\mathrm{Tr}[|\psi_{\theta}\rangle\langle\psi_{\theta}|H]$, and the evaluation of its von Neumann entropy $\mathrm{Tr}[|\psi_{\theta}\rangle\langle\psi_{\theta}|\log(|\psi_{\theta}\rangle\langle\psi_{\theta}|)]$. We consider here the simplest scenario imaginable in order to find the minimal number of qubits required to run these methods, using current state-of-the-art techniques. For that, we consider our model Hamiltonian to be that of a DBM and QBM whose connectivity follows the natural connectivity of the quantum hardware at hand. This Hamiltonian acts on n qubits (the number of the units in the DBM and QBM) and has at most $d = n^2 + 2n$ terms. We can take our Ansatz $|\psi_{\theta}\rangle$ to be a quantum state that belongs to the hardware-efficient family of n qubits, as in Ref. [102], or a QAOA-inspired Ansatz, as in Ref. [101] (these are seemingly the easiest quantum states to prepare, as they do not require ancillas). For the average energy estimation subroutine, we can either take a phase estimation approach as in Fig. 3 of Ref. [100], which requires $\log(d) + 1$ ancillas and $\tilde{\mathcal{O}}\left(\|H\|/\varepsilon\right)$ iterations, or

an alternative approach that only requires measuring the state $|\psi_{\theta}\rangle$ in the $Z^{\otimes n}$ basis (and $X^{\otimes n}$ basis for a QBM) without any ancillas but $\tilde{\mathcal{O}}\left[(\|H\|/\varepsilon)^2\right]$ iterations. As for the entropy estimation subroutine, we again take the phase estimation approach of Ref. [100] (Fig. 1) using 1 controlling ancilla qubit and $n + 2$ ancillas for Hamiltonian simulation of the density operator $\rho = |\psi_{\theta}\rangle\langle\psi_{\theta}|$ [using the state-of-the-art techniques of Ref. [110] (last row of Table I)]. At most, these subroutines require $2n + 3$ logical qubits to be executed. A study of the number of quantum operations needed to implement these subroutines on quantum hardware and the noise resilience of these methods would be an interesting topic of further investigation, possibly inspired by the methods of Ref. [123,124].

APPENDIX E: COHERENT ACCESS TO THE TRANSITION MATRIX

In this appendix, we detail the construction of the diffusion operator U_P required to implement quantum walks (see Appendix A). We want to implement the operation $|i\rangle_1 |0\rangle_2 |0\rangle_{\mathrm{aux}} \rightarrow |i\rangle_1 \sum_{j\in N(i)} \sqrt{P_{ij}} |j\rangle_2 |0\rangle_{\mathrm{aux}}$ coherently for all actions i and for the transition matrix P and neighborhood structure N given by Metropolis-Hastings or Gibbs sampling (Appendix A). In order to achieve this, we assume access to the description of the classical circuits that compute the operations $|i\rangle_1 |j\rangle_2 |0\rangle \rightarrow |i\rangle_1 |j\rangle_2 |P_{ij}\rangle$ and $|i\rangle |0\rangle \rightarrow |i\rangle \bigotimes_{j\in N(i)} |j\rangle$, having complexity $C(P)$ and $\mathcal{O}[\mathrm{poly}(M)]$, respectively. Using either Fredkin or Toffoli gates, it is possible to design quantum circuits that implement coherently the operation $|i\rangle |0\rangle^{\otimes M} |0\rangle_{\mathrm{aux}} \rightarrow |i\rangle |P_{i1}\rangle\cdots|P_{iM}\rangle |0\rangle_{\mathrm{aux}}$ with comparable complexities by *reversibilization* (Sec. 3.2.5. of Ref. [142]), where $M = \max_{a\in\mathcal{A}} N(a)$ and the $j \in N(i)$ have been renamed $1,\ldots,M$ for each i for simplicity. Using this circuit and a procedure called *coherent controlization* [143,144], it is possible to implement the coherent access to the columns of the transition matrix P using $\mathcal{O}[M \times C(P) + \mathrm{poly}(M)]$ operations, same as classically. This implementation goes as follows:

$$|i\rangle_1 |0\rangle_{\mathrm{aux}}^{\otimes M} |0\rangle_{\mathrm{aux}}^{\otimes M-2} |0\rangle_2^{\otimes \log(M)}$$

