Completely quantum neural networks

Steve Abel[*]
Institute for Particle Physics Phenomenology, Durham University, Durham DH1 3LE, United Kingdom and Department of Mathematical Sciences, Durham University, Durham DH1 3LE, United Kingdom

Juan C. Criado[†] and Michael Spannowsky[‡]
Institute for Particle Physics Phenomenology, Durham University, Durham DH1 3LE, United Kingdom and Department of Physics, Durham University, Durham DH1 3LE, United Kingdom

(Received 27 April 2022; accepted 19 July 2022; published 1 August 2022)

Artificial neural networks are at the heart of modern deep learning algorithms. We describe how to embed and train a general neural network in a quantum annealer without introducing any classical element in training. To implement the network on a state-of-the-art quantum annealer, we develop three crucial ingredients: binary encoding the free parameters of the network; polynomial approximation of the activation function; and reduction of binary higher-order polynomials into quadratic ones. Together, these ideas allow encoding the loss function as an Ising model Hamiltonian. The quantum annealer then trains the network by finding the ground state. We implement this for an elementary network and illustrate the advantages of quantum training: its consistency in finding the global minimum of the loss function and the fact that the network training converges in a single annealing step, which leads to short training times while maintaining a high classification performance. After training the network using a quantum annealer, one can then use the quantum network weights in a classical network algorithm of identical design for inference. Our approach opens an avenue for the quantum training of general machine learning models.

DOI: 10.1103/PhysRevA.106.022601

I. INTRODUCTION

Neural networks (NN) have become an important machine learning tool, in particular for classification tasks, and there is great interest in improving their performance using quantum computing techniques. Indeed a NN contains three essential features that one might seek to enhance this way, namely,

(1) an adaptable system that approximately encodes a complicated function,

(2) a loss function in the output layer whose minimization defines the task the NN algorithm should perform,

(3) a training algorithm that minimizes the loss function.

To date, it has been possible to enhance one or more of these three aspects using quantum computing. For example, the encoding in (1) can be implemented on a quantum gate and continuous variable quantum devices [1–10]. The loss function can also be implemented in an entirely quantum fashion [11–13] (as a function of the network outputs), and the minimization of the loss function can be implemented on quantum devices in several ways, in particular using quantum annealers [14–19]. The result has been hybrid quantum or classical implementations that nevertheless often demonstrated some improvement. However, it has to date not been possible to encode all three of these aspects, in other words, to implement an entire NN algorithm onto a single quantum device with no classical elements at all.

The purpose of this work is to implement such a completely quantum NN to solve the task of binary classification of certain well-known (and not so well-known) data sets and investigate how it compares with classical devices.

Our purpose in this implementation is somewhat different from previous studies. We envisage the NN conventionally as simply a gigantic function that we wish to optimize during a training phase by adjusting the weights and biases in the network. Our objective is to achieve this training in a quantum way in one step by encoding the network in its entirety on a quantum device. Thus our implementation must incorporate several aspects that have not yet been combined. Of primary importance are the following. First, the network must include nontrivial activation functions for the weights and biases. These activation functions, along with the weights and biases, must somehow be incorporated into the network by encoding onto the quantum device. Conversely, however, to be effective and competitive, the training stage must be able to utilize virtually unlimited data (or at least a data set that can easily be larger than a number of qubits on any device currently in existence). This means that one should avoid trying to encode the data itself onto the quantum device (otherwise, a one-step training would not be possible) but should include all the data directly in the loss function. We will produce practically applicable NNs by paying attention to these two essential requirements.

The device we utilize for this task is a quantum annealer [20–32]; that choice being dictated solely by the large numbers of qubits required to encode the NN. Indeed as we shall see, the main limitations of the method are the number

[*]steve.abel@durham.ac.uk
[†]juan.c.criado@durham.ac.uk
[‡]michael.spannowsky@durham.ac.uk

of qubits required to encode the network itself, which restricts the number of features, and the number and size of hidden layers. However, even working within these restrictions we will be able to show that the method works effectively and to illustrate the two main advantages of the quantum training: that it can consistently find the global minimum of the loss function and that this can be done in a single training step, as opposed to the iterative procedures commonly used classically. The general method we describe here will be applicable more generally as the technology develops.

