

Conditioned quantum-assisted deep generative surrogate for particle-calorimeter interactions

J. Quetzalcoatl Toledo-Marin,^{1,2} Sebastian Gonzalez,^{1,*} Hao Jia,^{3,*} Ian Lu,^{1,*} Deniz Sogutlu,¹ Abhishek Abhishek,⁴
Colin Gay,³ Eric Paquet,² Roger Melko,² Geoffrey C. Fox,⁵ Maximilian Swiatlowski,¹ and Wojciech Fedorko¹

¹*TRIUMF, Vancouver, BC V6T 2A3, Canada*

²*Perimeter Institute for Theoretical Physics, Waterloo, Ontario, N2L 2Y5, Canada*

³*Department of Physics and Astronomy, University of British Columbia,
Vancouver, BC V6T 1Z1, Canada*

⁴*Department of Electrical and Computer Engineering,
University of British Columbia, Vancouver, BC V6T 1Z4, Canada*

⁵*University of Virginia, Computer Science and Biocomplexity Institute,
994 Research Park Blvd, Charlottesville, Virginia, 22911, USA*

(Dated: December 4, 2025)

Particle collisions at accelerators such as the Large Hadron Collider, recorded and analyzed by experiments such as ATLAS and CMS, enable exquisite measurements of the Standard Model and searches for new phenomena. Simulations of collision events at these detectors have played a pivotal role in shaping the design of future experiments and analyzing ongoing ones. However, the quest for accuracy in Large Hadron Collider (LHC) collisions comes at an imposing computational cost, with projections estimating the need for millions of CPU-years annually during the High Luminosity LHC (HL-LHC) run [1]. Simulating a single LHC event with GEANT4 currently devours around 1000 CPU seconds, with simulations of the calorimeter subdetectors in particular imposing substantial computational demands [2]. To address this challenge, we propose a conditioned quantum-assisted deep generative model. Our model integrates a conditioned variational autoencoder (VAE) on the exterior with a conditioned Restricted Boltzmann Machine (RBM) in the latent space, providing enhanced expressiveness compared to conventional VAEs. The RBM nodes and connections are meticulously engineered to enable the use of qubits and couplers on D-Wave’s Pegasus-structured *Advantage* quantum annealer (QA) for sampling. We introduce a novel method for conditioning the quantum-assisted RBM using *flux biases*. By adapting flux bias on D-Wave’s systems, we effectively incorporate the flexibility of classical Restricted Boltzmann Machines (RBMs), as universal approximators for discrete distributions, with the potential speedup and scalability of quantum annealing. We further propose a faster and more robust adaptive mapping to estimate the effective inverse temperature in quantum annealers. The effectiveness of our framework is illustrated using Dataset 2 of the CaloChallenge [3].

I. INTRODUCTION

By the end of the decade, the LHC is expected to begin an upgraded “High Luminosity” phase, which will ultimately increase the collision rate by a factor of 10 higher than the initial design. Increasing the number of collisions will generate more experimental data, enabling the observation of rare processes and increased precision in measurements, furthering our understanding of the universe. The path toward the HL-LHC presents great technological challenges and commensurate innovations to overcome them. Monte Carlo simulations of collision events at the ATLAS experiment have played a key role in the design of future experiments and, particularly, in the analysis of current ones. However, these simulations are computationally intensive, projected to

reach into millions of CPU-years per year during the HL-LHC run [1]. Simulating a single event with GEANT4 [4] in an LHC experiment requires roughly 1000 CPU seconds. The calorimeter simulation is by far dominating the total simulation time [2]. To address this challenge, deep generative models are being developed to act as particle-calorimeter interaction surrogates, with the potential to reduce the simulation overall time by orders of magnitude. The key point to take into consideration is that one particle impacting a calorimeter can lead to thousands of secondary particles, collectively known as showers, to be traced through the detector, while only the total energy deposit per sensitive element (a cell) is actually measured in the experiment. Hence, through the generation of these showers, non-negligible computational resources are being employed in the detailed recording of the path of these particles. The problem being addressed is whether one can bypass the path-tracing step in the simulation and generate the cell energy deposits directly from a set of well-defined parameters (*e.g.*, type of particle, incidence energy, incidence angle,

* These authors contributed equally and are listed in alphabetical order.

etc.) via sampling from a deep generative model.

There is a large and growing body of literature addressing this critical problem via deep generative models. The earliest methods developed to address this challenge were of the kind of Generative Adversarial Networks [5? ?], which are now an integral part of the simulation pipeline [6?] of some experiments. Different deep generative frameworks have been proposed since then, including VAEs [7? ? ?], Normalizing Flows [8?], Transformers [9], Diffusion models [10? ?] and combinations thereof [11?]. A noteworthy development in the field is the CaloChallenge-2022 endeavour [12], which not only catalyzed research efforts but has also enabled better quantitative comparison across different frameworks. Similarly, defining benchmarks and metrics has been a rather active topic of research [13?]. A common feature of these models is the fast generation of showers via sampling on GPUs, but further speed increases may still be possible with alternative computing paradigms. In our work, we develop deep generative models which can be naturally encoded onto Quantum Annealers (QAs), allowing for potentially significant improvements in the speed of shower simulation.

In previous work by this group [14], a proof of concept of a quantum-assisted deep discrete variational autoencoder calorimeter surrogate called CaloQVAE was presented, whereby the performance in synthetic data generation is similar to its classical counterpart. As a follow up, in Ref [15], we improved our framework by re-engineering the encoder and decoder architectures by introducing two-dimensional convolutions as well as replacing the two-partite graph in the RBM with four-partite graph, enabling us to use D-wave’s quantum annealer AdvantageSystem [16]. The main contributions of this paper are: *i)* we condition the prior on specific incident energies which allows us to disentangle the latent space embeddings. Most relevant is that we propose a novel method to enforce this condition scheme on the qubits by means of the QA’s *flux bias* parameters. By adapting flux bias on D-Wave’s systems, we effectively incorporate the flexibility of classical RBM, as universal approximators for discrete distributions, with the potential speedup and scalability of quantum annealing. *ii)* We also propose a novel adaptive method to estimate the effective temperature in the quantum annealer. This method is a simple map with a stable fixed point on the effective temperature, and we show it to be more stable with a faster convergence than the KL-divergence-based method used previously. This contribution is a key methodological advancement that may benefit a wide range of quantum machine learning and sampling applications. *iii)* The new model uses 3D convolutional layers for the encoder and decoder as well as periodic boundaries to account for the cylindrical geometry of the shower, leading to an improved performance when compared to its previous counterpart. As the HL-LHC prepares to generate enormous datasets, rapid and accurate simulations are paramount. Our approach addresses an urgent and multidisciplinary chal-

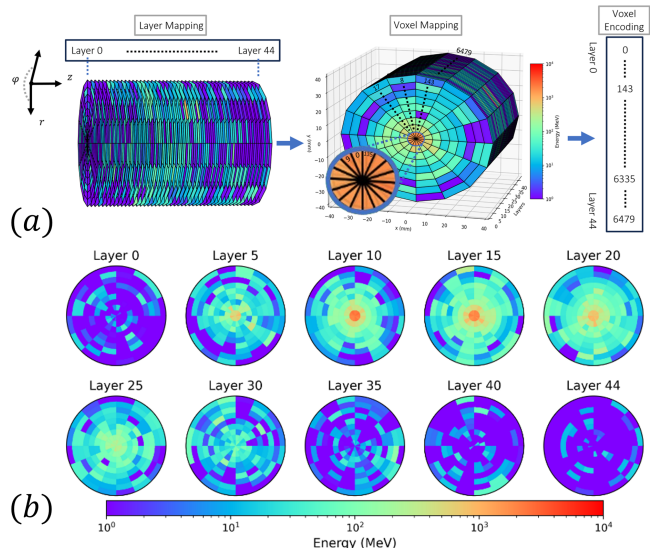


FIG. 1. (a) Calochallenge dataset showers are voxelized using cylindrical coordinates (r, φ, z) such that the showers evolve in the z direction. For any given event, each voxel value corresponds to the energy (MeV) in that vicinity. Dataset 2 contains 100k events and the voxelized cylinder has 45 stacked layers. Each layer has 144 voxels composed of 16 angular bins and 9 radial bins. The data set is parsed onto a 1D vector following the common way to eat a pizza, *i.e.*, grab a slice and start from the inside towards the crust. Each 1D vector has $45 \times 9 \times 16 = 6480$ voxels per each event. (b) Visualization of the voxels in an event in the dataset.

lenge, offering a potential solution that may transform simulation pipelines in high-energy physics while contributing to the broader quantum information and machine learning communities. We illustrate our framework by using Dataset 2 of the CaloChallenge. Henceforth, we refer to our framework as Calo4pQVAE.

The paper is organized as follows: In the Methods section we present the dataset we used, the preprocessing steps employed. This section also introduces the Calo4pQVAE framework, which is detailed in three subsections: the 4p-VAE, where we give a description over the classical framework; the 4-partite RBM where we describe in detail the sampling and training procedure for conditioned and non-conditioned RBM; and in Quantum Annealer we give a brief overview of the motivations behind quantum annealers, explain how they work, and discuss how we incorporate them into our framework, including sampling methods and conditioning techniques. In the Results section we outline the training process of our model and define the criteria used to select the best-performing model. We also showcase the performance of the model using various quantitative and qualitative metrics. We interpret the results and provide a thorough discussion on the current limitations of our approach, propose next steps for improvement, and address the technical challenges ahead.

II. METHODS

The CaloChallenge 2022 [3] comprises three distinct datasets, each designed to facilitate research and testing in the field of calorimeter simulations. All three datasets are derived from GEANT4 simulations. The first dataset, referred to as *Dataset 1*, represent photon and charged pion showers within a specified η [17] range. The dataset covers a discrete range of incident energies, ranging from 256 MeV to 4 TeV, logarithmically spaced out evenly with varying sizes at higher energies. The calorimeter geometry is that of the ATLAS detector.

In the case of *Dataset 2* it is comprised by two files containing 100,000 GEANT4-simulated electron showers each. These showers encompass a wide energy range, spanning from 1 GeV to 1 TeV sampled from a log-uniform distribution. The detector geometry features a concentric cylinder structure with 45 layers as shown in Fig. 1, each consisting of active (silicon) and passive (tungsten) material. The dataset is characterized by high granularity, with $45 \times 16 \times 9 = 6,480$ voxels. The cylinder is 36 radiation lengths deep and spans nine Molière radii in diameter [3]. Lastly, *Dataset 3* contains four files, each housing 50,000 GEANT4-simulated electron showers. Similar to Dataset 2, these showers encompass energies ranging from 1 GeV to 1 TeV. The detector geometry remains consistent with Dataset 2 but exhibits significantly higher granularity, boasting 18 radial and 50 angular bins per layer, totaling $45 \times 50 \times 18 = 40,500$ voxels.

These datasets collectively offer a comprehensive resource for researchers interested in the development and evaluation of generative models and simulations within the field of calorimetry in High Energy experiments. The datasets are publicly available and accessible via [18? ?]. For our results, we consider Dataset 2 and we leave the testing of our framework using the remaining datasets for future work.

A. Data preprocessing

Before feeding the shower and incident energy data to the model, we apply several transformations to the shower and the incident particle energy *on-the-fly*. Given an event shower, \mathbf{v} , and corresponding incident energy, e , we first reduce the voxel energy, v_i , per event by dividing it by the incident energy, e , *viz.* $E_i = v_i/e$. Notice that $E_i \in [0, 1]$, where the left and right bounds correspond to when the voxel energy is zero and equal to the incident energy, respectively. To remove the strict bounds, we define $u_i = \delta + (1 - 2\delta)E_i$, where $\delta = 10^{-7}$, to prevent discontinuities during the *logit* transformation. Specifically, we use the transformation $x_i = \ln u_i / (1 - u_i) - \ln \delta / (1 - \delta)$, where the second term preserves the zero values in the transformed variable, *i.e.*, when the voxel energy is zero, $v_i = 0$, the transformed variable $x_i = 0$.

The incident energy is used as a conditioning paramete-

ter and we transform it by applying a logarithmic function followed by scaling it between 0 and 1. These transformations have been used before in the same context [8? ?]. However, in our case, we modify the process to preserve the zeroes in the transformed variables, x_i , and therefore omit the last step of standardizing the new variables, in contrast with other approaches.

B. 4p-QVAE

The Calo4pQVAE can be conceptualized as a variational autoencoder (VAE) with a restricted Boltzmann machine (RBM) as its prior. The modularity of our framework allows for the replacement of any component, such as the encoder, decoder, or the RBM, facilitating flexibility in its configuration [19]. The encoder, also referred to as the *approximating posterior*, is denoted as $q_\phi(\mathbf{z}|\mathbf{x}, e)$, while the latent space prior distribution is denoted as $p_\theta(\mathbf{z})$. The decoder, responsible for generating data from the latent variables, is represented as $p_\theta(\mathbf{x}|\mathbf{z}, e)$. We train the model to generate synthetic shower events given a specific incidence energy. In other words, we are interested in finding $p_\theta(\mathbf{x}|\mathbf{z}, e)$, such that $\int p_\theta(\mathbf{x}|\mathbf{z}, e)p_\theta(\mathbf{z})d\mathbf{z}$ matches the empirical dataset distribution. We use ϕ to denote the encoder parameters, θ for the prior and decoder parameters, and \mathbf{z} the latent space vector. We denote as \mathbf{x} a one-dimensional vector, such that $\mathbf{x} \in R^n$, and we say each element x_i contains the energy measured at the i th voxel for that specific instance (or event), defined by the bijective transformation described in the previous subsection, while e is incidence energy of the event. During training, the model takes as input an instance of \mathbf{x} and e . The input data is encoded into the latent space via the encoder. One key difference between VAEs and autoencoders is that in the latter, the same input generally yields the same encoded representation, provided the encoder does not use stochastic filters, such as dropout. In contrast, VAEs generate a different encoded representation with each pass of the same input. This is because the output of a VAE encoder consists of the parameters of a distribution from which the encoded data is sampled. The encoded data is then passed through the decoder which reconstructs the shower event vector, $\hat{\mathbf{x}}$. The incident particle energy is the label of the event and conditions the encoder and decoder, as depicted in Fig. 3. To condition the encoder and decoder with the incidence energy, we tested two different methods: simple concatenation of the label with the one-dimensional energy per voxel vector, and positional encoding similar to the techniques used by Meta [20] and Google [21]. Despite experimenting with these different encoding schemes, we observed no significant difference in performance. Therefore, we opted to use the simpler concatenation method henceforth and leave the positional encoding methods for future work when dealing with a greater number of features in the dataset.

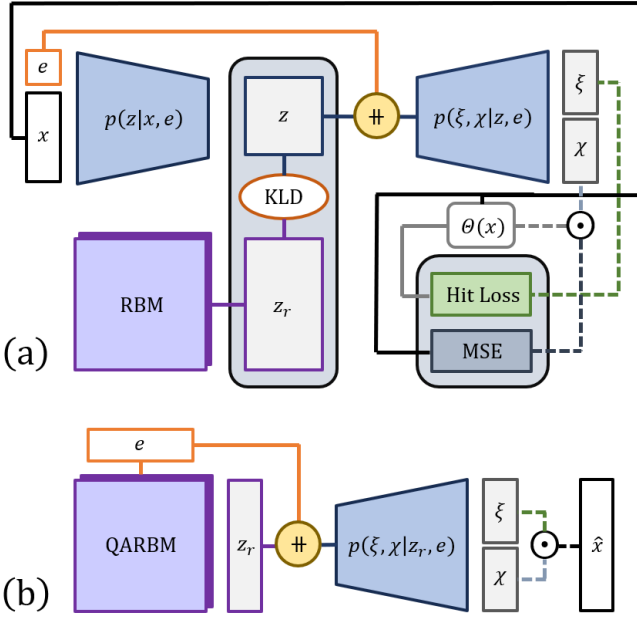


FIG. 2. Sketch of Calo4pQVAE. **(a)** The input data is composed by the energy per voxel, x and the incidence energy, e . During training, the data flows through the encoder gets encoded into a latent space, z , it then goes through the decoder and generates a reconstruction of the voxels per energy, while the incidence energy is the label of the event and conditions the encoder and decoder. The decoder outputs the activation vector, χ and the hits vector ξ . The model is trained via the optimization of the mean squared error between the input shower and the reconstructed shower, the binary cross entropy (hit loss) between the hits vector and the input shower and the Kullback-Liebler divergence which is composed by the entropy of the encoded sample and the restricted Boltzmann machine log-likelihood. **(b)** For inference, we sample from the RBM or the QA conditioned to an incidence energy, afterwards the sample goes through the decoder to generate a shower.