$$\rightarrow |i\rangle_1 |P_{i1}\rangle\cdots|P_{iM}\rangle |0\rangle_{\mathrm{aux}}^{\otimes M-2} |0\rangle_2^{\otimes \log(M)} \quad \# \, \mathrm{cost} M \times C(P) + \mathcal{O}[\mathrm{poly}(M)]$$

$$\rightarrow |i\rangle_1 |P_{i1}\rangle\cdots|P_{iM}\rangle \left|P_{i1}+\cdots+P_{i\frac{M}{2}}\right\rangle\cdots\left|P_{i,\frac{M}{2}-1}+P_{i\frac{M}{2}}\right\rangle |0\rangle_2^{\otimes \log(M)} \quad \# \, \mathrm{cost} \, M-2 \, \mathrm{additions}$$

$$\rightarrow |i\rangle_1 |P_{i1}\rangle\cdots|P_{iM}\rangle \left|P_{i1}+\cdots+P_{i\frac{M}{2}}\right\rangle\cdots\left|P_{i,\frac{M}{2}-1}+P_{i\frac{M}{2}}\right\rangle \sum_{j=1}^{M}\sqrt{P_{ij}}|j\rangle_2 \quad \# \, \mathrm{cost \, of \, CC}: \mathcal{O}[\mathrm{poly}(M)]$$

$$\rightarrow |i\rangle_1 |0\rangle_{\mathrm{aux}}^{\otimes M} |0\rangle_{\mathrm{aux}}^{\otimes M-2} \sum_{j=1}^{M}\sqrt{P_{ij}}|j\rangle_2 \quad \# \, \mathrm{cost} \, M \times C(P) + \mathcal{O}[\mathrm{poly}(M)].$$

APPENDIX F: DETAILS OF THE NUMERICAL SIMULATIONS

For reference, the Python code used for all simulations is available on GitHub [94].

For the simulations of GridWorld, we chose ADAM as an optimizer with a learning rate of $\alpha = 0.0001$. The training batches of the main network contain 100 samples and the replay memory has a capacity of 5000 interactions, updated in a first-in-first-out manner. The target network is updated every 100 calls to the replay memory while the experience replay memory is called every 100 timesteps. We fix the glow parameters of the PS update rule to a value of 0.99. The agent's policy is a softmax policy and for the β parameter we choose a hyperbolic tangent schedule that starts with a value of 0.001 and ends with a value of 0.664 (hyperbolic tangent applied to a linear schedule that starts with a value of 0.001 and ends with a value of 0.8). The neural networks have 64 units per layer.

For the simulations of CartPole, in order to have a fair comparison of the performance of neural networks with different depths, we set their number of units per hidden layer such that all neural networks have approximately the same total number of weights and biases. More precisely, we set the number of units per layer to be 10, 19, 73 for neural networks with, respectively, 5, 2, and 1 hidden layer(s), leading to a total number of weights and biases of 525, 528, 526, respectively.

For neural networks with five hidden layers, we choose ADAM as an optimizer with a learning rate of $\alpha = 0.001$. The training batches of the main network contain 100 samples and the replay memory has a capacity of 2000 interactions, updated in a first-in first-out manner. The target network is updated every 10 000 calls to the replay memory while the experience replay memory is called every five timesteps. For neural networks with one and two hidden layer(s), we choose ADAM as an optimizer with a learning rate of $\alpha = 0.01$. The training batches of the main network contain 200 samples and the replay memory has a capacity of 2000 interactions, updated in a first-in-first-out manner. The target network is updated every 5000 calls to the replay memory while the experience replay memory is called every five timesteps. For all simulations, we fix the glow parameters of the PS update rule to a value of 0.99. The agent's policy is a softmax policy and for the β parameter we choose a hyperbolic tangent schedule that starts with a value of 0.01 and ends with a value of 1 (hyperbolic tangent applied to a linear schedule that starts with a value of 0.01 and ends with a value of 10).

For the simulations of the policy-sampling experiment, in order to have a fair comparison of performance between DEBNs and DQNs, we choose to keep their total number of weights fixed to 10440, i.e., such that the minimal number of hidden units par layer of the DQNs are 10 in all instances of the experiment. For a given number of actions,

we adapt the size of the two hidden layers of the networks such that both layers have the same number of hidden units. We choose ADAM as an optimizer with a learning rate of $\alpha = 0.001$. The training batches of the neural network contain ten samples. The agent's policy is a softmax policy with a β parameter set to 1.