II. QUANTUM ANNEALING AND THE ISING MODEL ENCODING

Let us begin with a brief description of the device and the central features for this study. Generally a quantum annealer performs a restricted set of operations on a quantum system described by a Hilbert space that is the tensor product of several two-dimensional Hilbert spaces (i.e., the qubits) and a Hamiltonian of the form

$$\mathcal{H}(s) = A(s)\sum_{\ell}\sigma_{\ell,x} + B(s)\left(\sum_{\ell}h_{\ell}\sigma_{\ell,z} + \sum_{\ell m}J_{\ell m}\sigma_{\ell,z}\sigma_{m,z}\right), \tag{1}$$

where $\sigma_{\ell,x}$ and $\sigma_{\ell,z}$ are the corresponding Pauli matrices acting on the ℓth qubit, and $A(s)$, $B(s)$ are smooth functions such that $A(1) = B(0) = 0$ and $A(0) = B(1) = 1$, which are used to change the Hamiltonian during the anneal. The annealer can perform the following operations.

(1) Set an initial state that is either the ground state of $\mathcal{H}(0)$ (known as *forward annealing*) or any eigenstate of $\bigotimes_{\ell}\sigma_{\ell,z}$ (*backward annealing*).

(2) Fix the internal parameters h_{ℓ} and $J_{\ell m}$ of the Hamiltonian $\mathcal{H}(s)$.

(3) Allow the system to evolve quantum mechanically while controlling s as a piecewise-linear function $s(t)$ of time t, with $s(t_{\text{final}}) = 1$, and $s(t_{\text{init}}) = 0$ for forward annealing, or $s(t_{\text{init}}) = 1$ for backward annealing, where t_{init} and t_{final} are the initial and final times. The function $s(t)$ is called the *annealing schedule*.

(4) Measure the observable $\bigotimes_{\ell}\sigma_{\ell,z}$ at $t = t_{\text{final}}$.

Typically one chooses an annealing schedule such that the machine returns the ground state of the Ising-model Hamiltonian $\mathcal{H}(1)$. This allows one to use it to solve optimization problems that can be formulated as the minimization of a quadratic function H of spin variables $\sigma_{\ell} = \pm 1$:

$$H(\sigma_{\ell}) = \sum_{\ell}h_{\ell}\sigma_{\ell} + \sum_{\ell m}J_{\ell m}\sigma_{\ell}\sigma_{m}. \tag{2}$$

To differentiate between the physical system and the embedded abstract problem, we will refer to the elements of the physical system $\mathcal{H}(\sigma_{\ell,z})$ as *qubits* and to the ones of the abstract system $H(\sigma_{\ell})$ as *spins*. Note that the classical spin values σ_{ℓ} do not carry a z index. The objective is usually that the minimization of the physical Hamiltonian $\mathcal{H}(\sigma_{\ell,z})$ should yield a solution to the problem encoded by the minimization of the problem Hamiltonian $H(\sigma_{\ell})$.

Two crucial practical elements will need to be tackled to proceed with the encoding of a NN. The first is that Eq. (2) describes a generalized Ising model, but in practice, the quantum-annealing device only allows the setting of a limited number of nonvanishing couplings $J_{\ell m}$ between qubits. Let us be specific to the architecture we will be using in this work, namely D-Wave's [32] ADVANTAGE_SYSTEM4.1: this annealer contains 5627 qubits, connected in a PEGASUS structure, but only has a total of 40 279 couplings between them. Ising models with a higher degree of connectivity (more couplings) must be *embedded* into the physical system by chaining several qubits together with large couplings between them and treating the chain as if it were a single qubit. This step is carried out by an embedding algorithm.

The second aspect that we will need to address to treat NNs is the fact that the functions we will need to optimize [i.e., our problem Hamiltonians $H(\sigma_{\ell})$] are polynomial in spins, but the Ising model is only quadratic. Routines to reduce polynomial spin models to quadratic ones using auxiliary qubits (such as MAKE_QUADRATIC) do exist in the DIMOD package, but our experience with these was limited. Being able to treat higher-order problems is an essential extension to quantum annealers: therefore, in the Appendix, we prove that the issue of minimizing *any* higher-order polynomial in binary variables can be transformed into the minimization problem of a generalized quadratic Ising model of the form in Eq. (2), and is thus in principle solvable by a quantum annealer.