The encoder architecture uses hierarchy levels, as described in [19]. The encoder is composed by three sub-encoders. The first generates one-fourth of the encoded data, which is then fed to the second sub-encoder along with the input data to generate another quarter of the encoded data. This process is repeated with the third sub-encoder, each time integrating all previously generated data and the original input to produce the final encoded output, as depicted in Fig. 3. The purpose of these hierarchy levels is to enforce conditional relationships among latent units by introducing conditioning among latent nodes, *viz.*

$$\begin{aligned}
 q_\phi(\mathbf{z}|\mathbf{x}) &\equiv q_\phi(\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4|\mathbf{x}) \\
 &= q_\phi^{(1)}(\mathbf{z}_1|\mathbf{x}) \prod_{\alpha=2}^4 q_\phi^{(\alpha)}(\mathbf{z}_\alpha|\{\mathbf{z}_i\}_{i=1}^{\alpha-1}, \mathbf{x})
 \end{aligned} \tag{1}$$

The choice of three sub-encoders is designed on mimicking the connections between the four partitions in la-

tent space. Our results suggest that this hierarchical approach indeed fosters correlations between latent units, leading to a Boltzmann-like distribution. Additionally, this hierarchical structure introduces multiple paths for gradient backpropagation, similar to residual networks [22], enhancing the model’s learning capability. The fourth partition conditions the RBM and consists on a binary representation of the incident particle energy, which we describe in detail in Appendix G.

Besides the reconstruction of the event, the encoded data is also used to train the 4-partite RBM via the Kullback-Liebler (KL) divergence. As in traditional VAEs (full derivations can be found in Ref. [23] and in Appendix A), we optimize the *evidence lower bound* (ELBO), $\mathcal{L}_{\phi, \theta}(\mathbf{x})$, to train the Calo4pQVAE. Explicitly, the ELBO function is:

$$\mathcal{L}_{\phi, \theta}(\mathbf{x}) = \langle \ln p_\theta(\mathbf{x}|\mathbf{z}) \rangle_{q_\phi(\mathbf{z}|\mathbf{x})} - \langle \ln \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z})} \rangle_{q_\phi(\mathbf{z}|\mathbf{x})} \tag{2}$$

The first term in Eq. (2) represents the reconstruction accuracy and the second term corresponds to the KL divergence. To define a functional form for $p_\theta(\hat{\mathbf{x}}|\mathbf{z})$, certain assumptions about the distribution are typically made. A common assumption is that the likelihood of the reconstruction is Gaussian distributed, which simplifies to optimizing the mean squared error (MSE).

In the case of calorimeter data, one important aspect is differentiating between voxels that are hit ($x_i \neq 0$) from those that are not hit ($x_j = 0$) in a given event. Accurately training deep learning models to produce zero values can be challenging, as typical activation functions for regressions struggle to yield consistent zero outputs. Here we tackle the challenge by separating zero values from non-zero values by factorizing the data vector \mathbf{x} into two components, *i.e.*, $\mathbf{x} = \boldsymbol{\chi} \odot \boldsymbol{\xi}$ where $\boldsymbol{\chi} \in \mathbb{R}^n$ represents the continuous energy values of the hits and $\boldsymbol{\xi}_i = \Theta(x_i)$ is a binary vector indicating the presence of hits, with $\Theta(\bullet)$ representing the Heaviside function.

Next, we consider a joint probability $p_\theta(\boldsymbol{\chi}, \boldsymbol{\xi}|\mathbf{z})$ as the probability of the voxels being hit according to $\boldsymbol{\xi}$ and energy $\boldsymbol{\chi}$. We can express the joint probability as $p_\theta(\boldsymbol{\chi}, \boldsymbol{\xi}|\mathbf{z}) = p_\theta(\boldsymbol{\chi}|\boldsymbol{\xi}, \mathbf{z})p_\theta(\boldsymbol{\xi}|\mathbf{z})$. We model the event hitting probability, $p_\theta(\boldsymbol{\xi}|\mathbf{z})$, as a Bernoulli distribution $\prod_{i=1}^n p_{\xi_i}^{\xi_i} (1 - p_{\xi_i})^{1-\xi_i}$. The number of particles in the electromagnetic shower follows approximately a Poisson distribution. Furthermore, via the saddle point approximation, for large number of particles in the shower the multivariate Poisson distribution becomes a multivariate Gaussian distribution with the mean equal to the variance. One can show that the variables $\{\chi_i\}_{i=1}^n$ are also approximately Gaussian distributed provided $v_i/e \ll 1$ (see App. H). In the present paper, we assume a variance equal to unity in our generative model, as we observed better performance for this choice. Hence, the joint dis-

tribution $p(\boldsymbol{\chi}, \boldsymbol{\xi}|\mathbf{z})$ can be formally expressed as

$$p_\theta(\boldsymbol{\chi}, \boldsymbol{\xi}|\mathbf{z}) = \prod_{i=1}^n \left(\xi_i \frac{1}{\sqrt{2\pi}} e^{-\frac{(\chi_i - x_i)^2}{2}} + (1 - \xi_i) \delta(\chi_i) \right) p_{\xi_i}^{\xi_i} (1 - p_{\xi_i})^{1 - \xi_i}. \quad (3)$$

Maximizing the log-likelihood of $p_\theta(\boldsymbol{\chi}, \boldsymbol{\xi}|\mathbf{z})$ is rather difficult due to the factor containing the Dirac delta function. It may well be possible to replace the Dirac delta function with a smooth parameterized function. Yet, there is a simpler way where we instead mask $\boldsymbol{\chi}$ with $\boldsymbol{\xi}$, in which case we can neglect the term containing the Dirac delta function. The key point to stress here is that by means of the mask we split the tasks between generating zeroes and non-zeroes, and generating the voxel energy via $\boldsymbol{\xi}$ and $\boldsymbol{\chi}$, respectively. Therefore, the joint distribution in Eq. (3) becomes [24]:

$$p_\theta(\boldsymbol{\chi}, \boldsymbol{\xi}|\mathbf{z}) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi}} e^{-\frac{(\chi_i \cdot \xi_i - x_i)^2}{2}} p_{\xi_i}^{\xi_i} (1 - p_{\xi_i})^{1 - \xi_i}. \quad (4)$$

The reconstruction term transforms to

$$\langle \ln p_\theta(\boldsymbol{\chi}, \boldsymbol{\xi}|\mathbf{z}) \rangle_{q_\phi(\mathbf{z}|\mathbf{x})} = \left\langle \sum_{i=1}^n \left[-(\chi_i \cdot \xi_i - x_i)^2 + \xi_i \ln(p_{\xi_i}) + (1 - \xi_i) \ln(1 - p_{\xi_i}) + \text{const} \right] \right\rangle_{q_\phi(\mathbf{z}|\mathbf{x})}. \quad (5)$$

While Eq. (4) outlines the functional structure of the reconstruction of our generative model, the formulation of loss function requires further consideration. Specifically, there are three key aspects to address. First, the Bernoulli-distributed hits vector is designed to capture the hits distribution in the dataset, yet binary entropy alone does not adequately represent this. Next, the MSE term in Eq. (5) is biased towards low energy when the hits vector reconstruction, $\boldsymbol{\xi}$, incorrectly predicts values during training. To tackle both concerns, we replace $\xi \rightarrow \Theta(x_i)$, which converts the binary entropy into a binary cross entropy (BCE). The third consideration involves how BCE penalizes errors. In its standard form, BCE penalizes all incorrect hit predictions equally. However, it is intuitive that this should not be the case. For instance, predicting a voxel hit of 0 when the true value is 100 GeV should be penalized more heavily than predicting 0 when the true value is 1 GeV. To account for this, we experimented with reweighting the BCE terms proportional to the voxel value. Despite this modification, we did not observe a significant improvement.

Maximizing the ELBO function leads to maximizing the log-likelihood of $p_\theta(x)$. In practice, VAEs are trained by minimizing the negative ELBO function, from which it is straightforward to notice that the first term in the r.h.s. of Eq. (5) becomes mean squared error (MSE), while the second and third terms correspond to the binary cross entropy. After taking these considerations into account,

flipping the sign in the ELBO function and neglecting constant terms, Eq. (5) becomes:

$$-\langle \ln p_\theta(\boldsymbol{\chi}, \boldsymbol{\xi}|\mathbf{z}) \rangle_{q_\phi(\mathbf{z}|\mathbf{x})} = \left\langle \sum_{i=1}^n (\chi_i(\theta) \cdot \Theta(x_i) - x_i)^2 \right\rangle_{q_\phi(\mathbf{z}|\mathbf{x})} - \langle \Theta(x_i) \ln(p_{\xi_i}(\theta)) + (1 - \Theta(x_i)) \ln(1 - p_{\xi_i}(\theta)) \rangle_{q_\phi(\mathbf{z}|\mathbf{x})}. \quad (6)$$

In Eq. (6), we use the *Gumbel trick* for the hits vector. We explain the approach in what follows.

The second term in the r.h.s. of Eq. (2) is known as the VAE regularizer. This term can be parsed as:

$$\langle \ln \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z})} \rangle_{q_\phi(\mathbf{z}|\mathbf{x})} = \langle \ln q_\phi(\mathbf{z}|\mathbf{x}) \rangle_{q_\phi(\mathbf{z}|\mathbf{x})} - \langle \ln p_\theta(\mathbf{z}) \rangle_{q_\phi(\mathbf{z}|\mathbf{x})} \quad (7)$$

Let us focus on the first term on the r.h.s. of Eq. (7). When taking the gradient of the entropy with respect to the model parameters one faces two issues: *i*) due to the discrete nature of the latent variables z , the gradient becomes singular; and *ii*) given a function evaluated on a random variable, it is ill-defined taking the gradient of the function with respect to the parameters of the random variable's probability density function. Moreover, taking the gradient of a discrete estimator, $\sum_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})}(\dots)$, with respect to ϕ is not well-defined, since the gradient variables appear in the summation argument. The first issue can be easily addressed by simply considering a continuous step-like function, such as a sigmoid, $\sigma(\bullet)$. Specifically, we replace \mathbf{z} with $\boldsymbol{\zeta}$, where $\zeta_i = \sigma([\text{encoded data point}]_i \cdot \beta)$, and β is an annealing parameter, such that $\lim_{\beta \rightarrow \infty} \boldsymbol{\zeta} = \mathbf{z}$. The second issue can be addressed by means of the so-called *Gumbel trick* [25]. The Gumbel trick has become an umbrella term which refers to a set of methods to sample from discrete probabilities or to estimate its partition function. In our case, we simply generate latent variables $\boldsymbol{\zeta}$ via

$$\zeta_i = \sigma((l_i(\phi, x) + \sigma^{-1}(\rho_i))\beta) \quad \text{for all } i = 1, \dots, m, \quad (8)$$

where $\{\rho_i\}_{i=1}^m$ is a set of i.i.d. uniform random numbers, and $\{l_i(\phi, \mathbf{x})\}_{i=1}^m$ is a set of logits, *i.e.*, a set of unbounded real numbers generated from the encoder. The connection with Gumbel distributed random numbers is due to the fact that $\sigma^{-1}(\rho) \sim G_1 - G_2$, where G_1 and G_2 are two Gumbel distributed random numbers. Moreover, under the recipe given in Eq. (8) one is guaranteed that in the discrete regime of $\boldsymbol{\zeta}$ (*i.e.*, $\beta \rightarrow \infty$), $P(\zeta_i = 1) = \sigma(l_i(\phi, x))$ [25? ?]. Taking into account the previous, we can then express the entropy as

$$\langle \ln q_\phi(\boldsymbol{\zeta}|\mathbf{x}) \rangle_{q_\phi(\mathbf{z}|\mathbf{x})} = \left\langle \sum_{i=1}^m \zeta(l_i, \beta, \rho_i) \ln \sigma(l_i) + (1 - \zeta(l_i, \beta, \rho_i)) \ln(1 - \sigma(l_i)) \right\rangle_{q_\phi(\mathbf{z}|\mathbf{x})} \quad (9)$$

The second term in Eq. (7) can be expanded as follows. By design the functional form of $p_\theta(\mathbf{z})$ is that of a Boltzmann distribution in a heat bath at temperature $T = 1$ ($k_B = 1$). Therefore, we have:

$$\langle \ln p_\theta(\mathbf{z}) \rangle_{q_\phi(\mathbf{z}|\mathbf{x})} = -\langle E(\mathbf{z}) \rangle_{q_\phi(\mathbf{z}|\mathbf{x})} - \ln Z. \quad (10)$$

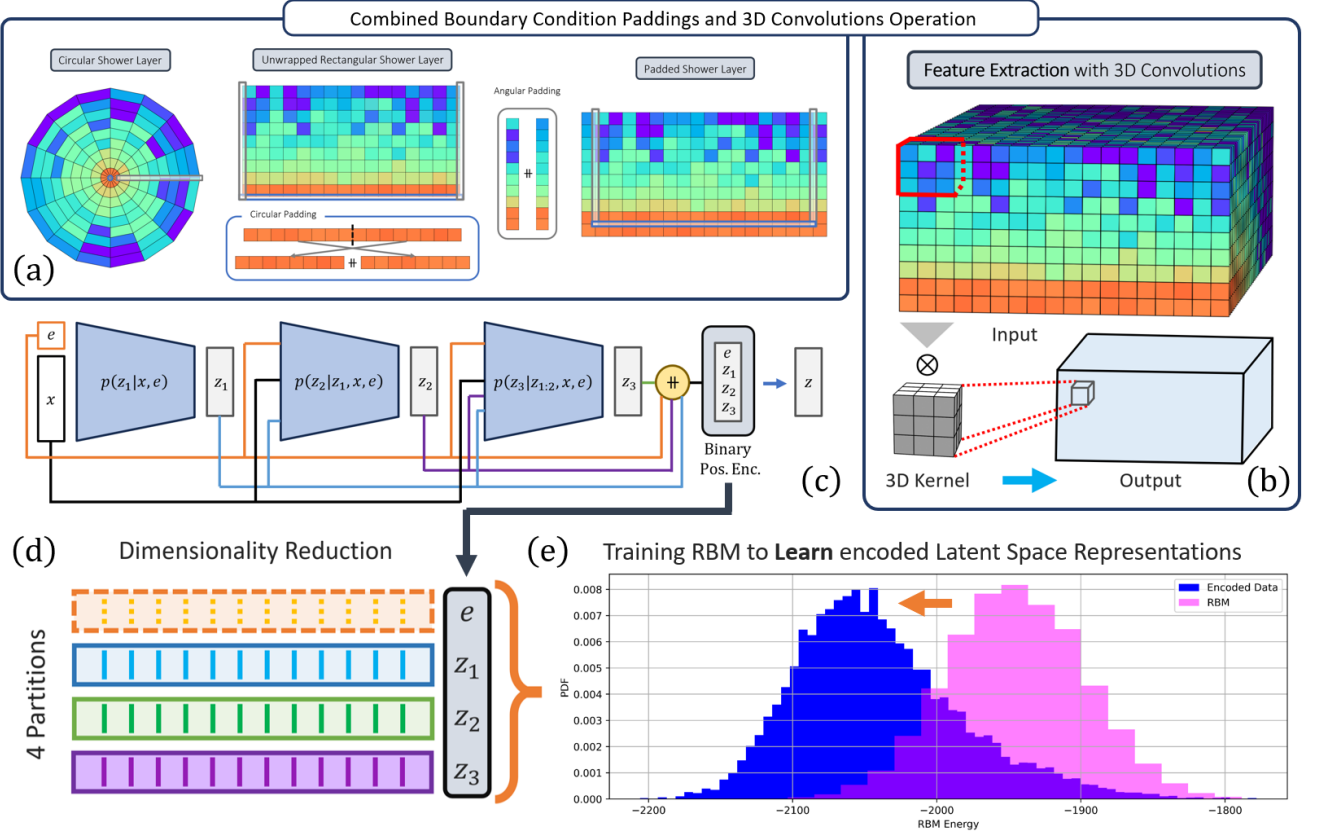


FIG. 3. Diagram of the encoding framework. (a) We unwrap the cylindrical shower into a tensor of rank 3 with indices. We account the angular periodicity of the cylindrical geometry by padding the tensor in *theta* dimension, such that the size becomes $45 \times 18 \times 9$. To account for the neighboring voxels in the center of the cylinder, we pad the tensor in the corresponding radial dimension. We pad it by taking the centermost voxels, splitting it in half and permuting the two halves. (b) These operations are performed several times, each prior to a 3D convolution operation for feature extraction. (c) The encoder embeds hierarchy levels, *i.e.*, the first encoder generates a fraction of the encoded data, which is then fed to the second encoder (together with the input) to generate the remaining fraction of the encoded data. The encoded data is used to train the QPU RBM. The encoded data and the incidence energy is passed to the decoder to reconstruct the energy per voxel. (d) The Calo4pQVAE uses a discrete binary latent space and assumes a Boltzmann distribution for prior. The energy function in the Boltzmann distribution corresponds to a sparse 4-partite graph, which allows the direct mapping to Pegasus-structured Advantage quantum annealer.