For the simulations of the reward-discounting experiment, in order to have a fair comparison of performance between DEBNs and DQNs, we choose to keep their total number of weights fixed to 2060, i.e., such that the minimal number of hidden units per layer of the DQNs are ten in all instances of the experiment. For a given number of actions, we adapt the size of the two hidden layers of the networks such that both layers have the same number of hidden units. We chose ADAM as an optimizer with a learning rate of $\alpha = 0.01$. The training batches of the main network contain ten samples and the replay memory has a capacity of 2000 interactions. The target network is updated every 100 timesteps while the experience replay memory is updated (along with the main network) every ten timesteps (in a first-in-first-out manner).

For the simulations of the circular GridWorld experiment, in order to have a fair comparison of performance between DEBNs and DQNs, we choose to keep their total number of weights fixed to 2660, i.e., such that the minimal number of hidden units per layer of the DQNs are ten in all instances of the experiment. For a given number of actions, we adapt the size of the two hidden layers of the networks such that both layers have the same number of hidden units. We choose ADAM as an optimizer with a learning rate of $\alpha = 0.01$. The training batches of the main network contain 100 samples and the replay memory has a capacity of 5000 interactions. The target network is updated every 500 timesteps while the experience replay memory is updated (along with the main network) every ten timesteps (in a first-in-first-out manner). The agent's policy is a softmax policy and for the β parameter we choose a linear schedule that starts with a value of 1 and ends with a value of 100.

---

[1] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, Nature **549**, 195 (2017).

[2] V. Dunjko and H. J. Briegel, Machine learning & artificial intelligence in the quantum domain: A review of recent progress, Rep. Prog. Phys. **81**, 074001 (2018).

[3] C. Ciliberto, M. Herbster, A. D. Ialongo, M. Pontil, A. Rocchetto, S. Severini, and L. Wossnig, Quantum machine learning: A classical perspective, Proc. R. Soc. A **474**, 20170551 (2018).

[4] M. Schuld, I. Sinayskiy, and F. Petruccione, An introduction to quantum machine learning, Contemp. Phys. **56**, 172 (2015).

[5] J. Adcock, E. Allen, M. Day, S. Frick, J. Hinchliff, M. Johnson, S. Morley-Short, S. Pallister, A. Price, and S. Stanisic, Advances in quantum machine learning, arXiv:1512.02900 (2015).

[6] M. Schuld, *Supervised Learning with Quantum Computers* (Springer, Cham, 2018).

[7] H. Neven, V. S. Denchev, G. Rose, and W. G. Macready, Training a binary classifier with the quantum adiabatic algorithm, arXiv:0811.0416 (2008).

[8] S. Lloyd, S. Garnerone, and P. Zanardi, Quantum algorithms for topological and geometric analysis of data, Nat. Commun. **7**, 1 (2016).

[9] P. Rebentrost, M. Mohseni, and S. Lloyd, Quantum Support Vector Machine for Big Data Classification, Phys. Rev. Lett. **113**, 130503 (2014).

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

[11] N. Wiebe, D. Braun, and S. Lloyd, Quantum Algorithm for Data Fitting, Phys. Rev. Lett. **109**, 050505 (2012).

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

[13] M. H. Amin, E. Andriyash, J. Rolfe, B. Kulchytskyy, and R. Melko, Quantum Boltzmann Machine, Phys. Rev. X **8**, 021050 (2018).

[14] M. Schuld, A. Bocharov, K. M. Svore, and N. Wiebe, Circuit-centric quantum classifiers, Phys. Rev. A **101**, 032308 (2020).

[15] I. Kerenidis, J. Landman, A. Luongo, and A. Prakash, in *Advances in Neural Information Processing Systems* (Curran Associates, Red Hook, 2019), p. 4134.

[16] B. T. Kiani, A. Villanyi, and S. Lloyd, Quantum medical imaging algorithms, arXiv:2004.02036 (2020).

[17] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, Mastering the game of go without human knowledge, Nature **550**, 354 (2017).

[18] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, A general reinforcement learning algorithm that masters chess, shogi, and go through self-play, Science **362**, 1140 (2018).

[19] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani, and P. Pérez, Deep reinforcement learning for autonomous driving: A survey, arXiv:2002.00444 (2020).

[20] M. Popova, O. Isayev, and A. Tropsha, Deep reinforcement learning for de novo drug design, Sci. Adv. **4**, eaap7885 (2018).