III. ENCODING A QUANTUM NEURAL NETWORK

A. Neural networks and classical training

A NN is a highly versatile machine-learning model built as a composition of functions with a vector input and output, known as layers. Each layer consists of a linear transformation followed by element-wise application of nonlinear functions g, known as the *activation function*. The ith component of the output of a layer L is thus given by

$$L_i(x) = g\left(\sum_j w_{ij}x_j + b_i\right), \tag{3}$$

where the w_{ij} and b_i are free parameters, the so-called *weights* and *biases*. Each function L_i of the vector input and scalar output is known as a *unit*. A NN is then defined by a collection of layers $L^{(k)}$ through

$$Y = L^{(n)} \circ \ldots \circ L^{(0)}. \tag{4}$$

A schematic representation of the NN structure is displayed in Fig. 1. The *depth* of a NN is the number of layers n, while its *width* is the number of units per layer (or in the largest layer if the layers vary in size). The various versions of the *universal approximation theorem* [33–35] ensure that NNs that are either sufficiently deep or sufficiently wide can approximate any function with arbitrary precision. This makes them suitable for general regression and classification tasks, on which they have proven to be an efficient parametrization.

The *training* of a NN is the procedure by which its internal parameters $w_{ij}^{(k)}$ and $b_i^{(k)}$ are adjusted so that it solves the problem at hand. This is done utilising a *loss function* $\mathcal{L}(Y)$, chosen such that a NN with the desired properties sits at its minimum. Typically a classical training algorithm will implement an improved version of gradient descent on $\mathcal{L}$ filled with

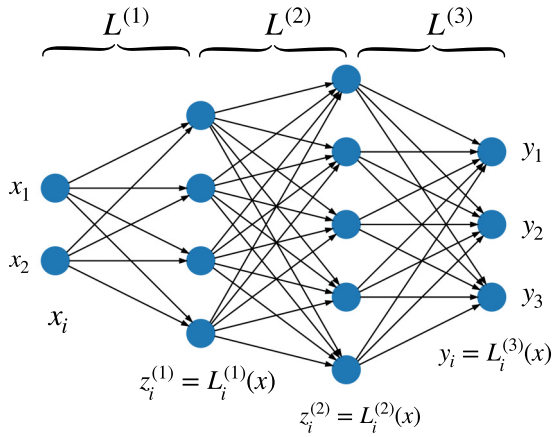

FIG. 1. Schematic representation of a NN with two inputs and three layers $L^{(1)}$, $L^{(2)}$, and $L^{(3)}$, with four, five, and three units, respectively. The inputs and units are shown as blue circles. The arrows represent the action of the layers, which are affine transformations followed by the element-wise application of the activation function, as displayed in Eq. (3). The outputs of layers $L^{(1)}$ and $L^{(2)}$ are denoted $z^{(1)}$ and $z^{(2)}$

training data to find this optimum configuration of weights and biases.

In the *supervised learning* framework, the input data for the training consists of a collection of N_d data points $x_a \in \mathbb{R}^{N_f}$ and a collection of the N_d corresponding outputs $y_a \in \mathbb{R}^{N_o}$ to be reproduced as $y_a \simeq Y(x_a)$. The dimension N_f of the input data-point space is known as the *number of features*. In general, when the outputs y_a are arbitrary points in a vector space $\mathbb{R}^{N_o}$, the problem to be solved is a *regression* one. In the particular case in which y_a takes values in a discrete set, then one has a *classification* problem, but the set of abstract labels can still be encoded as a set of isolated points in $\mathbb{R}^{N_o}$. For example if we seek a set of N_o yes or no decisions then the outputs live in $y_a \in (\mathbb{Z}_2)^{N_o} \subset \mathbb{R}^{N_o}$. Thus both kinds of problems can be treated within the general framework described here.

A typical loss function for supervised learning is the mean squared error (MSE) for the outputs

$$\mathcal{L}(Y) = \frac{1}{N_d} \sum_a |y_a - Y(x_a)|^2. \tag{5}$$

This is widely used for general regression problems. It is also a viable candidate for classification, although, depending on the training method, other loss functions, such as the binary or categorical cross entropy, can be more effective.

B. Training a NN in a quantum annealer

Let us now consider the task at hand, namely how to encode and train such a NN on a quantum annealer. Since the purpose of an annealer is to find the minimum of a function, the Hamiltonian, we aim to write the loss function $\mathcal{L}$ as an Ising-model Hamiltonian. Then, we expect the final state of the annealing process to give the optimal NN for the problem under consideration.