The first term in the r.h.s. is the energy function averaged over the approximate posterior, whereas the second term is the free energy, which is independent of the encoded samples. The free energy can be replaced by the internal energy, U , minus the entropy, S , times the temperature, *viz.* $U - TS$. We use the *high temperature* gradient approximation to replace the second term in the r.h.s. of Eq. (10) with the internal energy, which is a common practice in approximating the free energy when training RBMs. In the high temperature gradient approximation, as the energy in the system saturates, the specific heat converges to zero. We demonstrate in Appendix D that when $\langle E(\mathbf{z}) \rangle \langle \partial E(\mathbf{z}) / \partial \phi \rangle = \langle E(\mathbf{z}) \partial E(\mathbf{z}) / \partial \phi \rangle$, the specific heat is zero, which allows us to rewrite the previous Equation as

$$\langle \ln p_{\theta}(\mathbf{z}) \rangle_{q_{\phi}(\mathbf{z}|\mathbf{x})} = -\langle E(\mathbf{z}) \rangle_{q_{\phi}(\mathbf{z}|\mathbf{x})} + U. \quad (11)$$

In this regime, the entropy is solely configurational, scales

linearly with the number of units in the RBM and is independent of the energy parameters.

Computing U in Eq. (11) requires calculating the partition function which becomes intractable beyond a few tens of nodes due to the exponential scaling of the number of states with the number of nodes. Instead, we estimate U using uncorrelated samples. To obtain these uncorrelated samples, one performs Markov Chain Monte Carlo simulations also known as block Gibbs sampling. In the following section we elaborate on how we sample from the 4-partite RBM.

C. 4-partite Restricted Boltzmann Machine

Let us consider a 4-partite restricted Boltzmann machine. For this purpose, we denote each of the four layers

as v, h, s and t . The energy function is defined by:

$$\begin{aligned} E(\mathbf{v}, \mathbf{h}, \mathbf{s}, \mathbf{t}) = & -a_i v_i - b_i h_i - c_i s_i - d_i t_i \\ & -v_i W_{ij}^{(0,1)} h_j - v_i W_{ij}^{(0,2)} s_j \\ & -v_i W_{ij}^{(0,3)} t_j - h_i W_{ij}^{(1,2)} s_j \\ & -h_i W_{ij}^{(1,3)} t_j - s_i W_{ij}^{(2,3)} t_j, \end{aligned} \quad (12)$$

where we are using the double indices convention for summation.

The Boltzmann distribution has the following form:

$$p(\mathbf{v}, \mathbf{h}, \mathbf{s}, \mathbf{t}) = \frac{\exp(-E(\mathbf{v}, \mathbf{h}, \mathbf{s}, \mathbf{t}))}{Z}. \quad (13)$$

where Z is the partition function,

$$Z = \sum_{\{\mathbf{v}, \mathbf{h}, \mathbf{s}, \mathbf{t}\}} \exp(-E(\mathbf{v}, \mathbf{h}, \mathbf{s}, \mathbf{t})). \quad (14)$$

Similar to the 2-partite RBM, we can express the distribution over any one of the layers conditioned to the remaining three in terms of the ratio of the Boltzmann distribution and the marginalized distribution over the layer of interest. Without any loss in generality, let us assume the layer of interest is h , then the probability distribution on $p(\mathbf{h}|\mathbf{v}, \mathbf{s}, \mathbf{t})$ is given by:

$$\begin{aligned} p(\mathbf{h}|\mathbf{v}, \mathbf{s}, \mathbf{t}) &= \frac{p(\mathbf{v}, \mathbf{h}, \mathbf{s}, \mathbf{t})}{p(\mathbf{v}, \mathbf{s}, \mathbf{t})} \\ &= \frac{\exp(-E(\mathbf{v}, \mathbf{h}, \mathbf{s}, \mathbf{t}))}{\sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}, \mathbf{s}, \mathbf{t}))} \\ &= \prod_i \frac{\exp(h_i \cdot (b_i + B_i))}{1 + \exp(b_i + B_i)} \end{aligned} \quad (15)$$

with

$$B_i = W_{ji}^{(01)} v_j + W_{ij}^{(12)} s_j + W_{ij}^{(13)} t_j \quad (16)$$

We can rewrite Eq. (15) in a rather compact form. Furthermore, we do the same for the case where we condition any of the four partitions on the remaining three. We obtain:

$$p(\mathbf{v} = \mathbf{1}|\mathbf{h}, \mathbf{s}, \mathbf{t}) = \prod_i \sigma(a_i + A_i), \quad (17a)$$

$$p(\mathbf{h} = \mathbf{1}|\mathbf{v}, \mathbf{x}, \mathbf{y}) = \prod_i \sigma(b_i + B_i), \quad (17b)$$

$$p(\mathbf{s} = \mathbf{1}|\mathbf{v}, \mathbf{h}, \mathbf{t}) = \prod_i \sigma(c_i + C_i), \quad (17c)$$

$$p(\mathbf{t} = \mathbf{1}|\mathbf{v}, \mathbf{h}, \mathbf{s}) = \prod_i \sigma(d_i + D_i). \quad (17d)$$

where

$$A_i = W_{ij}^{(01)} h_j + W_{ij}^{(02)} s_j + W_{ij}^{(03)} t_j, \quad (18a)$$

$$B_i = W_{ji}^{(01)} v_j + W_{ij}^{(12)} s_j + W_{ij}^{(13)} t_j, \quad (18b)$$

$$C_i = W_{ji}^{(02)} v_j + W_{ji}^{(12)} h_j + W_{ij}^{(23)} t_j, \quad (18c)$$

$$D_i = W_{ji}^{(03)} v_j + W_{ji}^{(13)} h_j + W_{ji}^{(23)} s_j. \quad (18d)$$

We then propose a Gibbs sampling process to sample from the joint distribution $p(\mathbf{v}, \mathbf{h}, \mathbf{s}, \mathbf{t})$ of the 4-partite RBM. Notice that the joint distribution can be factorized as $p(\mathbf{v}, \mathbf{h}, \mathbf{s}, \mathbf{t}) = p(\mathbf{v}|\mathbf{h}, \mathbf{s}, \mathbf{t})p(\mathbf{h}, \mathbf{s}, \mathbf{t})$. Extending the approach used in 2-partite RBM, we perform Markov chain Monte Carlo by sequentially updating each partition conditioned on the remaining, *i.e.*, the prior in step $n+1$ is deemed the posterior in step n . The Gibbs sampling process, at iteration n , is done in the following steps:

1. Sample partition \mathbf{v} : $p(\mathbf{v}|\mathbf{h}^{(n)}, \mathbf{s}^{(n)}, \mathbf{t}^{(n)})$
2. Sample partition \mathbf{h} : $p(\mathbf{h}|\mathbf{v}^{(n+1)}, \mathbf{s}^{(n)}, \mathbf{t}^{(n)})$
3. Sample partition \mathbf{s} : $p(\mathbf{s}|\mathbf{v}^{(n+1)}, \mathbf{h}^{(n+1)}, \mathbf{t}^{(n)})$
4. Sample partition \mathbf{t} : $p(\mathbf{t}|\mathbf{v}^{(n+1)}, \mathbf{h}^{(n+1)}, \mathbf{s}^{(n+1)})$

Notice that one block Gibbs sampling step corresponds to four sampling steps. By repeating this process iteratively, we generate samples that approximate the joint distribution.

We have shown how to generalize an RBM to a four-partite graph. It is important to emphasize that, as with any deep generative model, the features of the dataset are expressed in latent space. In this context, each state in latent space encodes specific features of the dataset. Different techniques such as t-SNE [26] and LSD [27], can illustrate how dataset features are embedded in latent space after training. These methods allow for the conditioning of the *prior* on specific features. There is a rather straightforward way to condition the *prior* on given dataset parameters during training, which is widely used across different frameworks. Generally, this involves using specific channels to condition the prior and specifics depend on the actual framework. An accurate and robust method to sample from latent space constrained to specific features is crucial for practical applications in generative deep models. In the present context, a calorimeter surrogate requires input related to particle type, incident particle energy and incident angle, among other parameters. In the following, we show how to condition the 4-partite RBM sampling process.

1. Conditioned 4-partite Restricted Boltzmann Machine

In this section we outline the approach to condition the 4-partite RBM. In this context, the conditioned RBM refers to the scenario where a subset of nodes in the RBM is kept fixed during the block Gibbs sampling process. We adapt the methodology from [28] to our Calo4pQVAE. A crucial element of our approach is the use of one complete partition to deterministically encode the data features. Although utilizing the entire partition for feature encoding might appear excessive, it is necessary due to the sparsity of the quantum annealer. Exploring the optimal number of nodes required to efficiently encode the features is beyond the scope of this paper. Moreover, as we move forward towards more

complex datasets, we anticipate an increase in the number of features that need to be encoded, underscoring the importance of our chosen approach as a baseline.

Notice that by treating one partition as a conditioning parameter, we effectively modify the self-biases of the remaining partitions. To fix ideas, we consider partition v as the conditioning partition, therefore we may rewrite Eq. (12) as

$$E(\mathbf{h}, \mathbf{s}, \mathbf{t}|\mathbf{v}) = -(b_i + v_j W_{ji}^{(0,1)})h_i - (c_i + v_j W_{ji}^{(0,2)})s_i \\ - (d_i + v_j W_{ji}^{(0,3)})t_i - h_i W_{ij}^{(1,2)}s_j \\ - h_i W_{ij}^{(1,3)}t_j - s_i W_{ij}^{(2,3)}t_j \quad (19)$$

The block Gibbs sampling procedure is the same as before via Eqs. (17) although we skip Eq. (17a). The authors in [28] caution on the use of contrastive divergence when training a conditioned RBM due to the possibility of vanishing gradient even when the conditioned RBM might not be fully trained. This happens when mixing times are larger than the number of Gibbs sampling steps. Despite the previous, we use contrastive divergence in our framework and we show how the estimated log-likelihood saturates after 250 epochs. We further elaborate and discuss this approach in the discussion section.

D. Quantum Annealers

A quantum annealer (QA) is an array of superconducting flux quantum bits with programmable spin-spin couplings and biases with an annealing parameter [29]. The motivation for QAs comes from the adiabatic approximation [30], which asserts that if a quantum system is in an eigenstate of its Hamiltonian (which describes the total energy of the system), and the Hamiltonian changes slowly enough, then the system will remain in an eigenstate of the Hamiltonian, although the state itself may change (see Appendix F for a formal derivation). QAs were initially thought of as a faster method to find the ground state of complex problems that could be mapped onto a Hamiltonian H [31]. This can be done by initializing the system in the ground state of some Hamiltonian H_0 , which is easy to prepare both theoretically and experimentally. In addition, by design the commutator $[H, H_0] \neq 0$. The QA interpolates between the two Hamiltonians via

$$H_{QA} = \frac{\mathcal{A}(s)}{2}H_0 + \frac{\mathcal{B}(s)}{2}H \quad (20)$$

with

$$\begin{cases} H_0 = -\sum_i \hat{\sigma}_x^{(i)} \\ H = \sum_i \Delta_i \hat{\sigma}_z^{(i)} + \sum_{i>j} J_{ij} \hat{\sigma}_z^{(i)} \hat{\sigma}_z^{(j)} \end{cases} \quad (21)$$

such that the annealing parameters, $\mathcal{A}(s)$ and $\mathcal{B}(s)$, are two slowly-varying controllable parameters constrained to $\mathcal{A}(0) \gg \mathcal{B}(0)$ and $\mathcal{A}(1) \ll \mathcal{B}(1)$ and $s \in [0, 1]$ [32].

In practice, quantum annealers have a strong interaction with the environment which lead to thermalization and decoherence. Evidence suggests that the culprit are the σ_z operator which couple to the environment [33]. There has been efforts towards mitigating decoherence via *zero noise extrapolation* methods [34].

Thermalization and decoherence are usually unwanted features in quantum systems as it destroys the quantum state. In our case, these features allows us to replace the RBM with the QA. In other words, RBMs are classical simulations of QAs. A non-desired feature in our framework correspond to system arrest or freeze-out during annealing [35], akin to glass melts subject to a rapid quench [36]. Similar to glasses, the annealing time and protocol can have a dramatic impact on the end state [37]. It has been shown that the distribution in this freeze-out state can be approximated with a Boltzmann distribution [19].

In our pipeline we use Dwave's *Advantage_system6.4* [38] which is composed by 5627 qubits and are coupled such that it forms a quadri-partite graph. Typically each qubit is coupled with 16 other qubits. In Fig. 4 we show the histogram for number of connections between each of the four partitions.

Notice that the RBM data are binary vectors such that each element can take value of 0 or 1, whereas the qubits can have values -1 or 1 . Hence, to map the RBM onto the QA one redefines the RBM variables $x_i \rightarrow (\sigma_z^{(i)} + 1)/2$ and rearrange the terms in the RBM Hamiltonian to be mapped onto the QA (this is explicitly shown in Appendix F). The previous variable change will lead to an energy offset between the RBM and the QA, which henceforth we neglect. It is important to stress that the QA is in a heat bath with temperature $T_{QA} \lesssim 15mK$ [39], while the annealing parameter has an upper bound $\mathcal{B}(1) \approx 5.0 \cdot 10^{-24}J$ [40], which implies that the prefactor $\beta_{QA} = \frac{\mathcal{B}(1)}{2k_B T_{QA}} \approx 12$ while in the RBM by design $\beta_{RBM} = 1$. To ensure that both, the RBM and the QA describe the same Boltzmann distribution with the same temperature, one can either re-scale the QA Hamiltonian by $1/\beta_{QA}$ or the RBM Hamiltonian by β_{QA} . Currently, there is not a way to measure the QA prefactor, however there are different ways to estimate it [33?]. In addition, there is not a direct way to control this prefactor. The most common way to estimate this prefactor is via the Kullback-Liebler divergence between the QA and the RBM distributions, whereby one introduces a tuning parameter β as a prefactor in the RBM distribution which is tuned to minimize the Kullback-Liebler divergence. It is in the process of estimating the β_{QA} that we also indirectly control the parameter by rescaling the Hamiltonian. The previous can be mathematically expressed as

$$\beta_{t+1} = \beta_t - \eta(\langle H \rangle_{QA} - \langle H \rangle_{RBM}) . \quad (22)$$

The previous converges when the average energy from the RBM matches that from the QA, in which case,

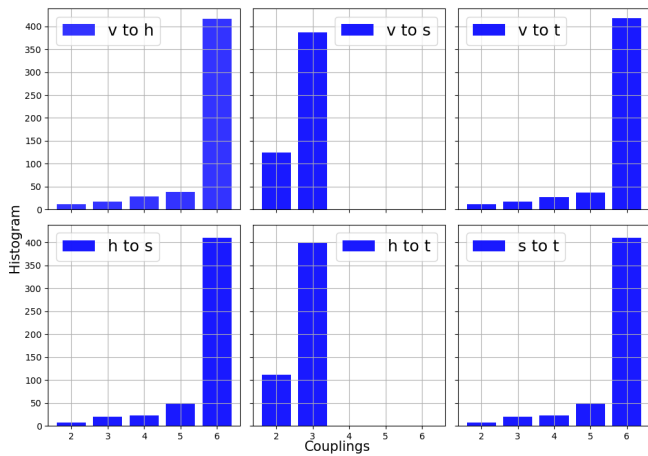


FIG. 4. Quadripartite RBM weight matrices. Each panel correspond to the histogram of connections between partition A and partition B (see legend).