[21] A. A. Melnikov, H. P. Nautrup, M. Krenn, V. Dunjko, M. Tiersch, A. Zeilinger, and H. J. Briegel, Active learning machine learns to create new quantum experiments, Proc. Natl. Acad. Sci. **115**, 1221 (2018).

[22] A. A. Melnikov, P. Sekatski, and N. Sangouard, Setting up Experimental Bell Tests with Reinforcement Learning, Phys. Rev. Lett. **125**, 160401 (2020).

[23] M. August and J. M. Hernández-Lobato, in *International Conference on High Performance Computing* (Springer, Cham, 2018), p. 591.

[24] T. Fösel, P. Tighineanu, T. Weiss, and F. Marquardt, Reinforcement Learning with Neural Networks for Quantum Feedback, Phys. Rev. X **8**, 031084 (2018).

[25] Z. T. Wang, Y. Ashida, and M. Ueda, Deep Reinforcement Learning Control of Quantum Cartpoles, Phys. Rev. Lett. **125**, 100401 (2020).

[26] L. J. Fiderer, J. Schuff, and D. Braun, Neural-network heuristics for adaptive bayesian quantum estimation, arXiv:2003.02183 (2020).

[27] J. Yao, M. Bukov, and L. Lin, Policy gradient based quantum approximate optimization algorithm, arXiv:2002.01068 (2020).

[28] P. Andreasson, J. Johansson, S. Liljestrand, and M. Granath, Quantum error correction for the toric code using deep reinforcement learning, Quantum **3**, 183 (2019).

[29] H. P. Nautrup, N. Delfosse, V. Dunjko, H. J. Briegel, and N. Friis, Optimizing quantum error correction codes with reinforcement learning, Quantum **3**, 215 (2019).

[30] R. Sweke, M. Kesselring, E. Van Nieuwenburg, and J. Eisert, Reinforcement learning decoders for fault-tolerant quantum computation, Machine Learning: Sci. Technol. **2**, 025005 (2020).

[31] G. Dulac-Arnold, D. Mankowitz, and T. Hester, Challenges of real-world reinforcement learning, arXiv:1904.12901 (2019).

[32] Y. Lecun, S. Chopra, R. Hadsell, M. A. Ranzato, and F. J. Huang, in *Predicting Structured Data* (MIT Press, Cambridge, 2006).

[33] H. J. Kappen, Learning quantum models from quantum or classical data, J. Phys. A: Math. Theor. **53**, 214001 (2020).

[34] N. Wiebe, A. Kapoor, and K. M. Svore, Quantum deep learning, Quantum Inf. Comput. **16**, 541 (2016).

[35] N. Wiebe and L. Wossnig, Generative training of quantum Boltzmann machines with hidden units, arXiv:1905.09902 (2019).

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

[37] D. Zhu, N. M. Linke, M. Benedetti, K. A. Landsman, N. H. Nguyen, C. H. Alderete, A. Perdomo-Ortiz, N. Korda, A. Garfoot, C. Brecque, *et al.*, Training of quantum circuits on a hybrid quantum computer, Sci. Adv. **5**, eaaw9918 (2019).

[38] W. Huggins, P. Patil, B. Mitchell, K. B. Whaley, and E. M. Stoudenmire, Towards quantum machine learning with tensor networks, Quantum Sci. Technol. **4**, 024001 (2019).

[39] M. Kieferová and N. Wiebe, Tomography and generative training with quantum Boltzmann machines, Phys. Rev. A **96**, 062327 (2017).

[40] Y. Du, M.-H. Hsieh, T. Liu, and D. Tao, The expressive power of parameterized quantum circuits, arXiv:1810.11922 (2018).

[41] X. Gao, Z.-Y. Zhang, and L.-M. Duan, A quantum machine learning algorithm based on generative models, Sci. Adv. **4**, eaat9004 (2018).

[42] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller,

A. K. Fidjeland, G. Ostrovski, *et al.*, Human-level control through deep reinforcement learning, Nature **518**, 529 (2015).

[43] B. Sallans and G. E. Hinton, Reinforcement learning with factored states and actions, J. Mach. Learn. Res. **5**, 1063 (2004).

[44] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction* (The MIT Press, Cambridge, 1998).

[45] H. J. Briegel and G. De las Cuevas, Projective simulation for artificial intelligence, Sci. Rep. **2**, 400 (2012).