The loss is ultimately a function of the internal parameters of the NN, the weights w_{ij}, and biases b_i. Meanwhile the Ising-model Hamiltonian $H(\sigma_\ell)$ is a function of the Ising-model spins σ_ℓ. Therefore, as a first step, we need a translation between the w_{ij}, b_i parameters and the σ_ℓ spins. It is simpler for this purpose to use a quadratic unconstrained binary optimization (QUBO) encoding, related to the spin encoding as

$$\tau_\ell = \tfrac{1}{2}(\sigma_\ell + 1), \tag{6}$$

where $\tau_\ell = 0, 1$. Then, each of the parameters $p \sim w_{ij}^{(k)}, b_i^{(k)}$ is encoded in a binary fashion in terms of the annealer spins as

$$p = -1 + \frac{1}{1 - 2^{-N_b}} \sum_{\alpha=0}^{N_b-1} 2^{-\alpha} \tau_\alpha^p. \tag{7}$$

We will use a superindex p on the τ to indicate which particular block of N_b qubits (labeled by $\alpha = 0 \ldots N_b - 1$) is being used to encode that weight or bias. The above encoding yields $p \in [-1, 1]$.

Using Eq. (7), we can write the loss as a function of the Ising model spins τ. In general, this will not take the form of an Ising-model Hamiltonian [defined in Eq. (2)], so the next step is to transform it into one. For this purpose, we first approximate the activation function by a polynomial. Since the weights and biases are bounded in this approach, the input to the activation is bounded, and therefore a polynomial can approximate it arbitrarily well in the input range. Some polynomial approximations to standard loss functions are shown in Fig. 2. The use of polynomials as activation functions is a delicate issue because some versions of the universal approximation theorem require the activation to be nonpolynomial. In the present context, this need not concern us, however, since the boundedness of the input implies here that there is boundedness of the output, and this, together with nonlinearity, is enough to guarantee universal approximation for NNs with a single hidden layer [34]. Moreover, the nonlinearity of the activation functions is sufficient for the universal approximation property to hold for neural networks of fixed width and arbitrary depth [36]. More generally, given any traditional NN, it can be approximated to arbitrarily good precision within our framework by increasing the number of spins per parameter and the degree of the polynomial activations. Thus, any universal approximation result for the given NN will similarly hold for its approximate version.

Apart from their universal approximation properties, activation functions are commonly selected because they generate well-behaved gradients, which are crucial for classical training algorithms. In our case, these properties are irrelevant since the gradients are not used in any way.

The loss function $\mathcal{L}$ of the output value $Y(x)$ of the NN can either be a polynomial, as in Eq. (5), or not. In the case it is not, we make use again of the boundedness of $Y(x)$: if we make an approximation for $\mathcal{L}(Y)$, it only needs to be valid in a bounded domain. It follows that polynomials are enough for this approximation to achieve arbitrary precision. Finally, we show a polynomial approximation to a typical loss function for classification: the logistic loss, at the bottom plot of Fig. 2. We arrive, therefore, with the polynomial activation functions at a loss function that is a polynomial in the Ising-model spins.