$\beta = \beta_{QA}$. Due to coupling-dependent temperature fluctuations, β_{QA} is expected to change in the process of training the model. For this reason the method described by Eq. (22) should be employed, in principle, after each parameter update. Notice that optimizing β in Eq. (22) requires samples from the RBM, furthermore, it is the classical RBM temperature which is being tuned to match the distribution of the QA. In our case, we want the opposite, to fit the QA distribution to that of the RBM and not the other way around. To address this point, $H(x)$ is replaced with an annealed Hamiltonian $H(x, \beta) = H(x)/\beta$, *i.e.*, instead of tuning the temperature in the classical RBM, to match the distribution in the QA, one iteratively rescales the Hamiltonian in the QA to effectively tune the QA's temperature to match that of the classical RBM. This approach, as already mentioned, is well-known [33? ?] and has been proven to be empirically robust yet slow to converge. We therefore propose a different protocol which empirically converges faster than the KL divergence:

$$\beta_{t+1} = \beta_t \left(\frac{\langle H \rangle_{QA}}{\langle H \rangle_{RBM}} \right)^\delta. \quad (23)$$

Notice that the previous map has a fixed point at $\beta_t = \beta_{QA}$. The condition for a stable fixed point is $\lambda(\delta) < 1$, where

$$\lambda(\delta) = \begin{cases} |1 + \frac{\sigma_{QA}^2}{\langle H \rangle_{B^{(1)}}^2}|, & \delta = 1 \\ |1 + \delta \frac{\sigma_{QA}}{\langle H \rangle_{QA}}|, & \delta \neq 1. \end{cases} \quad (24)$$

In Fig. 5 we show Eq. (24) *vs* β for different values of δ . The values of β chosen for this plot correspond to where we typically find the fixed point. We call δ a stability parameter since we can tune it to stabilize the mapping per iteration and modulate the fixed point attractor feature to ultimately converge in a smaller number of iterations,

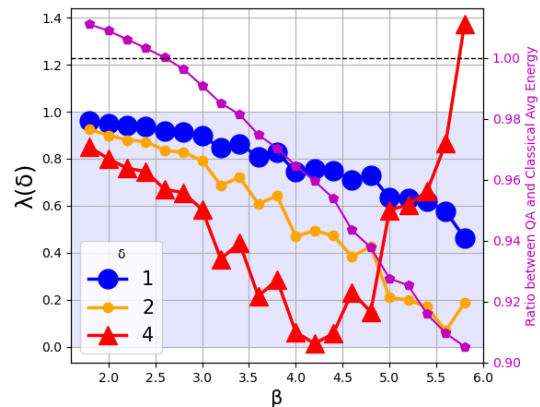


FIG. 5. $\lambda(\delta)$ (see Eq. (24)) *vs* β for different δ values (see legend). The stability region is shaded in light blue. Different δ values affect the stability depending on the β values. For instance, for low β values, large δ parameters leads to better stability; conversely, large β values with large δ parameter leads to instabilities. The purple pentagons correspond to the average energy ratio between QPU samples and classical samples. The intersection between the black dashed line and the purple curve define the fixed point.

as we show in the next section. The purple markers in Fig. 5 correspond to the ratio of the average RBM energy obtained from the QA and that obtained from classical sampling. When the ratio equals one, the mapping in Eq. (23) is at the fixed point. In Appendix F we provide a fully detailed derivation and in the next section we compare both methods.

Up to this point, we have detailed the process of replacing the RBM with the QA in the Calo4pQVAE. As previously mentioned, we utilize a conditioned RBM to enable the sampling of showers with specific characteristics. In the next section, we will explore various approaches for conditioning the QA, aiming to achieve targeted sampling of particle showers with desired features.

1. Conditioned Quantum Annealer

Quantum annealers were designed to find solutions to optimization problems by identifying a Boolean vector that satisfies a given set of constraints. Typically, QAs are not intended to be conditioned or manipulated in terms of fixing specific qubits during the annealing process. However, there are different approaches that can be employed to fix a set of qubits during the annealing process. In this context, we will present two such approaches. For clarity, let us revisit the concept of conditioning in the realm of QAs. Our goal is to utilize QAs in a manner that allows us to fix a subset of qubits, denoted as $\sigma_z^{(k)}$ (see Eq. (21)), *a priori*, such that these qubits remain in their predetermined states throughout and after the annealing process.

Reverse annealing with zero transverse field for conditioning qubits. This approach requires control over the biases in H_0 from Eq. (21), where $H_0 = \sum_i \kappa_i \hat{\sigma}_x^{(i)}$ and $\{\kappa_i\}$ are directly specified by the user. Initially, we set the qubits encoding the condition $\sigma_z^{(k)}$ while the rest of the qubits are randomly initialized. We also set the biases $\kappa_k = 0$ to ensure that the transverse field does not alter the state of $\sigma_z^{(k)}$. Subsequently, we perform reverse annealing, by starting from $s = 1$ and reversing the annealing process towards $s = 0$, before completing the annealing process as usual. The primary drawbacks in this approach are:

- Speedup compromise: The annealing process is effectively doubled in duration due to the reverse annealing step.
- Condition destructed by thermal fluctuations: There is a possibility of the conditioned state being altered due to thermal fluctuations.

Both drawbacks can be mitigated by decreasing the annealing time, as this would not only reduce the overall duration but also minimize the destruction of the conditioned-encoding state by thermal fluctuations. However, as previously noted, reducing the annealing time can lead to the issue of dynamical arrest, where the system becomes trapped in a local minimum [35]. Given that our current framework relies on thermodynamic fluctuations, we leave this approach for future exploration and proceed to discuss an alternative method that does not suffer from these drawbacks and is more practical for immediate implementation.

Conditioning qubits through flux biases. This approach relies on using the external flux bias, Φ_k^x , as effective biases to fix the qubits during annealing. The external flux bias were introduced as a practical remedy to render the biases h_i time-independent during the annealing by having $B(s)h_i = 2\Phi_i^x I_p(s)$, where $I_p(s)$ denotes the magnitude of the supercurrent flowing about the rf-SQUID loop. This flux bias is applied to the qubit loop about which the supercurrent flows [41]. Flux biases work as effective biases on qubits. Hence, before the annealing we specify the encoded incidence energy via the flux biases. Through out our experiments, we always made sure that the partition encoding the incidence energy in the QA after annealing matched the encoded incidence energy before the annealing.

III. RESULTS

In this section, we present the results concerning training and evaluating our model on Dataset 2. Therefore, we present as results the training aspects of our framework.

Training: We train our model for 250 epochs. Each epoch typically has 625 updating steps. The number

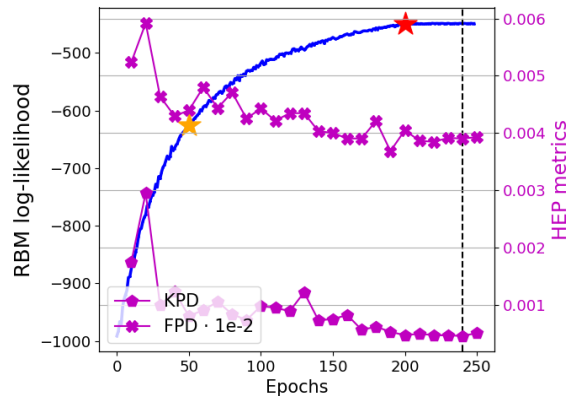


FIG. 6. Restricted Boltzmann Machine log-likelihood *vs* epochs. *Annealed importance sampling* and *reverse annealed importance sampling* methods were used to estimate the partition function. The annealed step was set to $1/30$. The yellow star marks the epoch were the annealed training parameters have reached their final values. The red star marks the point after which the encoder and decoder training parameters are frozen. In magenta, we plot the KPD and FPD (right axis) *vs* epochs.

TABLE I. Fréchet Particle Distance (FPD) and Kernel Particle Distance (KPD) metrics, implemented in the JetNet [42] library and adapted for the CaloChallenge [3].

Calo4pQVAE	FPD	KPD
MCMC (100k)	$(390.35 \pm 1.85) \times 10^{-3}$	$(0.46 \pm 0.05) \times 10^{-3}$

of block Gibbs sampling steps was set to 3000. During the first 50 epochs we anneal the model parameters, such as those used in the Gumbel trick, from smooth to a sharp step, to mitigate the discontinuities in the gradient related to the use of discrete variables. After the annealing parameters have reached their final values, we train the model for the next 150 epochs, after which we freeze the encoder and decoder parameters at epoch 200, while the prior distribution parameters keep being updated. We use an Nvidia A100 GPU. Our experiments show that this last step was necessary for the RBM log-likelihood to saturate. In Fig. 6 we show the RBM log-likelihood *vs* epochs. The yellow star points to when the annealed parameters reached their final values, whereas the purple star points to when the encoder and decoder parameter were frozen, after which it is clear the log-likelihood saturates. To estimate the log-likelihood, after training, we used *annealed importance sampling* and *reverse annealed importance sampling* methods [43?] with an annealed step set to $1/30$. As the model updates its parameters, the encoded data used to estimate the log-likelihood will also be modified from epoch to epoch, hence, we stored the validation encoded data for each epoch in order to accurately estimate the RBM log-likelihood. During training we save an instance of the model every ten epochs.

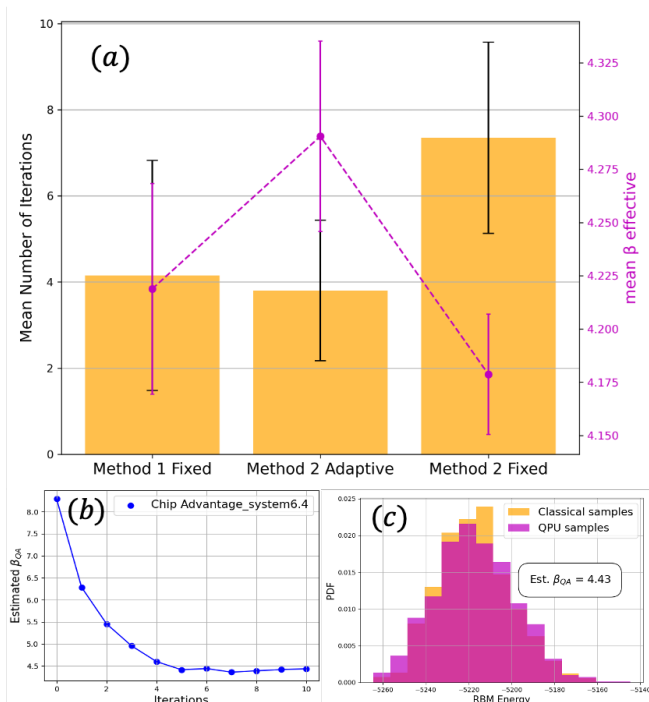


FIG. 7. **a)** Mean number of iterations to meet the reduced standard error threshold in Eq. (25) using different iterative methods. Method 1 uses the KL divergence as in Eq. (22). Method 2 and 3 use Eq. (23), in addition Method 2 adapts the δ parameter after each iteration such that $\lambda \approx 0$ (see Eq. (24)). The dashed purple line corresponds to the ratio between final minus initial effective β and the number of iterations, such that higher values implies faster convergence. The bars correspond to standard deviation. **b)** Estimated inverse temperature *vs* iterations using method 2 adaptive. **c)** RBM histogram using classically generated and QA-generated samples after estimated temperature convergence.

Validation: To validate our model, we use Fréchet physics distance (FPD) and the kernel physics distance (KPD) [13]. These are *integral probability* metrics for high energy physics, specifically designed to be sensitive to modelling of shower shape variables. In Fig. 6 we show these metrics *vs* epochs. Henceforth, the results are generated using the 240-epoch model, corresponding to the best KPD metric with good FPD metric in the saturated RBM log-likelihood regime. We further validate our model’s reconstruction, classical sampling and QA sampling with GEANT4 data. To sample using the Advantage_system6.4 QA [16], we first estimate the effective β by iteratively using Eq. (23) until the absolute difference between the QA’s energy and that of the RBM is smaller than the reduced standard error, *viz.*,

$$|\langle H \rangle_{QA} - \langle H \rangle_{RBM}| < \frac{2}{\sqrt{N}} \frac{\sigma_{QA} \sigma_{RBM}}{\sigma_{RBM} + \sigma_{QA}}. \quad (25)$$

In Fig. 7 we show the mean number of iterations to meet the reduced standard error threshold in Eq. (25)

using different iterative methods. Method 1 uses the KL divergence as in Eq. (22). Method 2 uses Eq. (23), where Method 2 *adaptive* adapts the δ parameter after each iteration such that $\lambda \approx 0$ from Eq. (24). The dashed purple line corresponds to the ratio between final minus initial effective β and the number of iterations, such that higher values implies faster convergence.

In each API call the user specifies the bias and coupler parameters, as well as the number of samples to be generated. The QA is programmed once using these biases and couplers and then performs the annealing process sequentially for the number of iterations requested, returning samples for each anneal. In addition, the user has the option to specify the flux bias parameters with each API call.

In the case of conditioned sampling using the QA, we estimate the temperature under flux biases. We observed that using flux biases increases the QA’s effective temperature. Additionally, programming the QA increases temperature fluctuations. In Fig. 8 (a-c) we present the RBM energy histogram obtained from encoded GEANT4 showers, classically generated samples and QA-generated showers. Specifically, in Fig. 8 a) we estimate the inverse temperature for each incident energy condition, as outlined previously, before generating the QA sample. Fig. 8 d) shows the estimated inverse temperature per incident energy condition. Notably, we observe rather large but rare thermal fluctuations of the order of 10%. In Fig. 8 e) we show the same estimated inverse temperature shown in Fig. 8 d) versus the incident energy condition, suggesting that these large thermal fluctuations are independent of the incident energy condition. To further investigate these effects, we repeated the process of generating conditioned samples using the QA, but estimated the QA inverse temperature only once at the beginning of the sampling process. This estimate is depicted as a dashed black line Fig. 8 d). In Fig. 8 b) we show the RBM energy, where a small shift in the energy of the QA samples is noticeable. Using the same inverse temperature obtained in the previous setting, we generated new samples and introduced a 2.5s pause between QA API calls, that is per sample. In Fig. 8 c) we show the RBM energy obtained from these new samples, from which it is clear that the energy shift has disappeared. Additionally, in Fig. 8 d) we show the estimated inverse temperature in the absence of flux biases in dashed red. We conclude that there are two main contributions to the thermal fluctuations: the programming of the QA and the flux biases. The former can be substantially mitigated by either pausing the sampling process between samples or estimating the inverse temperature per sample. The latter can be accounted for by estimating the inverse temperature in the presence of flux biases. We discuss this further in the Discussion section.

After estimating the effective β in the QA, we use the validation dataset composed of 10,000 data points to benchmark the model. In Fig. 9 (a) we show the

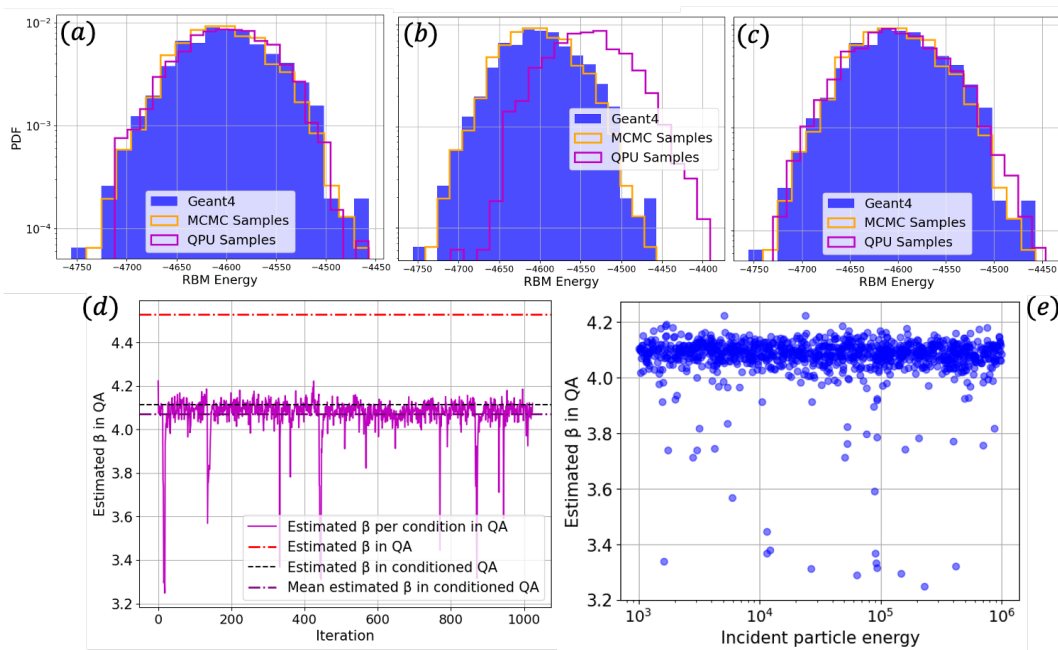


FIG. 8. RBM energy histogram obtained from encoded GEANT4 showers, classically generated samples and QA-generated samples. **a)** QA samples obtained via inverse temperature estimation per incident energy condition, **b)** single estimated inverse temperature, **c)** single estimated inverse temperature and a sleep time of 2.5s between API call. **d)** Estimated inverse temperature per incident energy condition in solid purple, estimated inverse temperature in the absence of flux biases in dashed red, estimated inverse temperature for single incident energy condition used in **b)** and **c)**. **e)** Estimated inverse temperature *vs* incident energy condition.

shower energy histograms, while in Fig. 9 (b) we show the sparsity histogram. The sparsity index is a measure of the sparsity of the shower and we define it as the ratio between zero-value voxels in the shower and total number of voxels. Each of the histograms include the GEANT4 data and the model’s reconstruction. Furthermore, we test the generative model by generating samples from the RBM and feeding these samples together with the incidence energy from the validation dataset to the decoder and generating shower samples. Similarly, we test the quantum-assisted generative model by generating samples from the QA instead of the classical RBM. Both, the classically generated samples and the quantum-assisted generated ones are included in Figs. 9 (a) and (b) and are labelled *Sample* and *Sample w/ QPU*, respectively. In Fig. 9 (c) we show the RBM energy distribution corresponding to the encoded validation dataset and, both, the classically and QA sampled data.