[46] Note that, in standard VBMs, states and actions are part of a mathematical object describing the agent's task environment: a Markov decision process. In PS, although similar to VBMs in its two-layer form, the perspective is different and more agent centered: representations of states and actions have a separate status as entities in the agent's memory, they can be created, erased, or modified as part of the agent's dynamics of memory processing and deliberation.

[47] M. Bukov, A. G. R. Day, D. Sels, P. Weinberg, A. Polkovnikov, and P. Mehta, Reinforcement Learning in Different Phases of Quantum Control, Phys. Rev. X **8**, 031086 (2018).

[48] M. Bukov, Reinforcement learning for autonomous preparation of Floquet-engineered states: Inverting the quantum kapitza oscillator, Phys. Rev. B **98**, 224305 (2018).

[49] P. Palittapongarnpim, P. Wittek, and B. C. Sanders, in *24th European Symposium on Artificial Neural Networks, Bruges, April 27–29, 2016* (Ciaco, Louvain-la-Neuve, 2016), p. 327.

[50] S. S. Vedaie, P. Palittapongarnpim, and B. C. Sanders, in *2018 IEEE Photonics Society Summer Topical Meeting Series (SUM)* (IEEE, Piscataway, 2018), p. 163.

[51] R. Porotti, D. Tamascelli, M. Restelli, and E. Prati, Coherent transport of quantum states by deep reinforcement learning, Commun. Phys. **2**, 61 (2019).

[52] J. Wallnöfer, A. A. Melnikov, W. Dür, and H. J. Briegel, Machine learning for long-distance quantum communication, PRX Quantum **1**, 010301 (2020).

[53] H. Xu, J. Li, L. Liu, Y. Wang, H. Yuan, and X. Wang, Transferable control for quantum parameter estimation through reinforcement learning, arXiv:1904.11298 (2019).

[54] M. Dalgaard, F. Motzoi, J. J. Sorensen, and J. Sherson, Global optimization of quantum dynamics with alphazero deep exploration, npj Quantum Inf. **6**, 1 (2020).

[55] P. Ronagh, Quantum algorithms for solving dynamic programming problems, arXiv:1906.02229 (2019).

[56] V. Dunjko, J. M. Taylor, and H. J. Briegel, in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (IEEE, Piscataway, 2017), p. 282.

[57] L. K. Grover, in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing* (Association for Computing Machinery, New York, 1996), p. 212.

[58] G. Brassard, M. Mosca, and A. Tapp, Quantum amplitude amplification and estimation, Contemp. Math. **305**, 53 (2002).

[59] V. Dunjko, J. M. Taylor, and H. J. Briegel, Quantum-Enhanced Machine Learning, Phys. Rev. Lett. **117**, 130501 (2016).

[60] V. Dunjko, J. M. Taylor, and H. J. Briegel, Framework for learning agents in quantum environments, arXiv:1507.08482 (2015).

[61] A. Cornelissen, *Quantum Gradient Estimation and its Application to Quantum Reinforcement Learning* (TU Delft Library, Delft, 2018).

[62] V. Dunjko, Y.-K. Liu, X. Wu, and J. M. Taylor, Exponential improvements for quantum-accessible reinforcement learning, arXiv:1710.11160 (2017).

[63] G. D. Paparo, V. Dunjko, A. Makmal, M. A. Martin-Delgado, and H. J. Briegel, Quantum Speedup for Active Learning Agents, Phys. Rev. X **4**, 031002 (2014).

[64] D. Crawford, A. Levit, N. Ghadermarzy, J. S. Oberoi, and P. Ronagh, Reinforcement learning using quantum Boltzmann machines, Quantum Inf. Comput. **18**, 51 (2018).

[65] A. Levit, D. Crawford, N. Ghadermarzy, J. S. Oberoi, E. Zahedinejad, and P. Ronagh, Free energy-based reinforcement learning using a quantum processor, arXiv:1706.00074 (2017).

[66] F. Neukart, D. Von Dollen, C. Seidel, and G. Compostella, Quantum-enhanced reinforcement learning for finite-episode games with discrete state spaces, Front. Phys. **5**, 71 (2018).

[67] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, in *Thirty-Second AAAI Conference on Artificial Intelligence* (AAAI Press, Palo Alto, 2018).

[68] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, in *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (JMLR. org, Cambridge, 2017), p. 1352.