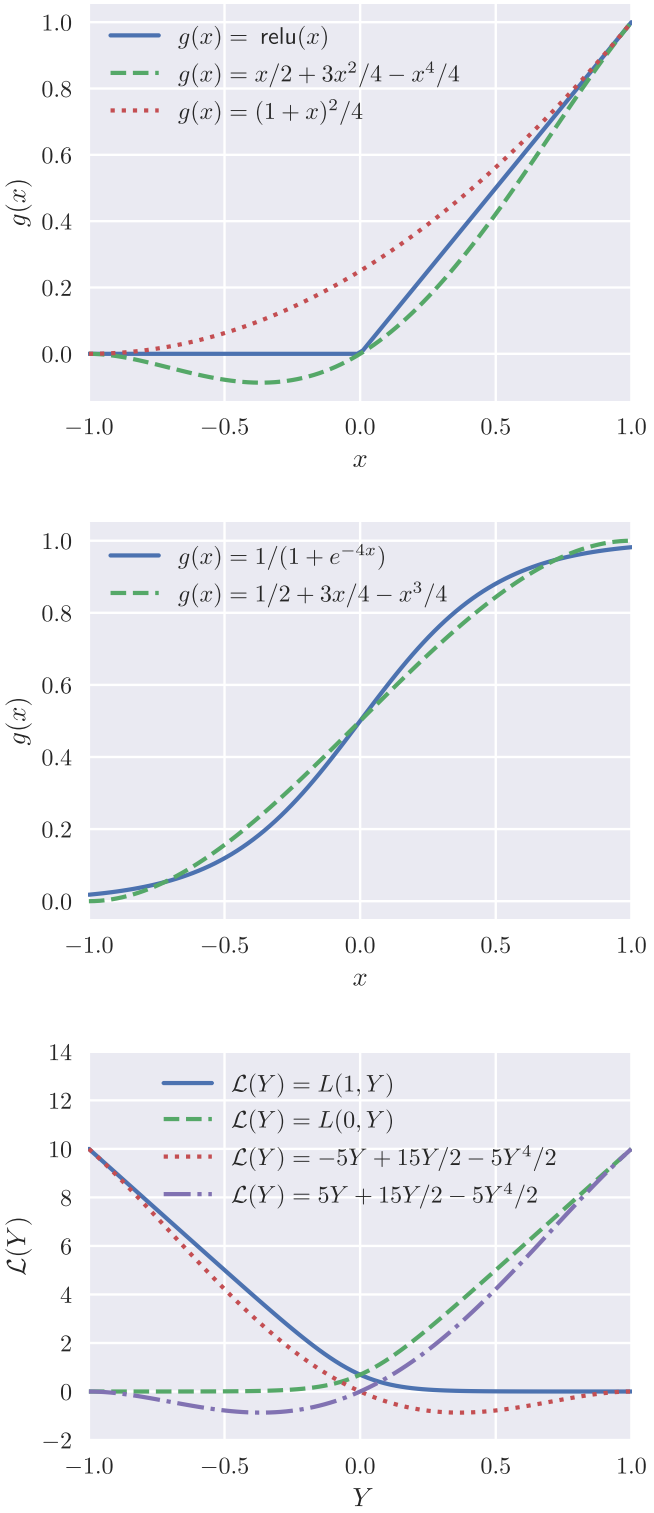

FIG. 2. Polynomial approximations to the popular ReLU (top) and sigmoid (middle) activation functions, and to the logistic loss function L (bottom). The logistic loss is defined here as $L(y, Y) = -y \log p(Y) - (1-y)\log[1-p(Y)]$, where $p(x) = 1/(1+e^{-10x})$.

The final step is to transform this polynomial into a quadratic, which would then match the form of the Ising-model Hamiltonian defined in Eq. (2). To do this, we may now employ the reduction method derived in the Appendix, which can, in principle, be used to reduce a polynomial of any degree to a quadratic. The reduction method makes iterated use of the following quadratic polynomial in binary variables $x, y, z = 0, 1$:

$$Q(z; x, y) = \Lambda[xy - 2z(x+y) + 3z]. \tag{8}$$

As discussed in the Appendix, this polynomial has degenerate global minima, which are achieved for every possible value of the (x, y) pair, if and only if $z = xy$, and one can check that $Q(xy; x, y) = 0$ at these minima. Thus Q can be used as a *constraint Hamiltonian* because reaching its minimum implies that the $z = xy$ constraint is satisfied, at which point there is no net contribution to the Hamiltonian. We can then, in principle, reduce the degree of the loss function polynomial by replacing products xy of spins with auxiliary spins z and simultaneously adding $Q(z; x, y)$ to the Hamiltonian with a sufficiently large value of Λ.

This completes the general method for encoding the loss function into an Ising-model Hamiltonian. Before we consider specific examples let us summarize the steps we took in the encoding.

(1) Write the loss as a function of the Ising model spins by means of Eq. (7).

(2) Rewrite it as a general polynomial in the spins by approximating the activation functions (and possibly the final operation in the loss function) by polynomials.

(3) Transform it to a quadratic polynomial by introducing auxiliary variables and adding copies of the constraint Hamiltonian defined in Eq. (A2).