To further validate our model, we generate a synthetic dataset composed by 100,000 events with the same incidence energy distribution as the training dataset. We compare our synthetic dataset with Dataset 2’s test set [3] by computing the mean energy in the r , θ and z directions of the cylinder (see Fig. 1). We show these results in Figs. 9 (d-f).

IV. DISCUSSION

In the previous section we outlined the process of training our conditioned quantum-assisted Calo4pQVAE and described a novel and faster method to estimate the QA’s effective inverse temperature before sampling using the QPU. We evaluated the performance of our model by means of the KPD and FPD metrics, achieving results on the same order and one order of magnitude higher than CaloDiffusion, respectively, as shown in Table I [11]. Furthermore, when compared with the models in the CaloChallenge under this metric, our framework performs better than more than half of the 18 models considered [12]. Our results demonstrate that *i)* our framework is able of reconstructing the showers and preserving sparsity, and *ii)* the prior is effectively learning the structure of the encoded data.

Another critical metric to evaluate is the shower generation time, which in our framework depends on the decoder processing time and the RBM generation time. Estimating the RBM generation time is not straightforward and there are multiple approaches for this task. This estimation depends on the size of the RBM, its coordination number and the values of the couplings and biases, among other things. Furthermore, it has been shown that the RBM mixing time increases with training [44]. On the other hand, as mentioned earlier, QA can suffer from freeze-out, depending on features such as the cou-

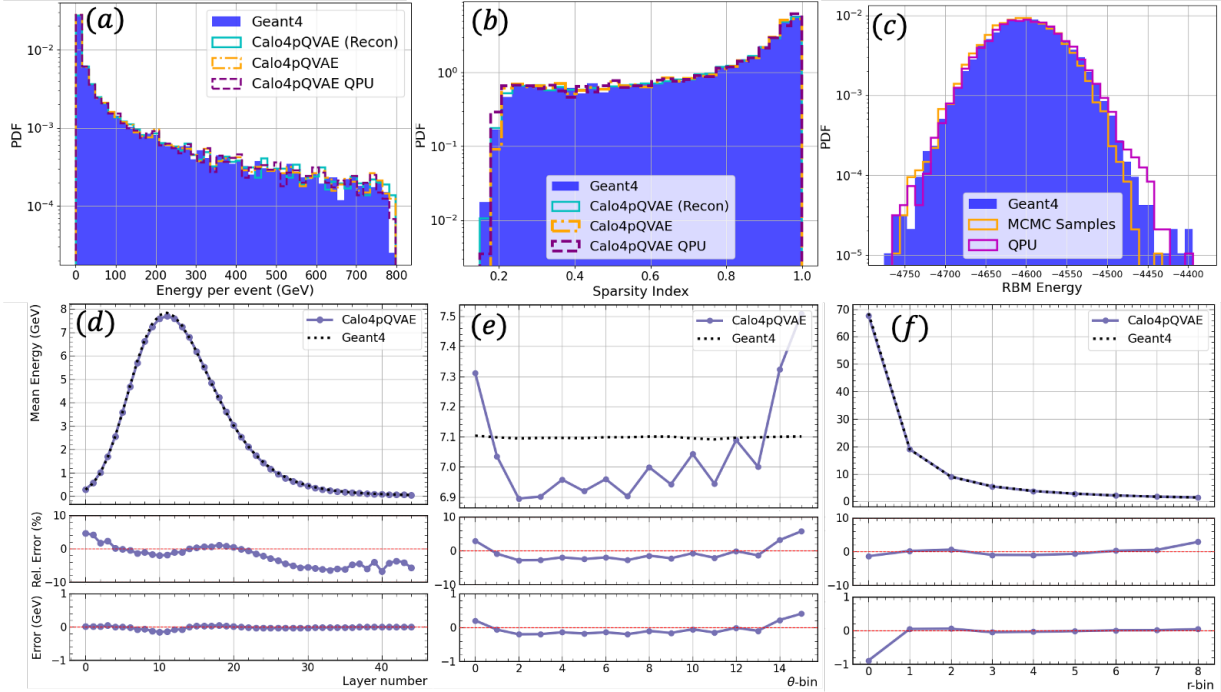


FIG. 9. Comparison between GEANT4 samples, reconstructed samples using Calo4pQVAE, classically generated and QA-generated samples. **a)** Energy per event histogram, **b)** sparsity index per event, **c)** RBM energy per event. Comparison between GEANT4 and classically generated samples. Mean event energy *vs* layer number (**panel d**), *vs* angular number (**panel e**), and *vs* radial number (**panel f**). Each panel shows the relative error and the error underneath. The average is taken from 100k events.

pling and bias values, the coordination number and the annealing time. Therefore, benchmarking the classical and quantum-assisted generation times requires careful consideration, and we leave this research for a future paper. Ultimately, both frameworks are several orders of magnitude faster than GEANT4.

We previously mentioned that we set the number of block Gibbs sampling steps to 3000. We observed that with this number of steps, the RBM log-likelihood versus epochs increased monotonically. In this case, generating 1024 samples classically takes approximately 1 s with $\mathcal{O}(1000)$ MCMC steps. Conversely, when sampling with QA, there are three characteristic timescales to consider: the programming time (approximately 10 ms), the annealing time (approximately $20\mu\text{s}$) and the readout time (approximately $100\mu\text{s}$). This results in a total generation time of approximately 0.1 s for $\mathcal{O}(1000)$ samples, *i.e.*, one order of magnitude faster than the classical method, assuming the QPU programming step is performed only once. These estimates per sample are shown in Table II, which are highly competitive [12]. However, in our current framework, as mentioned earlier, the flux biases conditioning is done during the programming step, effectively increasing the overall time to approximately 10s.

Future iterations of D-wave’s QA are expected to decouple the flux biases conditioning from the programming step. Therefore, we anticipate that the quantum-assisted

TABLE II. Shower generation time estimates using GEANT4 [10], Calo4pQVAE on GPUs (assuming 3k BGS) and Calo4pQVAE with QPU without conditioning.

	GEANT4	GPU (A100)	QPU	Anneal time
Time	$\mathcal{O}(0.1) - \mathcal{O}(10^2)$ s	~ 2 ms	~ 0.2 ms	~ 0.02 ms

framework to be competitive if the flux biasing conditioning remain below the order of milliseconds. Additionally, uncoupling the flux biases step from the QA programming will help mitigate the undesired large temperature fluctuations. Furthermore, since the self-correlation time is bound to increase with the coordination number, we expect the competitiveness of QAs to improve as the number of couplers per qubit increases.

In the immediate future, we will focus on three key aspects:

- **Exploring RBM Configurations:** In the present work, we set the number of units in the RBM to 512 units per partition, which implies a compression factor in the Calo4pQVAE of approximately 30%, similar to that in Ref. [19]. Increasing this number will require increasing the number of block Gibbs sampling steps, as shown in [44?]. We will explore different RBM sizes with varying numbers of block Gibbs sampling steps to optimize performance and computational efficiency.

- **Enhancing the Decoder Module:** We will investigate the use of hierarchical structures and skip connections in the decoder module, as these can have a positive effect on performance, as observed in diffusion models [11]. Implementing these architectures may improve the model’s ability to reconstruct complex data patterns.
- **Upgrading to the Latest Quantum Annealer:** We plan to replace the current QA with D-Wave’s latest version, Advantage2_prototype2.4 [45]. Although it has a smaller number of qubits, it offers a greater number of couplers per qubit and reduced noise, which could enhance the quality of quantum simulations and overall model performance.

By successfully conditioning a quantum annealer to sample from a desired subspace of data and demonstrating robust, resource-efficient temperature-scaling procedures, our work lays the groundwork for broader applications of quantum annealing in generative modeling. This paves the way for additional quantum training strategies and sets a precedent for feature disentanglement as in Ref. [46] with quantum hardware.

As a final comment, our framework combines deep generative models with quantum simulations via Quantum Annealers (QAs). We speculate that the transition to

a QA presents new opportunities not only in the *noisy intermediate-scale quantum* stage but also by paving the way toward utilizing large-scale quantum-coherent simulations [47] as priors in deep generative models.

V. ACKNOWLEDGMENTS

We gratefully acknowledge Jack Raymond, Mohammad Amin, Trevor Lanting and Mark Johnson for their feedback and discussions. We gratefully acknowledge funding from the National Research Council (Canada) via Agreement AQC-002, Natural Sciences and Engineering Research Council (Canada), in particular, via Grants SAPPJ-2020-00032 and SAPPJ-2022-00020. This research was supported in part by Perimeter Institute for Theoretical Physics. Research at Perimeter Institute is supported by the Government of Canada through the Department of Innovation, Science and Economic Development and by the Province of Ontario through the Ministry of Research, Innovation and Science. The University of Virginia acknowledges support from NSF 2212550 OAC Core: Smart Surrogates for High Performance Scientific Simulations and DE-SC0023452: FAIR Surrogate Benchmarks Supporting AI and Simulation Research. JQTM acknowledges a Mitacs Elevate Postdoctoral Fellowship (IT39533) with Perimeter Institute for Theoretical Physics.

-
- [1] ATLAS Collaboration, *ATLAS software and computing HL-LHC roadmap*, Tech. Rep. (Technical report, CERN, Geneva. <http://cds.cern.ch/record/2802918>, 2022).
- [2] D. Rousseau, Experimental particle physics and artificial intelligence, in *Artificial Intelligence for Science: A Deep Learning Revolution* (World Scientific, 2023) pp. 447–464.
- [3] Michele Fauci Giannelli, Gregor Kasieczka, Claudius Krause, Ben Nachman, Dalila Salamani, David Shih, Anna Zaborowska, Fast calorimeter simulation challenge 2022 - dataset 1,2 and 3 [data set]. zenodo., <https://doi.org/10.5281/zenodo.8099322>, <https://doi.org/10.5281/zenodo.6366271>, <https://doi.org/10.5281/zenodo.6366324> (2022), online; accessed TO FILL.
- [4] S. Agostinelli, J. Allison, K. a. Amako, J. Apostolakis, H. Araujo, P. Arce, M. Asai, D. Axen, S. Banerjee, G. Barrand, *et al.*, Geant4—a simulation toolkit, Nuclear instruments and methods in physics research section A: Accelerators, Spectrometers, Detectors and Associated Equipment **506**, 250 (2003).
- [5] L. de Oliveira, M. Paganini, and B. Nachman, Learning particle physics by example: location-aware generative adversarial networks for physics synthesis, Computing and Software for Big Science **1**, 4 (2017).
- [6] ATLAS collaboration *et al.*, Fast simulation of the ATLAS calorimeter system with generative adversarial networks, ATLAS PUB Note, CERN, Geneva (2020).
- [7] E. Buhmann, S. Diefenbacher, E. Eren, F. Gaede, G. Kasieczka, A. Korol, and K. Krüger, Decoding photons: Physics in the latent space of a bib-ae generative network, in *EPJ Web of Conferences*, Vol. 251 (EDP Sciences, 2021) p. 03003.
- [8] C. Krause and D. Shih, Caloflow: fast and accurate generation of calorimeter showers with normalizing flows, arXiv preprint arXiv:2106.05285 (2021).
- [9] L. Favaro, A. Ore, S. P. Schweitzer, and T. Plehn, Calodream - detector response emulation via attentive flow matching, arXiv preprint arXiv:2405.09629 (2024).
- [10] V. Mikuni and B. Nachman, Caloscore v2: single-shot calorimeter shower simulation with diffusion models, Journal of Instrumentation **19** (02), P02001.
- [11] O. Amram and K. Pedro, Denoising diffusion models with geometry adaptation for high fidelity calorimeter simulation, Physical Review D **108**, 072014 (2023).
- [12] C. Krause, M. F. Giannelli, G. Kasieczka, B. Nachman, D. Salamani, D. Shih, A. Zaborowska, O. Amram, K. Borrás, M. R. Buckley, *et al.*, Calochallenge 2022: A community challenge for fast calorimeter simulation, arXiv preprint arXiv:2410.21611 (2024).
- [13] R. Kansal, A. Li, J. Duarte, N. Chernyavskaya, M. Pierini, B. Orzari, and T. Tomei, Evaluating generative models in high energy physics, Physical Review D **107**, 076017 (2023).
- [14] S. Hoque, H. Jia, A. Abhishek, M. Fadaie, J. Q. Toledomarin, T. Vale, R. G. Melko, M. Swiatlowski, and W. T. Fedorko, Caloqvae: Simulating high-energy particle-calorimeter interactions using hybrid quantum-classical generative models, The European Physical Journal C **84**,