[69] G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, and B. Coppin, Deep reinforcement learning in large discrete action spaces, arXiv:1512.07679 (2015).

[70] G. E. Hinton and T. J. Sejnowski, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Citeseer, Silver Spring, 1983), p. 448.

[71] E. Ising, Contribution to the theory of ferromagnetism, Z. Phys. **31**, 253 (1925).

[72] N. Heess, D. Silver, and Y. W. Teh, in *Proceedings of the Tenth European Workshop on Reinforcement Learning* Vol. 24 (PMLR, Cambridge, 2013), p. 45.

[73] M. Otsuka, J. Yoshimoto, and K. Doya, in *ESANN* (D-side, Evere, 2010).

[74] S. Elfwing, E. Uchibe, and K. Doya, From free energy to expected energy: Improving energy-based value function approximation in reinforcement learning, Neural Netw. **84**, 17 (2016).

[75] J. Martens, A. Chattopadhya, T. Pitassi, and R. Zemel, in *Advances in Neural Information Processing Systems* (Curran Associates, Red Hook, 2013), p. 2877.

[76] J. Ngiam, Z. Chen, P. W. Koh, and A. Y. Ng, in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)* (International Machine Learning Society, Madison, 2011), p. 1105.

[77] Y. Du and I. Mordatch, in *Advances in Neural Information Processing Systems*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc., 2019), Vol. 32, p. 3608.

[78] P. M. Long and R. Servedio, in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (Springer, Berlin, 2010), p. 703.

[79] G. E. Hinton, in *Neural Networks: Tricks of the Trade* (Springer, Berlin, 2012), p. 599.

[80] Under maximum-likelihood learning. In practice, a different gradient is used to train RBMs, derived by the *contrastive divergence* method [145,146]. Even though this "gradient" is computed more efficiently, it does not correspond to the gradient of any true objective function [147], it can lead to very large approximation errors in the learned distributions [148–150] and cannot be used to train full (all-to-all connected) Boltzmann machines [34].

[81] The approximation comes in SARSA and Q-learning from considering $Q(s^{(t+1)}, a)$ constant with respect to θ, which is an assumption made in temporal difference learning.

[82] N. Le Roux and Y. Bengio, Representational power of restricted Boltzmann machines and deep belief networks, Neural Comput. **20**, 1631 (2008).

[83] G. F. Montúfar, J. Rauh, and N. Ay, in *Advances in Neural Information Processing Systems* (Curran Associates, Red Hook, 2011), p. 415.

[84] V. Vapnik and A. Y. Chervonenkis, On the uniform convergence of relative frequencies of events to their probabilities, Measures Complexity **16**, 11 (1971).

[85] Originally used to quantify the expressive power of classifiers, its definition can also be extended to regression models.

[86] N. Harvey, C. Liaw, and A. Mehrabian, in *Conference on Learning Theory* (PMLR, Amsterdam, 2017), p. 1064.

[87] S. Liang and R. Srikant, Why deep neural networks for function approximation? arXiv:1610.04161 (2016).

[88] R. Salakhutdinov and G. Hinton, in *Artificial in Telligence and Statistics* (PMLR, Clearwater Beach, 2009), p. 448.

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

[90] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning representations by back-propagating errors, Cognitive Model. **5**, 1 (1986).

[91] J. N. Tsitsiklis and B. Van Roy, in *Advances in Neural Information Processing Systems* (Kaufmann, San Mateo, 1997), p. 1075.

[92] J. Fu, A. Kumar, M. Soh, and S. Levine, *Diagnosing Bottlenecks in Deep Q-learning Algorithms* (PMLR, Long Beach, California, USA, 2019), p. 2021.

[93] Note that, combined with the policy-sampling effect, we also have that the target function is moving, as the policy of the agent changes over training.

[94] L. M. Trenkwalder, S. Jerbi, and H. Poulsen Nautrup, *Quantum-Enhanceable Deep Reinforcement Learning* (GitHub, 2020). https://github.com/LeaMarion/quantum-enhanceable-deep-rl.

[95] A. A. Melnikov, A. Makmal, and H. J. Briegel, Benchmarking projective simulation in navigation problems, IEEE Access **6**, 64639 (2018).

[96] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, Openai gym, arXiv:1606.01540 (2016).

[97] A. W. Harrow and A. Y. Wei, in *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (SIAM, Philadelphia, 2020), p. 193.