Following these steps provides an Ising-model Hamiltonian whose ground state corresponds to the global minimum of the loss function. A quantum annealer can then be used to find its ground state. The weights of the corresponding classical NN can be recovered from the ground state using Eq. (7). One can then use the resulting NN for classical inference. For its activation functions, either one can directly use the polynomials that appear in the encoding, which guarantees that the NN is the optimal one for the given dataset, or one can apply the original nonpolynomial activation functions, in which case optimality depends on the quality of the approximation. Finally, we remark that, although the training is quantum, inference can be performed purely classically. In particular, properties such as universal approximation are unaffected by the quantum nature of the training and depend only on the classical approximations made.

C. Example encoding

Let us now illustrate the procedure outlined with a concrete example. We consider a binary classification problem, with labels $y = \pm 1$. We will use a NN with a single hidden layer (two layers in total) without an activation function for the last layer to classify any input data. In detail, such a NN produces a classification output from the inputs, which, following Eq. (3), is as follows:

$$Y_{v,w}(x_j) = v_i g(w_{ij} x_j) + v_0, \tag{9}$$

where the w_{ij} and v_i are the weights for first and second layers, respectively; summation over the i, j indices is implicit; and $i = 1 \ldots N_h$ labels the N_h units in the hidden layer. The x_j are

assumed to contain a constant feature incorporated by adding a 0 index for biases, so that w_{i0} is the bias in the first layer, and v_0 is the bias in the hidden layer, while w_{ij} are the weights. To reduce clutter we can also express the hidden layer bias v_0 by adding a w_{00} weight as well, so that the combined system of weights and biases is encompassed by simply extending all the sums to $i = 0 \ldots N_h$, $i = 0 \ldots N_f$. With this shorthand being understood, we will therefore write

$$Y_{v,w}(x_j) = v_i g(w_{ij} x_j). \tag{10}$$

A given state of the NN has a certain w and v, and the result of inputting data x_j is a single output $Y_{v,w}(x_j)$ that is used to decide its classification. The prediction is $y = 1$ if $Y_{v,w}(x) > 0$ and $y = -1$ otherwise. (Or, to put it another way, there is a final Heaviside activation function, $y = 2\vartheta[Y_{v,w}(x)] - 1$ feeding into the binary classification y.)

As mentioned, the crux of the matter is now to optimize v and w by training the NN on data. To perform this optimization, we need to feed into the NN a set of input data x_{ai}, where a labels the data points, and to optimize the weights and biases to give the best match with the known classifications $y_a = \pm 1$ that correspond to this data. We will use Eq. (5) as the loss function.

For the activation function, consider the simple approximation to the rectified linear activation (ReLU) function (shown in Fig. 2) given by $g(x) = (1 + x)^2/4$. We first use Eq. (7) to encode the weights, and then this results in a loss function that is sextic in spins. Since it is a total square, it is more straightforward to reduce the loss function to a quadratic by eliminating pairs of spins in

$$Y_{v,w}(x) = y_a - \tfrac{1}{4} - \tfrac{1}{2} v_i w_{ij} x_{aj} - \tfrac{1}{4} v_i w_{ij} w_{ij'} x_{aj} x_{aj'}, \tag{11}$$

with the use of auxiliary variables as described in Sec. III B, until $Y_{v,w}(x)$ becomes a linear function of the spins.

To replace the $v_i w_{ij}$ term, for example, for every quadruple $\alpha i; \beta j$ we trade the pairs of binary variables that appear in the product with auxilliary variables by adding

$$H_{vw} = \sum_{i,j} Q(\tau_{\alpha\beta}^{(ij)}; \tau_\alpha^{v_i} \tau_\beta^{w_{ij}}). \tag{12}$$

This requires

$$N_{vw} = (N_h + 1)(N_f + 1) N_b^2$$

auxiliary qubits, so that, for example, $N_h = N_f = 2$ and $N_b = 1$ requires only nine auxiliary qubits. We may then go on to replace the terms in $v_i w_{ij} w_{ij'}$, by adding for every sextuple $\alpha i; \beta j; \gamma j'$ the constraint Hamiltonian

$$H_{vww} = \sum_{i,j,j'} Q\big(\tau_{\alpha\beta\gamma}^{(ijj')}; \tau_{\alpha\beta}^{(ij)} \tau_\gamma^{w_{ij}}\big), \tag{13}$$

requiring a further

$$N_{vww} = (N_h + 1)(N_f + 1)^2 N_b^3$$

auxiliary qubits, and so forth for higher terms in the approximated activation function if included. It should be noted that the number of qubits grows geometrically with the degree of the term being reduced.