- 1 (2024).
- [15] S. Gonzalez, H. Jia, J. Q. Toledo-Marin, S. Hoque, A. Abhishek, I. Lu, D. Sogutlu, S. Andersen, C. Gay, E. Paquet, R. Melko, G. Fox, M. Swiatlowski, and W. Fedorko, Calo4pqvae: Quantum-assisted 4-partite vae surrogate for high energy particle-calorimeter interactions, 2024 IEEE International Conference on Quantum Computing and Engineering (QCE) (2024).
- [16] K. Boothby, P. Bunyk, J. Raymond, and A. Roy, Next-generation topology of D-wave quantum processors, arXiv preprint arXiv:2003.00133 (2020).
- [17] Particle physics experiments typically use a right-hand coordinate system with the interaction point at the center of the detector and the z -axis pointing along the beam pipe. The x -axis points to the center of the accelerating ring, and the y -axis points upwards. Polar coordinates (r , ϕ) are used to describe the transverse directions of the detector, with ϕ being the azimuthal angle around the z -axis. The pseudorapidity η is defined relative to the polar angle θ as $\eta = -\ln \tan(\theta/2)$.
- [18] M. F. Giannelli, G. Kasieczka, C. Krause, B. Nachman, D. Salamani, D. Shih, and A. Zaborowska, Fast Calorimeter Simulation Challenge 2022 - Dataset 1, 10.5281/zenodo.8099322 (2023).
- [19] W. Winci, L. Buffoni, H. Sadeghi, A. Khoshaman, E. Andriyash, and M. H. Amin, A path towards quantum advantage in training deep generative models with quantum annealers, Machine Learning: Science and Technology **1**, 045028 (2020).
- [20] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, Convolutional sequence to sequence learning, in *International conference on machine learning* (PMLR, 2017) pp. 1243–1252.
- [21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, Attention is all you need, Advances in neural information processing systems **30** (2017).
- [22] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016) pp. 770–778.
- [23] D. P. Kingma and M. Welling, Auto-encoding variational bayes, arXiv preprint arXiv:1312.6114 (2013).
- [24] We removed the ξ coefficient due to normalization.
- [25] C. J. Maddison, A. Mnih, and Y. W. Teh, The concrete distribution: A continuous relaxation of discrete random variables, arXiv preprint arXiv:1611.00712 (2016).
- [26] L. Van der Maaten and G. Hinton, Visualizing data using t-sne., Journal of machine learning research **9** (2008).
- [27] J. Q. Toledo-Marin and J. A. Glazier, Using deep lsd to build operators in gans latent space with meaning in real space, Plos one **18**, e0287736 (2023).
- [28] V. Mnih, H. Larochelle, and G. E. Hinton, Conditional restricted Boltzmann machines for structured output prediction, arXiv preprint arXiv:1202.3748 (2012).
- [29] M. W. Johnson, M. H. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, *et al.*, Quantum annealing with manufactured spins, Nature **473**, 194 (2011).
- [30] J. Sakurai and J. Napolitano, *Modern Quantum Mechanics* (Cambridge University Press Cambridge, 2017) pp. 328–331.
- [31] T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar, and M. Troyer, Defining and detecting quantum speedup, science **345**, 420 (2014).
- [32] P. Hauke, H. G. Katzgraber, W. Lechner, H. Nishimori, and W. D. Oliver, Perspectives of quantum annealing: Methods and implementations, Reports on Progress in Physics **83**, 054401 (2020).
- [33] M. Benedetti, J. Realpe-Gómez, R. Biswas, and A. Perdomo-Ortiz, Estimation of effective temperatures in quantum annealers for sampling applications: A case study with possible applications in deep learning, Physical Review A **94**, 022308 (2016).
- [34] M. H. Amin, A. D. King, J. Raymond, R. Harris, W. Bernoudy, A. J. Berkley, K. Boothby, A. Smirnov, F. Altomare, M. Babcock, *et al.*, Quantum error mitigation in quantum annealing, arXiv preprint arXiv:2311.01306 (2023).
- [35] M. H. Amin, Searching for quantum speedup in quasi-static quantum annealers, Physical Review A **92**, 052323 (2015).
- [36] P. G. Debenedetti and F. H. Stillinger, Supercooled liquids and the glass transition, Nature **410**, 259 (2001).
- [37] J. Marshall, D. Venturelli, I. Hen, and E. G. Rieffel, Power of pausing: Advancing understanding of thermalization in experimental quantum annealers, Physical Review Applied **11**, 044083 (2019).
- [38] D-Wave Systems, Advantage processor overview, https://www.dwavesys.com/media/3xvdipecn/14-1058a-a_advantage_processor_overview.pdf (2022), accessed: 2023-11-07.
- [39] D-Wave Systems, Advantage: The first and only computer built for business, https://www.dwavesys.com/media/htjclcey/advantage_datasheet_v10.pdf (2023), accessed: 2023-11-07.
- [40] D-Wave Systems, D-wave system documentation, https://docs.dwavesys.com/docs/latest/doc_physical_properties.html#anneal-schedules (2023), accessed: 2023-11-07.
- [41] R. Harris, M. W. Johnson, T. Lanting, A. Berkley, J. Johansson, P. Bunyk, E. Tolkacheva, E. Ladizinsky, N. Ladizinsky, T. Oh, *et al.*, Experimental investigation of an eight-qubit unit cell in a superconducting optimization processor, Physical Review B **82**, 024511 (2010).
- [42] Raghav Kansal, Javier Duarte1, Carlos Pareja, Lint Action, Zichun Hao, mova, jet-net/jetnet: v0.2.3.post3, <https://doi.org/10.5281/zenodo.5597892> (2023), online; accessed July 2024.
- [43] R. Salakhutdinov, Learning and evaluating boltzmann machines, Utm1 Tr **2**, 21 (2008).
- [44] A. Decelle, C. Furtlehner, and B. Seoane, Equilibrium and non-equilibrium regimes in the learning of restricted boltzmann machines, Advances in Neural Information Processing Systems **34**, 5345 (2021).
- [45] K. Boothby, A. D. King, and J. Raymond, Zephyr topology of d-wave quantum processors, D-Wave Technical Report Series (2021).
- [46] J. Fernandez-de Cossio-Diaz, S. Cocco, and R. Monasson, Disentangling representations in restricted boltzmann machines without adversaries, Physical Review X **13**, 021003 (2023).
- [47] A. D. King, A. Nocera, M. M. Rams, J. Dziarmaga, R. Wiersema, W. Bernoudy, J. Raymond, N. Kaushal, N. Heinsdorf, R. Harris, *et al.*, Computational supremacy in quantum simulation, arXiv preprint arXiv:2403.00910 (2024).

- [48] G. E. Box and M. E. Muller, A note on the generation of random normal deviates, *The annals of mathematical statistics* **29**, 610 (1958).

Appendix A: Variational Autoencoder

In this section we describe the VAE framework first proposed by Kingma and Welling in [23]. Suppose we have a data set $\{\mathbf{x}^{(i)}\}_{i=1}^{|\mathcal{D}|}$, where each element in the data set lives in \mathcal{R}^N . The goal in training a Variational Autoencoder (VAE) on this data set is to fit a probability distribution, $p(\mathbf{x})$, to the data. This is done by maximizing the log-likelihood (LL) of $p(\mathbf{x})$ over the data set. A key component in generative models is the introduction of latent variables, \mathbf{z} , such that the joint distribution can be expressed as $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$, where $p(\mathbf{z})$ is the *prior* distribution of $\mathbf{z} \in \mathcal{R}^M$. VAEs are composed by an encoder and a decoder, and are trained using the Evidence Lower Bound (ELBO) as a proxy loss function for the LL. To understand the relationship between the LL and the ELBO, we first write the following identity:

$$\ln p_\theta(\mathbf{x}) = \langle \ln p_\theta(\mathbf{x}) \rangle_{q_\phi(\mathbf{z}|\mathbf{x})} \quad (\text{A1})$$

where $\langle \bullet \rangle_{q_\phi(\mathbf{z}|\mathbf{x})}$ denotes expectation value of \bullet over $q_\phi(\mathbf{z}|\mathbf{x})$. Here, $q_\phi(\mathbf{z}|\mathbf{x})$ is the encoding function, also known as the approximate posterior. This function encodes the data \mathbf{x} into \mathbf{z} in the latent space. We can further manipulate the r.h.s. in Eq. (A1), *viz.*,

$$\begin{aligned} \ln p_\theta(\mathbf{x}) &= \left\langle \ln \frac{p_\theta(\mathbf{x}, \mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})} \right\rangle_{q_\phi(\mathbf{z}|\mathbf{x})} \\ &= \left\langle \ln \frac{p_\theta(\mathbf{x}, \mathbf{z})q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})p_\theta(\mathbf{z}|\mathbf{x})} \right\rangle_{q_\phi(\mathbf{z}|\mathbf{x})} \\ &= \left\langle \ln \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right\rangle_{q_\phi(\mathbf{z}|\mathbf{x})} + \left\langle \ln \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})} \right\rangle_{q_\phi(\mathbf{z}|\mathbf{x})} \\ &= \mathcal{L}_{\phi, \theta}(\mathbf{x}) + D_{kl}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})). \end{aligned} \quad (\text{A2})$$

In the last line in the previous Eq., $\mathcal{L}_{\phi, \theta}(\mathbf{x})$ is the ELBO and $D_{kl}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))$ is the Kullback-Liebler (KL) divergence. The KL divergence is a positive functional and equals zero when both distributions are the same. Therefore:

$$\mathcal{L}_{\phi, \theta}(\mathbf{x}) = \ln p_\theta(\mathbf{x}) - D_{kl}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) \leq \ln p_\theta(\mathbf{x}) \quad (\text{A3})$$

The previous Eq. shows that maximizing the ELBO implies maximizing the LL (since the LL is the upper bound), as well as to minimizing the KL divergence between $q_\phi(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{z}|\mathbf{x})$.

We can express the ELBO in a more tractable way:

$$\begin{aligned} \mathcal{L}_{\phi, \theta}(\mathbf{x}) &= \left\langle \ln \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right\rangle_{q_\phi(\mathbf{z}|\mathbf{x})} \\ &= \left\langle \ln p_\theta(\mathbf{x}|\mathbf{z}) \right\rangle_{q_\phi(\mathbf{z}|\mathbf{x})} - \left\langle \ln \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z})} \right\rangle_{q_\phi(\mathbf{z}|\mathbf{x})}. \end{aligned} \quad (\text{A4})$$

The first term in the last equality is called the reconstruction term since it is a measure of how well the model is able to reconstruct the input \mathbf{x} from a latent vector \mathbf{z} . The second term in the last equality is called a regularizer and measures the divergence between the prior and the approximate posterior.

The legacy VAE [23] assumes the functional forms:

$$p_\theta(\mathbf{x}|\mathbf{z}) = \prod_{i=1}^N \frac{1}{\sqrt{2\pi\sigma_{\hat{x}_i}^2}} \exp\left(-\frac{(x_i - \hat{x}_i)^2}{2\sigma_{\hat{x}_i}^2}\right) \quad (\text{A5a})$$

$$p_\theta(\mathbf{z}) = \prod_{i=1}^M \frac{1}{\sqrt{2\pi}} \exp(z_i^2/2) \quad (\text{A5b})$$

$$q_\phi(\mathbf{z}|\mathbf{x}) = \prod_{i=1}^M \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(z_i - \mu_i)^2}{2\sigma_i^2}\right). \quad (\text{A5c})$$

This leads to the following expression for the ELBO:

$$\mathcal{L}_{\phi, \theta}(\mathbf{x}) = -\sum_{i=1}^N \langle (x_i - \hat{x}_i)^2 \rangle_{q_\phi(\mathbf{z}|\mathbf{x})} - \sum_{i=1}^M \frac{1}{2} (\mu_i^2 + \sigma_i^2 - 1 - \ln \sigma_i^2) + \text{const}. \quad (\text{A6})$$

To obtain the previous equality we assumed $\sigma_{\hat{x}} = \mathbf{1}$. We also used the fact that $\langle z_i^2 \rangle_{q_\phi} = \mu_i^2 + \sigma_i^2$. While the goal is to maximize the ELBO, it is common practice to minimize the negative ELBO during training. An important step in the context of VAEs is the so-called *reparameterization trick*. When dealing with a finite data set and optimizing the loss function we need to compute gradients of expectations with respect to distributions that may not be explicitly expressible. For instance, consider computing the gradient of an estimator:

$$\nabla_\phi \langle f_\phi(z) \rangle_{q_\phi(z)} = \nabla_\phi \int dz q_\phi(z) f_\phi(z) \quad (\text{A7})$$

$$\sim \nabla_\phi \sum_{z \sim q_\phi(z)} f_\phi(z) \quad (\text{A8})$$

The issue here is that taking the gradient over $\sum_{z \sim q_\phi(z)}$ is rather ill-defined because the samples are drawn from a distribution parameterized by ϕ . To circumvent this, the reparameterization trick consists in a change in variable in a way that makes the sampling process differentiable with respect to ϕ . This variable change needs to preserve the metric, *i.e.*, the distribution in the old variable and that in the new variable need to both be normalized:

$$\int dz q_\phi(z) = \int d\epsilon \rho(\epsilon) \implies \rho(\epsilon) = \left| \frac{dz}{d\epsilon} \right| q_\phi(z) \quad (\text{A9})$$

The simplest change in variable is $z = \mu + \sigma\epsilon$ with $\epsilon \sim \mathcal{N}(1, 0)$, which leads to

$$\nabla_\phi \sum_{z \sim q_\phi(z)} f_\phi(z) \rightarrow \sum_{\epsilon \sim \mathcal{N}(0,1)} \nabla_\phi f_\phi(\mu_\phi + \sigma_\phi \epsilon) \quad (\text{A10})$$

The *reparameterization trick* is merely a change in variable akin to that used in the Box-Muller method to generate Gaussian distributed random numbers from Uniform distributed random numbers [48]. This is quite often used in the context of deep generative models in order to be able to take the gradient over an estimator.

Appendix B: Discrete Variational Autoencoder

Discrete Variational Autoencoders (DVAEs) are a type of VAE where the latent space is discrete. The main two challenges with DVAEs are *i)* how does one backpropagate the gradient since the latent space is discrete? *ii)* what reparameterization can be employed to enable gradient-based optimization? To address the former, one can simply relax the discrete condition by introducing annealed sigmoids. Specifically, we replace the Heaviside function $\Theta(x)$ with the sigmoid function $\sigma(x\beta)$, where β is the annealing parameter. Notice that $\lim_{\beta \rightarrow \infty} \sigma(x\beta) = \Theta(x)$. To address the latter issue one can employ the *Gumbel trick*. The *Gumbel trick* has become an umbrella term which refers to a set of methods to sample from discrete probabilities or to estimate its partition function. In our case, we simply generate latent variables ζ via

$$\zeta = \sigma((l(\phi, x) + \sigma^{-1}(\rho))\beta), \quad (\text{B1})$$

where ρ is a uniform random number, and $l(\phi, x)$ is a logit, *i.e.*, the inverse of a sigmoid function, such that in the discrete regime of ζ (*i.e.*, $\beta \rightarrow \infty$) $P(\zeta = 1) = \sigma(l(\phi, x))$. Notice that in this approach, we generate the random variable ζ using a deterministic equation, σ ; a logit, $l(\phi, x)$; and a uniformly-distributed random number, ρ . The connection with Gumbel distributed random numbers is due to the fact that $\sigma^{-1}(\rho) \sim G_1 - G_2$, where G_1 and G_2 are two Gumbel distributed random numbers [25? ?].

Appendix C: Bipartite Restricted Boltzmann Machines

Suppose a data set $\{\mathbf{v}^{(i)}\}_{i=1}^{|\mathcal{D}|}$, and each element in the data set lives in $\{0, 1\}^N$. The goal behind training a Restricted Boltzmann Machine (RBM) over this data set consists on fitting a probability mass function, $p(\mathbf{v})$, that models the distribution of the data. This is achieved by maximizing the log-likelihood (LL) of $p(\mathbf{v})$ over the data set. We denote the joint probability of the dataset as $P_{\mathcal{D}} = (\prod_{\mathbf{v} \in \mathcal{D}} p(\mathbf{v}))^{1/|\mathcal{D}|}$. Maximizing the LL corresponds to:

$$\operatorname{argmax}_{\Omega} \ln P_{\mathcal{D}} \quad (\text{C1})$$

By design, $p(\mathbf{v})$ follows a Boltzmann distribution *viz.*

$$p(\mathbf{v}) = \frac{\sum_{\mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h}; \mathbf{a}, \mathbf{b}, \mathbf{W})}}{Z(\mathbf{a}, \mathbf{b}, \mathbf{W})}, \quad (\text{C2})$$

where $Z(\mathbf{a}, \mathbf{b}, \mathbf{W})$ is the partition function and $E(\mathbf{v}, \mathbf{h}; \mathbf{a}, \mathbf{b}, \mathbf{W})$ is the energy function defined as

$$E(\mathbf{v}, \mathbf{h}; \mathbf{a}, \mathbf{b}, \mathbf{W}) = - \sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} v_i W_{ij} h_j. \quad (\text{C3})$$

The parameters \mathbf{a} , \mathbf{b} and \mathbf{W} are fitting parameters and \mathbf{h} is called the hidden vector such that $\mathbf{h} \in \{0, 1\}^M$. Notice that the matrix \mathbf{W} couples the nodes in \mathbf{v} with the nodes in \mathbf{h} , while there are no explicit couplings between nodes in \mathbf{v} nor between nodes in \mathbf{h} , which is the same to say that the RBM is a bipartite graph.