[98] M.-H. Yung and A. Aspuru-Guzik, A quantum–quantum metropolis algorithm, Proc. Natl. Acad. Sci. **109**, 754 (2012).

[99] J. Van Apeldoorn, A. Gilyén, S. Gribling, and R. de Wolf, Quantum sdp-solvers: Better upper and lower bounds, Quantum **4**, 230 (2020).

[100] A. N. Chowdhury, G. H. Low, and N. Wiebe, A variational quantum algorithm for preparing quantum gibbs states, arXiv:2002.00055 (2020).

[101] J. Wu and T. H. Hsieh, Variational Thermal Quantum Simulation via Thermofield Double States, Phys. Rev. Lett. **123**, 220502 (2019).

[102] Y. Wang, G. Li, and X. Wang, Variational quantum gibbs state preparation with a truncated taylor series, arXiv:2005.08797 (2020).

[103] D. J. Aldous, Some inequalities for reversible markov chains, J. London Math. Soc. **2**, 564 (1982).

[104] N. Srivastava and R. R. Salakhutdinov, in *Advances in Neural Information Processing Systems* (Curran Associates, Red Hook, 2012), p. 2222.

[105] X. Gao and L.-M. Duan, Efficient representation of quantum many-body states with deep neural networks, Nat. Commun. **8**, 1 (2017).

[106] D. W. Berry and A. M. Childs, Black-box hamiltonian simulation and unitary implementation, arXiv:0910.4157 (2009).

[107] For QBMs, we are as well interested in Gibbs distributions for configurations $\{x\}$ that are not part of the eigenbasis of H, leading to a different expression of Eq. (15) that involves the eigendecomposition of H. For classical Hamiltonians (e.g., DBMs), these eigenvectors are simply computational basis states.

[108] Actually, the general goal of purified Gibbs state preparation is to prepare a state $|\phi\rangle_A B$ such that $\text{Tr}_B|\phi\rangle\langle\phi|_A B \approx |\psi\rangle\langle\psi| = e^{-\beta H}/\text{Tr} e^{-\beta H}$. The state of Eq. (16) is a particular instance, called a *coherent encoding*, which can be more useful than a general purified Gibbs state, e.g., when estimating the partition function of the associated Gibbs distribution, Eq. (15) [122].

[109] C. Durr and P. Hoyer, A quantum algorithm for finding the minimum, quant-ph/9607014 (1996).

[110] G. H. Low and I. L. Chuang, Hamiltonian simulation by qubitization, Quantum **3**, 163 (2019).

[111] T. J. Yoder, G. H. Low, and I. L. Chuang, Fixed-Point Quantum Search with an Optimal Number of Queries, Phys. Rev. Lett. **113**, 210501 (2014).

[112] M. Szegedy, in *FOC'04: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science* (IEEE, Los Alamitos, 2004), p. 32.

[113] F. Magniez, A. Nayak, J. Roland, and M. Santha, Search via quantum walk, SIAM J. Comput. **40**, 142 (2011).

[114] P. Wocjan and A. Abeyesinghe, Speedup via quantum sampling, Phys. Rev. A **78**, 042336 (2008).

[115] D. Orsucci, H. J. Briegel, and V. Dunjko, Faster quantum mixing for slowly evolving sequences of markov chains, Quantum **2**, 105 (2018).

[116] This is also the reason why we deem classical Gibbs sampling to be inapplicable to QBMs in Table I, and why exact sampling has larger complexity for QBMs than DBMs: one should additionally diagonalize the QBM Hamiltonian to be able to exponentiate it.

[117] K. Temme, T. J. Osborne, K. G. Vollbrecht, D. Poulin, and F. Verstraete, Quantum metropolis sampling, Nature **471,** 87 (2011).

[118] R. D. Somma, S. Boixo, H. Barnum, and E. Knill, Quantum Simulations of Classical Annealing Processes, Phys. Rev. Lett. **101,** 130504 (2008).

[119] Y. Subaşı, L. Cincio, and P. J. Coles, Entanglement spectroscopy with a depth-two quantum circuit, J. Phys. A: Math. Theor. **52,** 044001 (2019).

[120] J. Nocedal and S. Wright, *Numerical Optimization* (Springer Science & Business Media, New York, 2006).

[121] M. R. Jerrum, L. G. Valiant, and V. V. Vazirani, Random generation of combinatorial structures from a uniform distribution, Theor. Comput. Sci. **43,** 169 (1986).