IV. IMPLEMENTATION AND RESULTS

Let us now demonstrate that using the encoding discussed in Sec. III one can train a NN in a D-Wave quantum annealer. Since the encoding requires a high degree of connectivity between spins, only small networks can be currently implemented. To reduce the number of qubits needed to embed the network in the annealer, we fix the biases for the first layer to zero and set the activation function to $g(x) = x^2$. The resulting model retains all the building blocks described in Sec. III: binary encoding of the network parameters; polynomial activation function; and encoding of products through a constraint Hamiltonian.

As the embedding is a function of the connectedness of the final model, and therefore rather hard to quantify, we first perform a scan over the number of features N_f, the number of hidden units N_h, and the number of spins per network parameter N_b, to record the number of abstract spins required by the encoding in Sec. III and the corresponding number of qubits of the embedded model. The results are shown in Fig. 3. Networks of this kind with up to ∼180 spins can thus be embedded in the currently available annealers.

In practice, we find that the annealer performs best when the number of spins is well below its maximum capacity. Therefore, in this study we pick a network with $N_f = N_h = 2$ and $N_b = 1$ for testing the performance of the training. The expressiveness of this network is limited, of course, but it is already enough to accurately classify samples in the following datasets (shown in Fig. 4).

(1) **Circles.** Points in a noisy circle around the origin labeled $y = 1$, together with points in a blob inside the circle labeled $y = -1$.

(2) **Quadrants.** Uniformly distributed points in a around the origin, labeled $y = -1$ if in the first or third quadrand, and $y = 1$ otherwise.

(3) **Bands.** Three partially overlapping bands, parallel to the $x_2 = x_1$ direction, with labels $y = 1$, $y = -1$, and $y = 1$, respectively.

(4) **ttbar.** A set of events generated by simulations of proton-proton collisions at the large hadron collider (LHC) with final state containing two top quarks (see Ref. [13]). The label $y = 1$ (the signal) indicates that the two tops are the decay products of a hypothetical new particle Z' [37]. The label $y = -1$ (the background) means they arise from the known Standard Model of physics. The features x_0 and x_1 correspond to the highest transverse momentum of a b jet and the missing energy, respectively. Separating the signal from the background is relevant for experimental searches in this context [38–40].

These datasets are best fitted with a smaller last-layer bias than ± 1. We thus modify Eq. (7) for v_0 so that it takes the values $-1/2$ or 0 instead. We use the D-Wave ADVANTAGE_SYSTEM4.1 annealer to train the network with an annealing schedule given by

$$s(t) = \begin{cases} s_q \frac{t}{20} & \text{if } 0 \leqslant t < 20, \\ s_q & \text{if } 20 \leqslant t < 80, \\ s_q + (1 - s_q)\frac{t-80}{20} & \text{if } 80 \leqslant t < 100, \end{cases} \tag{14}$$