Notice that

$$\frac{\partial E}{\partial w} = \begin{cases} -v_k & w = a_k, \\ -h_k & w = b_k, \\ -v_k h_l & w = W_{kl}. \end{cases} \quad (\text{C4})$$

Taking the derivative of the LL with respect to some generic parameter w yields:

$$\frac{\partial \ln P_{\mathcal{D}}}{\partial w} = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \left\langle -\frac{\partial E}{\partial w} \right\rangle_{p(\mathbf{h}|\mathbf{v}^{(i)})} - \left\langle -\frac{\partial E}{\partial w} \right\rangle_{p(\mathbf{v}, \mathbf{h})} \quad (\text{C5})$$

The previous simplifies to

$$\frac{\partial \ln P_{\mathcal{D}}}{\partial a_k} = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \langle v_k \rangle_{p(\mathbf{h}|\mathbf{v}^{(i)})} - \langle v_k \rangle_{p(\mathbf{v}, \mathbf{h})} \quad (\text{C6a})$$

$$\frac{\partial \ln P_{\mathcal{D}}}{\partial b_k} = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \langle h_k \rangle_{p(\mathbf{h}|\mathbf{v}^{(i)})} - \langle h_k \rangle_{p(\mathbf{v}, \mathbf{h})} \quad (\text{C6b})$$

$$\frac{\partial \ln P_{\mathcal{D}}}{\partial w_{kl}} = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \langle v_k h_l \rangle_{p(\mathbf{h}|\mathbf{v}^{(i)})} - \langle v_k h_l \rangle_{p(\mathbf{v}, \mathbf{h})} \quad (\text{C6c})$$

where

$$\langle \bullet \rangle_{p(\mathbf{h}|\mathbf{v}^{(i)})} = \frac{\sum_{\mathbf{h}} \bullet e^{-E(\mathbf{v}^{(i)}, \mathbf{h})}}{\sum_{\mathbf{h}} e^{-E(\mathbf{v}^{(i)}, \mathbf{h})}} \quad (\text{C7})$$

and

$$\langle \bullet \rangle_{p(\mathbf{v}, \mathbf{h})} = \frac{\sum_{\mathbf{v}, \mathbf{h}} \bullet e^{-E(\mathbf{v}, \mathbf{h})}}{\sum_{\mathbf{v}, \mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h})}}. \quad (\text{C8})$$

Since the number of \mathbf{v} and \mathbf{h} states are 2^N and 2^M , respectively, the number of terms in the summations in Eqs. (C7) and (h)eq:p(v,h) are $|\mathcal{D}| \times 2^M$ and 2^{N+M} , respectively. This exponential dependence on the dimensionality makes computing these averages intractable. To overcome this challenge, importance sampling is employed.

Notice that $q(\mathbf{h}|\mathbf{v}) = p(\mathbf{v}, \mathbf{h})/p(\mathbf{v})$, from which it is straightforward showing

$$q(\mathbf{h}|\mathbf{v}) = \prod_{j=1}^M q(h_j|\mathbf{v}), \quad (\text{C9})$$

where

$$q(h_j|\mathbf{v}) = \frac{e^{h_j C_j(\mathbf{v})}}{1 + e^{C_j(\mathbf{v})}} \quad (\text{C10})$$

and $C_j(\mathbf{v}) = \sum_i v_i W_{ij} + b_j$. Therefore, $q(h_j = 1|\mathbf{v}) = \sigma(C_j(\mathbf{v}))$. Conversely, $p(v_i = 1|\mathbf{h}) = \sigma(D_i(\mathbf{h}))$, with $D_i(\mathbf{h}) = \sum_j W_{ij} h_j + a_i$. Hence, we can employ the expressions $\sigma(D_i(\mathbf{h}))$ and $\sigma(C_j(\mathbf{v}))$ to perform Gibbs sampling.

We can further simplify the expectation values from Eq. (C7):

$$\langle v_k \rangle_{p(\mathbf{h}|\mathbf{v}^{(i)})} = v_k \quad (\text{C11a})$$

$$\langle h_j \rangle_{p(\mathbf{h}|\mathbf{v}^{(i)})} = \frac{e^{C_j(\mathbf{v})}}{1 + e^{C_j(\mathbf{v})}} \quad (\text{C11b})$$

$$\langle v_k h_j \rangle_{p(\mathbf{h}|\mathbf{v}^{(i)})} = v_k \frac{e^{C_j(\mathbf{v})}}{1 + e^{C_j(\mathbf{v})}} \quad (\text{C11c})$$

$$(\text{C11d})$$

Note that $p(\mathbf{v}, \mathbf{h}) = p(\mathbf{v}|\mathbf{h})p(\mathbf{h})$ and $p(\mathbf{v}, \mathbf{h}) = q(\mathbf{h}|\mathbf{v})p(\mathbf{v})$. Therefore, starting from an observed data point, \mathbf{v} , we can generate samples of the hidden units, \mathbf{h} , via $p(\mathbf{h}|\mathbf{v})$. These samples can then be used as prior samples to generate new samples of \mathbf{v} vectors. We can repeat the process a number of times K , called the number of Gibbs sampling steps. We denote this process as $(\mathbf{v}, \mathbf{h}) \sim [q(\mathbf{h}|\mathbf{v})p(\mathbf{v}|\mathbf{h})]^K$

Therefore, for very large K , we can estimate Eq. (h)eq:p(v,h) as:

$$\langle \bullet \rangle_{p(\mathbf{v}, \mathbf{h})} \simeq \frac{1}{N} \sum_{(\mathbf{v}, \mathbf{h}) \sim [q(\mathbf{h}|\mathbf{v})p(\mathbf{v}|\mathbf{h})]^K} \bullet \quad (\text{C12})$$

The Gibbs sampling number of steps ultimately should be larger than the mixing time. The mixing time will depend on the size of the RBM, the data set being used and, interestingly, it has been shown that as the number of updates during training increases, the Gibbs sampling number of steps must increase for the RBM to reach equilibrium and avoid getting stuck in a non-equilibrium state [44].

The standard procedure to train an RBM involves partitioning the data set \mathcal{D} into mini-batches \mathcal{D}_α , such that $\mathcal{D} = \cup_\alpha \mathcal{D}_\alpha$. Then the RBM parameters are updated by:

$$a_k^{(t)} = a_k^{(t-1)} + \eta \frac{\partial \ln P_{\mathcal{D}_\alpha}}{\partial a_k}, \quad (\text{C13a})$$

$$b_k^{(t)} = b_k^{(t-1)} + \eta \frac{\partial \ln P_{\mathcal{D}_\alpha}}{\partial b_k}, \quad (\text{C13b})$$

$$W_{kl}^{(t)} = W_{kl}^{(t-1)} + \eta \frac{\partial \ln P_{\mathcal{D}_\alpha}}{\partial W_{kl}}, \quad (\text{C13c})$$

where η is the learning rate. When performing importance sampling, it is common to generate Markov chains of the size of the mini-batch, $|\mathcal{D}_\alpha|$.

There are three primary methods for training RBMs in the literature. Each one mainly differs from the others in the manner in which the Markov chains are initialized. The simplest one correspond to the case where for each parameter update, the initial state is randomly sampled from a 1/2-Bernoulli distribution and is called *Rdm-K*, where K is the number for Gibbs sampling steps. Another way shown to yield more robust RBMs is called *Contrastive Divergence* (CD), whereby the Markov chain is initialized from a point in the dataset. Lastly, persistent contrastive divergence (PCD) is very similar to CD in the sense that the Markov chain is started using a data point in the data set for the first parameter update, while for the remaining parameter updates, the Markov chains are initialized using the last state in the previous parameter update. This is similar to the traditional way to sample from an Ising model when decreasing the temperature. Instead of restarting the Markov chain from a random state after each temperature update, the chain is restarted from the previous state before the temperature update.

Appendix D: High temperature gradient approximation

The previous section shows the derivation of the block Gibbs sampling Eqs. used to trained RBM. A quite common approach to training RBMs consists in replacing $\ln Z$ with the average energy before computing the gradient. The basis comes from noticing that the gradient of the logarithm of the partition function w.r.t. the RBM parameters is equal to the average value of the gradient of the energy w.r.t. the RBM parameters, *viz.*,

$$-\frac{\partial \ln Z}{\partial \phi} = \left\langle \frac{\partial E}{\partial \phi} \right\rangle \quad (\text{D1})$$

This expectation value is over the ensemble and it is approximated by an arithmetic average over samples obtained via block Gibbs sampling, *i.e.*,

$$\left\langle \frac{\partial E}{\partial \phi} \right\rangle \simeq \frac{1}{N} \sum_{\mathbf{z} \sim BGS} \frac{\partial E}{\partial \phi} = \frac{\partial}{\partial \phi} \frac{1}{N} \sum_{\mathbf{z} \sim BGS} E(\mathbf{z}). \quad (\text{D2})$$

The last equality is not general and only holds in certain scenarios, as we show here. In the following we show that this approximation corresponds to the high temperature gradient approximation where thermal energy is larger than typical spin interactions, such that the specific heat is zero and the only contribution to the entropy is configurational.

By definition, the average energy is given by $\langle E(\mathbf{z}) \rangle = \sum_{\mathbf{z}} E(\mathbf{z}) e^{-\beta E(\mathbf{z})} / Z(\beta)$. Deriving the energy w.r.t. some parameter ϕ the energy depends on leads to:

$$\frac{\partial}{\partial \phi} \langle E(\mathbf{z}) \rangle = \left\langle \frac{\partial E(\mathbf{z})}{\partial \phi} \right\rangle + \beta \left(\langle E(\mathbf{z}) \rangle \left\langle \frac{\partial E(\mathbf{z})}{\partial \phi} \right\rangle - \langle E(\mathbf{z}) \frac{\partial E(\mathbf{z})}{\partial \phi} \rangle \right) \quad (\text{D3})$$

Hence $\frac{\partial}{\partial \phi} \langle E(\mathbf{z}) \rangle = \left\langle \frac{\partial E(\mathbf{z})}{\partial \phi} \right\rangle$ implies the equality:

$$\langle E(\mathbf{z}) \rangle \left\langle \frac{\partial E(\mathbf{z})}{\partial \phi} \right\rangle - \langle E(\mathbf{z}) \frac{\partial E(\mathbf{z})}{\partial \phi} \rangle = 0. \quad (\text{D4})$$

Recall the specific heat relates to the second cumulant via $C_T = \frac{1}{kT^2} \sigma_E^2$. Notice that C_T depends on ϕ . The derivative of the second cumulant w.r.t. ϕ leads to

$$\frac{\partial \sigma_E^2}{\partial \phi} = \beta \left\langle \frac{\partial E(\mathbf{z})}{\partial \phi} \right\rangle \left(\langle E(\mathbf{z})^2 \rangle - 2 \langle E(\mathbf{z}) \rangle^2 \right) + 2 \left(\langle E(\mathbf{z}) \frac{\partial E(\mathbf{z})}{\partial \phi} \rangle - \langle E(\mathbf{z}) \rangle \left\langle \frac{\partial E(\mathbf{z})}{\partial \phi} \right\rangle \right) \quad (\text{D5})$$

$$- \beta \left(\langle E(\mathbf{z})^2 \frac{\partial E(\mathbf{z})}{\partial \phi} \rangle - 2 \langle E(\mathbf{z}) \rangle \left\langle E(\mathbf{z}) \frac{\partial E(\mathbf{z})}{\partial \phi} \right\rangle \right) \quad (\text{D6})$$

From the previous Eq. it is easy to show that when Eq. (D4) is satisfied, $\frac{\partial C_T}{\partial \phi} = 0$. In general, the previous occurs at very large temperatures where $C_T = 0$, *i.e.*, the energy of the system saturates such that increasing the temperature does not increase the energy. In such regime, the spins are uncorrelated and the entropy is solely defined by the logarithm of possible configurations.

Appendix E: Quadripartite RBM numerical verification

We considered 4-partite RBM with six nodes per partition. This setting allows for the explicit enumeration of all feasible states, facilitating the precise computation of the partition function. To rigorously assess the accuracy of our approach, we conducted a comparative analysis of the density of energy states. This entailed a direct comparison between the utilization of all feasible states and the implementation of the 4-partite Gibbs sampling method, as elaborated upon in Section II C. In Figure E1, we present a detailed visual comparison of the density of states obtained through both methodologies, across various iterations of the Gibbs sampling process. This comparison shows the convergence and consistency of these two approaches.

Appendix F: Quantum Annealers

1. Adiabatic Approximation

Here we derive the adiabatic approximation following [30], which is the theoretical foundation of quantum annealers. Let us suppose a time-dependent Hamiltonian $H(t)$, we denote the time-dependent eigenstates as $|n; t\rangle$ and the eigenvalues as $E_n(t)$, such that,

$$H(t)|n; t\rangle = E_n(t)|n, t\rangle, \quad (\text{F1})$$

which simply states that at any particular time t , the eigenstate and eigenvalue may change. Notice that one can write the general solution to Schrödinger's Eq., *viz.*,

$$i\hbar \frac{\partial}{\partial t} |\alpha; t\rangle = H(t)|\alpha; t\rangle \quad (\text{F2})$$

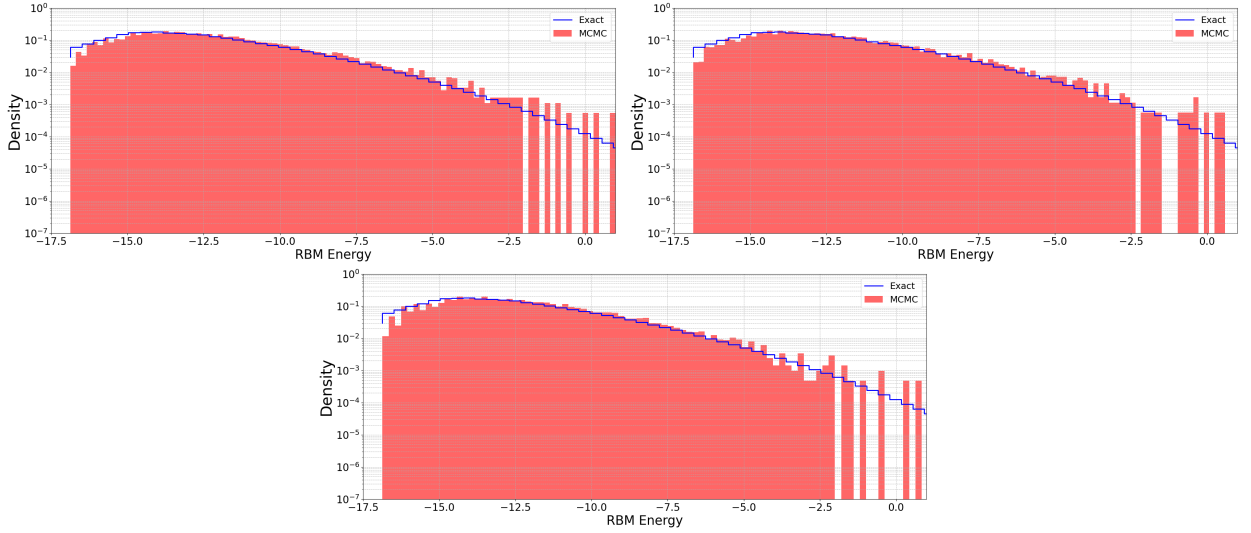


FIG. E1. Density of states for 4-partite RBMs with 6×4 nodes and $N = 10240$ samples, with Gibbs sampling steps set to (**upper left panel**) 200, (**upper right panel**) 500, and (**lower panel**) 3000 units.

as

$$|\alpha; t\rangle = \sum_n c_n(t) e^{i\theta(t)} |n; t\rangle \quad (\text{F3})$$

with

$$\theta_n(t) = -\frac{1}{\hbar} \int_0^t E_n(t') dt'. \quad (\text{F4})$$

By substituting Eqs. (F3) and (F4) in Eq. (F2) we reach the following equality:

$$\sum_n e^{i\theta_n(t)} \left[\dot{c}_n(t) |n; t\rangle + c_n(t) \frac{\partial}{\partial t} |n; t\rangle \right] = 0 \quad (\text{F5})$$

By taking the inner product w.r.t. $\langle m; t|$ and invoking orthonormality, yields the following differential equation for the time-dependent coefficients:

$$\dot{c}_m(t) = -\sum_n c_n(t) e^{i(\theta_n(t) - \theta_m(t))} \langle m; t| \frac{\partial}{\partial t} |n; t\rangle. \quad (\text{F6})$$

To get a better sense of the $\langle m; t| \frac{\partial}{\partial t} |n; t\rangle$ term, let us take the time derivative of the characteristic Eq. (F1):

$$\begin{aligned} \langle m; t| \frac{\partial}{\partial t} [H(t)|n; t\rangle = E_n(t)|n; t\rangle] &\implies \\ \langle m; t| \left[\dot{H}(t)|n; t\rangle + H(t) \frac{\partial}{\partial t} |n; t\rangle = \dot{E}_n(t)|n; t\rangle \right. & \\ \left. + E_n(t) \frac{\partial}{\partial t} |n; t\rangle \right] &\implies \end{aligned} \quad (\text{F7})$$

$$\langle m; t| \dot{H}(t)|n; t\rangle = (E_n(t) - E_m(t)) \langle m; t| \frac{\partial}{\partial t} |n; t\rangle \quad (\text{F8})$$

For $m \neq n$, we can write

$$\langle m; t| \frac{\partial}{\partial t} |n; t\rangle = \frac{\langle m; t| \dot{H}(t)|n; t\rangle}{E_n(t) - E_m(t)}. \quad (\text{F9})$$