[122] P. Wocjan, C.-F. Chiang, A. Abeyesinghe, and D. Nagaj, Quantum speed-up for approximating partition functions, arXiv:0811.0596 (2008).

[123] E. Campbell, A. Khurana, and A. Montanaro, Applying quantum algorithms to constraint satisfaction problems, Quantum **3,** 167 (2019).

[124] R. Babbush, J. McClean, C. Gidney, S. Boixo, and H. Neven, Focus beyond quadratic speedups for error-corrected quantum advantage, arXiv:2011.04149 (2020).

[125] S. Endo, S. C. Benjamin, and Y. Li, Practical Quantum Error Mitigation for Near-Future Applications, Phys. Rev. X **8,** 031027 (2018).

[126] B. Koczor, Exponential error suppression for near-term quantum devices, arXiv:2011.05942 (2020).

[127] R. LaRose and B. Coyle, Robust data encodings for quantum classifiers, arXiv:2003.01695 (2020).

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

[129] S. Y.-C. Chen, C.-H. H. Yang, J. Qi, P.-Y. Chen, X. Ma, and H.-S. Goan, Variational quantum circuits for deep reinforcement learning, IEEE Access **8,** 141007 (2020).

[130] M. Schuld, R. Sweke, and J. J. Meyer, The effect of data encoding on the expressive power of variational quantum machine learning models, arXiv:2008.08605 (2020).

[131] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, Supervised learning with quantum-enhanced feature spaces, Nature **567,** 209 (2019).

[132] M. Schuld and N. Killoran, Quantum Machine Learning in Feature Hilbert Spaces, Phys. Rev. Lett. **122,** 040504 (2019).

[133] S. Lloyd, M. Schuld, A. Ijaz, J. Izaac, and N. Killoran, Quantum embeddings for machine learning, arXiv:2001.03622 (2020).

[134] I. A. Murray, Ph.D. thesis, University of London, 2007.

[135] Y. Iba, N. Saito, and A. Kitajima, Multicanonical MCMC for sampling rare events: An illustrative review, Ann. Inst. Stat. Math. **66,** 611 (2014).

[136] Intuitively, this has to do with the fact that all eigenvalues modulus of P are strictly less than 1 (except for the eigenvalue 1 associated to the stationary distribution), and repeated applications of P make its (noneigenvalue 1) eigenspaces fade away with a geometric rate corresponding to their eigenvalue modulus. The slowest fading rate is given by the second largest eigenvalue modulus.

[137] W. K. Hastings, Monte Carlo sampling methods using Markov chains and their applications, Biometrika **57,** 97 (1970).

[138] S. Geman and D. Geman, Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, IEEE Trans. Pattern Anal. Mach. Intelligence **6,** 721 (1984).

[139] Note that the asterisk on P^* does not indicate complex conjugation.

[140] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, A learning algorithm for Boltzmann machines, Cogn. Sci. **9,** 147 (1985).

[141] A. Fischer and C. Igel, in *Iberoamerican Congress on Pattern Recognition* (Springer, Berlin, 2012), p. 14.

[142] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, 2010).

[143] V. Dunjko, N. Friis, and H. J. Briegel, Quantum-enhanced deliberation of learning agents using trapped ions, New J. Phys. **17,** 023006 (2015).

[144] L. Grover and T. Rudolph, Creating superpositions that correspond to efficiently integrable probability distributions, quant-ph/0208112 (2002).

[145] M. Welling and G. E. Hinton, in *International Conference on Artificial Neural Networks* (Springer, New York, 2002), p. 351.

[146] M. A. Carreira-Perpinan and G. E. Hinton, in *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, edited by R. G. Cowell and Z. Ghahramani (Society for Artificial Intelligence and Statistics, Fort Lauderdale, 2005), Vol. 10, p. 33.

[147] I. Sutskever and T. Tieleman, in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (PMLR, Sardinia, 2010), p. 789.

[148] A. Fischer and C. Igel, Bounding the bias of contrastive divergence learning, Neural Comput. **23,** 664 (2011).

[149] T. Tieleman and G. Hinton, in *Proceedings of the 26th Annual International Conference on Machine Learning* (MIT Press, Cambridge, 2009), p. 1033.

[150] Y. Bengio and O. Delalleau, Justifying and generalizing contrastive divergence, Neural Comput. **21,** 1601 (2009).