Substituting the previous Eq. in Eq. (F6) leads to:

$$\dot{c}_m(t) = -c_m(t)\langle m; t | \frac{\partial}{\partial t} | m; t \rangle \quad (\text{F10})$$

$$- \sum_{n \neq m} c_n(t) e^{i(\theta_n(t) - \theta_m(t))} \frac{\langle m; t | \dot{H}(t) | n; t \rangle}{E_n(t) - E_m(t)}. \quad (\text{F11})$$

The previous Eq. shows that states with $n \neq m$ will mix with $|m; t\rangle$ due to the time dependence of the Hamiltonian. The adiabatic approximation consists in neglecting the mixing terms which correspond to the regime whereby

$$\frac{|\langle m; t | \dot{H}(t) | n; t \rangle|}{E_n(t) - E_m(t)} \equiv \frac{1}{\tau} \ll \langle m; t | \frac{\partial}{\partial t} | m; t \rangle \sim \frac{E_m}{\hbar}. \quad (\text{F12})$$

The previous gives us the condition where the adiabatic approximation holds, *i.e.*, that in which the timescale τ for changes in the Hamiltonian is much larger than the inverse of the characteristic frequency of the state phase factor. In such regime,

$$c_n(t) = e^{i\gamma(t)} c_n(0) \quad (\text{F13})$$

with

$$\gamma_n(t) \equiv i \int_0^t dt' \langle n; t' | \frac{\partial}{\partial t'} | n; t' \rangle \quad (\text{F14})$$

Notice that

$$0 = \frac{\partial}{\partial t} \langle n; t' | n; t' \rangle = \left[\frac{\partial}{\partial t} \langle n; t' | \right] | n; t' \rangle + \langle n; t' | \frac{\partial}{\partial t} | n; t' \rangle \quad (\text{F15})$$

which implies that

$$\left(\langle n; t' | \frac{\partial}{\partial t} | n; t' \rangle \right)^* = - \langle n; t' | \frac{\partial}{\partial t} | n; t' \rangle \quad (\text{F16})$$

Therefore, the integral argument is imaginary in which case $\gamma(t)$ is real. Hence, in the adiabatic approximation ($\tau \gg 1/\omega_n$), if the system starts out in eigenstate $|n; 0\rangle$, it will remain there since $c_n(t) = e^{i\gamma(t)} c_n(0)$ and $c_l(t) = 0$ for all $l \neq n$. Finally, it is important to stress that the adiabatic approximation does not correspond to short time regimes, *i.e.*, the time t is not relevant here but only the Hamiltonian change rate and the characteristic time of the state phase factor.

2. Dwave parameter mapping

This subsection shows the explicit parameter mapping between an RBM and a QA. Recall the quadripartite RBM energy function is:

$$\begin{aligned} E(\mathbf{v}, \mathbf{h}, \mathbf{s}, \mathbf{t}) = & -a_i v_i - b_i h_i - c_i s_i - d_i t_i \\ & -v_i W_{ij}^{(0,1)} h_j - v_i W_{ij}^{(0,2)} s_j \\ & -v_i W_{ij}^{(0,3)} t_j - h_i W_{ij}^{(1,2)} s_j \\ & -h_i W_{ij}^{(1,3)} t_j - s_i W_{ij}^{(2,3)} t_j, \end{aligned} \quad (\text{F17})$$

where we are using the double indices convention for summation. Since RBM data values are 0s and 1s, while qubits can take the values $\{-1, 1\}$, we map the quantum states to RBM states as:

$$\begin{pmatrix} \mathbf{v} \\ \mathbf{h} \\ \mathbf{s} \\ \mathbf{t} \end{pmatrix} = \frac{1}{2} \begin{pmatrix} \mathbf{z}_v + \mathbf{1} \\ \mathbf{z}_h + \mathbf{1} \\ \mathbf{z}_s + \mathbf{1} \\ \mathbf{z}_t + \mathbf{1} \end{pmatrix} \quad (\text{F18})$$

By substituting Eq. (F18) in Eq. (F17) and reading out the new couplings and biases, we obtain:

$$\begin{aligned}
\Delta_i &= -\frac{a_i}{2} \\
&\quad - \frac{1}{4} \left(\sum_j W_{ij}^{(01)} + W_{ij}^{(02)} + W_{ij}^{(03)} \right), i \in \Phi_0 \\
\Delta_i &= -\frac{b_i}{2} \\
&\quad - \frac{1}{4} \left(\sum_j (W^{(01)})_{ij}^t + W_{ij}^{(12)} + W_{ij}^{(13)} \right), i \in \Phi_1 \\
\Delta_i &= -\frac{c_i}{2} \\
&\quad - \frac{1}{4} \left(\sum_j (W^{(02)})_{ij}^t + (W^{(12)})_{ij}^t + W_{ij}^{(23)} \right), i \in \Phi_2 \\
\Delta_i &= -\frac{d_i}{2} \\
&\quad - \frac{1}{4} \left(\sum_j (W^{(03)})_{ij}^t + (W^{(13)})_{ij}^t + (W^{(23)})_{ij}^t \right), i \in \Phi_3 \\
J_{ij} &= -\frac{W_{ij}^{(\gamma,\delta)}}{4}, i \in \Phi_\gamma \text{ and } j \in \Phi_\delta
\end{aligned} \tag{F19}$$

where Φ_i denotes the partition i and $(\bullet)^t$ denotes transpose. In addition, this mapping introduces an energy offset, H_o , in the Hamiltonian

$$H_o = - \left(\frac{1}{2} \sum_i a_i + b_i + c_i + d_i + \frac{1}{4} \sum_j \sum_{\gamma < \delta} W_{ij}^{(\gamma,\delta)} \right), \tag{F20}$$

which we ignore since it does not contribute to the state probability distribution. After applying this transformation, we obtain the new RBM Hamiltonian:

$$H_{RBM} = \sum_i \Delta_i^{RBM} z_i + \sum_{ij} J_{ij}^{RBM} z_i z_j. \tag{F21}$$

3. DWave β_{QA} parameter estimation: Method 1

Let us assume two RBMs which we denote as QA and B , both, described by the same Hamiltonian at different temperatures, *viz.*:

$$P_{QA}(x) = \frac{e^{-\beta_{QA}H(x)}}{Z(\beta_{QA})}, \tag{F22}$$

$$P_B(x) = \frac{e^{-\beta H(x)}}{Z(\beta)}. \tag{F23}$$

We denote as β_{QA} and β the inverse temperatures of system QA and B , respectively. The Kullback-Liebler divergence associated to these two system yields:

$$D_{KL}(P_{QA}||P_B) = (\beta - \beta_{QA})\langle H \rangle_{QA} + \ln \frac{Z(\beta)}{Z(\beta_{QA})}, \tag{F24}$$

from which it is trivial to show that $\beta = \beta_{QA}$ yields zero in the KL divergence. The KL divergence derivative w.r.t. β yields

$$\frac{\partial D_{KL}}{\partial \beta} = \langle H \rangle_{QA} - \langle H \rangle_{B(\beta)}, \tag{F25}$$

where we have made explicit the β dependence of system B . We can fit β through gradient descent using the KL divergence, which leads to:

$$\beta_{t+1} = \beta_t - \eta (\langle H(x) \rangle_{QA} - \langle H(x) \rangle_{B(\beta)}) \quad (\text{F26})$$

The pseudo-algorithm to fit β using Eq. (F26) is:

1. Fix learning rate η and initialize β_0 . Parse the RBM parameters onto the Quantum Annealer via Eqs. (F19).
2. Sample from system QA RBM and from system B at temperature $1/\beta_0$.
3. Compute expectation of $H(x)$ using the samples from QA and from B , respectively.
4. Update β using Eq. (F26).
5. Repeat steps 2 to 4 until a convergence criterion is fulfilled..

Afterwards, $\beta_T \approx \beta_{QA}$. Therefore, rescaling the Hamiltonian by $1/\beta_T$ and then parsing the new Hamiltonian parameters onto the QA via Eqs. (F19) will ensure that we are effectively sampling from H . Notice that in the previous method, $\langle H(x) \rangle_{QA}$ is independent of β , hence in the previous algorithm we need to generate samples only from system B every time we update β . This can be rather inconvenient, for instance, in the case where our interest is in having the QA mimic B, where the latter is a trained RBM. Changing the temperature of system B to match that of QA will affect the performance of the model. We might be tempted to invert the method and fit β_{QA} , but this is not a viable approach, since one does not have control over β_{QA} let alone a measurement of it. Instead, one can do the following: Replace the original Hamiltonian with that scaled by β , *i.e.*, $H(x) \rightarrow H(x)/\beta$, which leads to:

$$\beta_{t+1} = \beta_t - \eta (\langle H(x) \rangle_{QA^{(r)}} - \langle H(x) \rangle_{B(1)}) \quad (\text{F27})$$

where $QA^{(r)}$ correspond to rescaling $H(x)$ by $1/\beta_t$. In reaching the previous equation, we redefined $\frac{\eta}{\beta_t} \rightarrow \eta$ where η is fixed. The pseudo-algorithm to fit β using Eq. (F27) is:

1. Fix learning rate η and initialize β_0 . Parse the RBM parameters onto the Quantum Annealer via multiplying Eqs. (F19) by $1/\beta_0$.
2. Sample from the QA RBM and from the B RBM at temperature 1.
3. Compute expectation of $H(x)/\beta_0$ using the samples from QA and from B , respectively.
4. Update β_1 using Eq. (F27).
5. Repeat steps 2 to 4 until a convergence criterion is fulfilled.

The previous method is one of the common approaches used to estimate the β_{QA} parameter in QAs. However, it can be slow to converge which is why we propose a simple mapping with a stable fixed point at β_{QA} . We describe the method in full detail in the following.

4. DWave β_{QA} parameter estimation: Method 2

Once again, let us assume two RBMs which we denote as QA and B , both, described by the same Hamiltonian at different temperatures, *viz.*:

$$P_{QA}(x) = \frac{e^{-\beta_{QA}H(x)}}{Z(\beta_{QA})}, \quad (\text{F28})$$

$$P_B(x) = \frac{e^{-\beta H(x)}}{Z(\beta)}. \quad (\text{F29})$$

We denote as β_{QA} and β the inverse temperatures of system QA and B , respectively. Now, let us denote as S_{QA} and S_B as the entropy of QA and B, respectively, and assume $S_{QA} = S_B$, from which after some straightforward algebra:

$$\beta = \beta_{QA} \frac{\langle H \rangle_{QA}}{\langle H \rangle_{B(\beta)}} + \frac{\ln \frac{Z(\beta_{QA})}{Z(\beta)}}{\langle H \rangle_{B(\beta)}}. \quad (\text{F30})$$

We can further simplify the previous expression by introducing the variable $\Delta\beta = \beta_{QA} - \beta$:

$$\beta = \beta_{QA} \frac{\langle H \rangle_{QA}}{\langle H \rangle_{B(\beta)}} + \frac{\ln \langle e^{-\Delta\beta H} \rangle_{B(\beta)}}{\langle H \rangle_{B(\beta)}}. \quad (\text{F31})$$

Notice that the r.h.s. of Eq. (F31) has a fixed point at $\beta = \beta_{QA}$. Here on we will only keep the first term in the r.h.s. and we will show that the fixed point is stable. In addition, same as we did when deriving the previous method, we replace $H(x) \rightarrow H(x)/\beta$. Since we do not have any control over β_{QA} nor we know the value *a priori*, we replace the prefactor in the first term of the r.h.s. with β since it does not affect the fixed point value and we further introduce a stability parameter $\delta (> 0)$. After the previous considerations, we propose the following mapping:

$$\beta_{t+1} = f_\delta(\beta_t) \equiv \beta_t \left(\frac{\langle H \rangle_{QA^{(r)}}}{\langle H \rangle_{B(1)}} \right)^\delta \quad (\text{F32})$$

The function f_δ has a fixed point at $\beta = \beta_{QA}$. The stability condition close to the fixed point correspond to $|f'_\delta(\beta_{QA})| < 1$. The first derivative at the fixed point yields:

$$|f'_\delta(\beta_{QA})| = \begin{cases} |1 + \frac{\sigma_{QA}^2}{\langle H \rangle_{B(1)}^2}|, & \delta = 1 \\ |1 + \delta \frac{\sigma_{QA}^2}{\langle H \rangle_{QA}}|, & \delta \neq 1. \end{cases} \quad (\text{F33})$$

In Fig. 5 we have plotted Eq. (F33) *vs* β for different values of δ . The values of β chosen for this plot correspond to where we typically find the fixed point. We call δ a stability parameter since we can tune it to stabilize the mapping per iteration.

A similar analysis can be done for the previous method. Specifically, the stability condition becomes:

$$\left| 1 - \frac{\sigma_{QA}^2}{\beta_{QA}/\eta} \right| < 1. \quad (\text{F34})$$

From the previous it is easy to notice that the fixed point is unstable when the learning rate, η , such that $\eta > \beta_{QA}/\sigma_{QA}^2$ ($\beta_{QA}/\sigma_{QA}^2 \sim 2 \cdot 10^{-2}$).

Appendix G: Incident energy conditioning

In the conditioned Calo4pQVAE framework, we condition the latent space RBM using the incident energy. We perform the condition as follows:

1. By applying a floor function on the incident energy in MeV, bin the incident energy, e , $e_{bin} = \text{floor}(e)$; the logarithm of the incident energy multiplied by 10, $e_{bin}^{\ln} = \text{floor}(10 \cdot \ln e)$; and the square root of the incident energy multiplied by 10, $e_{bin}^{\sqrt{e}} = \text{floor}(10 \cdot \sqrt{e})$.
2. Convert to binary number the three previous binned numbers, $B_e = \text{binary}(e_{bin})$, $B_{\ln e} = \text{binary}(e_{bin}^{\ln})$, $B_{\sqrt{e}} = \text{binary}(e_{bin}^{\sqrt{e}})$. We allocate 20 bits for each of these binary numbers.
3. Concatenate the three binary numbers, $B = \text{cat}(B_e, B_{\ln e}, B_{\sqrt{e}})$.
4. Use one partition to fit as many repetitions of the concatenated binary number B .
5. Set residual nodes to zero.

We fixed the number of nodes per partition to 512, hence the binary number B fits 8 times and the number of residual nodes is 32.

Appendix H: Gaussian approximation to shower logits

In the main text we describe the data transformation used to train our model, where we first reduce the voxel energy per event by dividing it by the incident energy and we afterwards construct logits based on the reduced energy

random variable. The number of particles in the electromagnetic shower follows approximately a Poisson distribution. Furthermore, via the saddle point approximation, for large number of particles in the shower the multivariate Poisson distribution becomes a multivariate Gaussian distribution with the mean equal to the variance. Here we show that to zeroth approximation, the logits are Gaussian distributed.

Let us consider a Gaussian positive distributed random variable r with mean and variance Λ , *i.e.*,

$$f(r) = \frac{\mathcal{N}(r|\Lambda, \Lambda)}{\Omega}, \forall r > 0 \quad (\text{H1})$$

where Ω is a normalization constant. We define $u = \ln \frac{x}{1-x}$ with $x = r/R$ and $R \gg r$. To zeroth order approximation, $u \approx \ln r - \ln R$. To obtain the distribution of u we first introduce an auxiliary random variable $z = \ln r$ with distribution $g(z)$. By equating the cumulatives of r and z we obtain:

$$g(z) = e^z f(e^z). \quad (\text{H2})$$

The distribution of u , $h(u)$, is simply the distribution of z shifted by $\ln R$, namely, $h(u) = g(u + \ln R)$:

$$h(u) = \frac{R e^u}{\Omega} \frac{1}{\sqrt{2\pi\Lambda}} e^{-\frac{(R e^u - \Lambda)^2}{2\Lambda}}. \quad (\text{H3})$$

The previous distribution is highly sensitive to u and the main contribution comes from $R e^u \approx \Lambda$. Hence, we can expand $\ln r$ around $\ln \Lambda$:

$$\ln r \approx \ln \Lambda + \frac{r - \Lambda}{\Lambda}, \quad (\text{H4})$$

which translates to $u \approx \ln \frac{\Lambda}{R} + \frac{r - \Lambda}{\Lambda}$. Notice that since $\frac{r - \Lambda}{\Lambda} \sim \mathcal{N}(0, \frac{1}{\Lambda})$, then:

$$u \sim \mathcal{N}\left(\ln \frac{\Lambda}{R}, \frac{1}{\Lambda}\right). \quad (\text{H5})$$