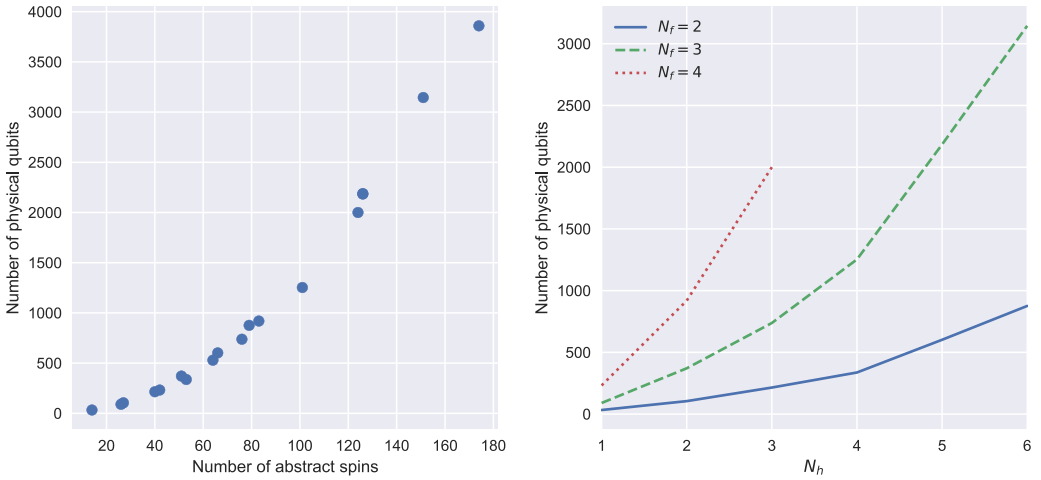

FIG. 3. Left: Number of physical qubits needed to embed the quantum NN as a function of the number of spins in its abstract encoding, for varying values of the number of features $N_f = 2$–4, the number of units in the hidden layer $N_h = 2$–4 and the number of spins per parameter $N_b = 1$–3. The graph shows that the number of physical qubits only depends on these parameters through the number of abstract spins, which is a function of them. Right: Number of qubits as function of N_h, for each value of N_f and fixed $N_b = 1$.

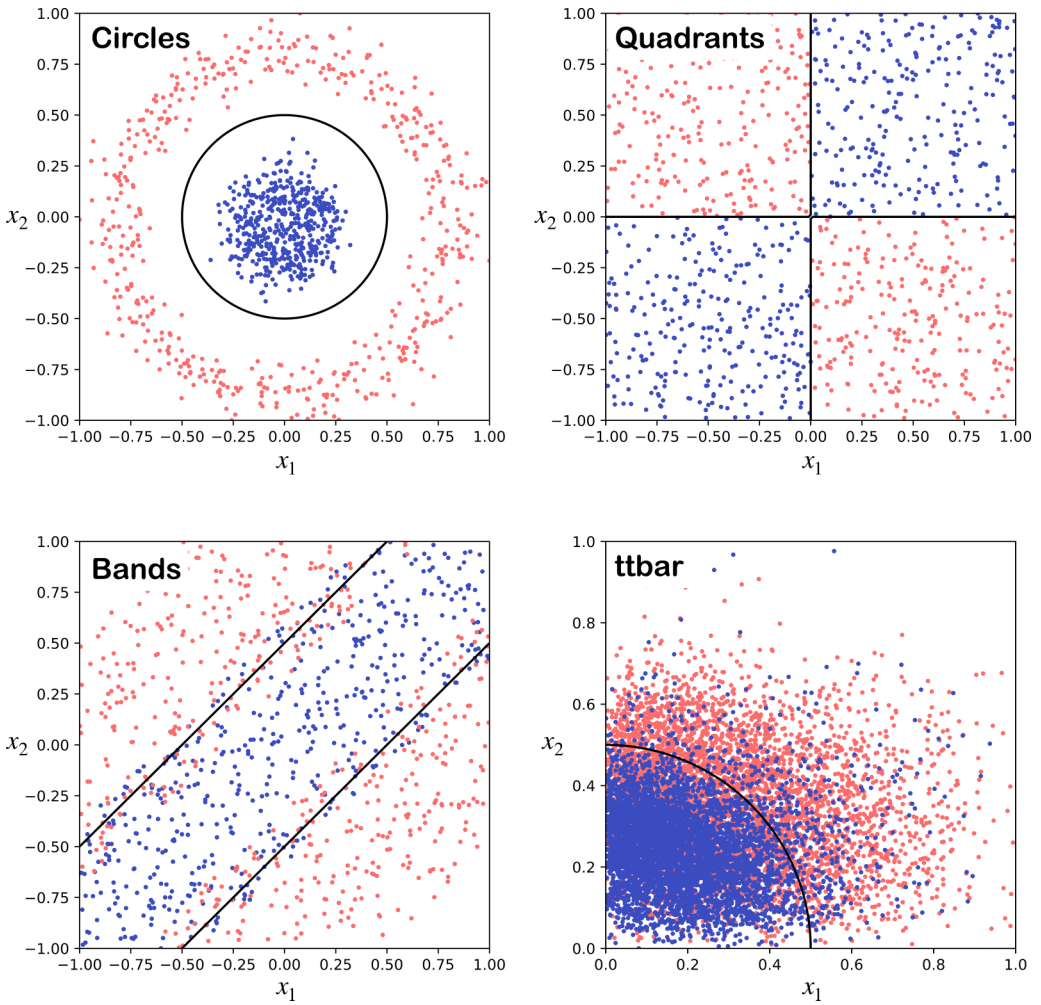

FIG. 4. Decision boundary obtained with the quantum NN for each dataset, together with the points in the dataset. Points labeled as signal and background are shown in blue and red, respectively. The decision boundary, displayed as a black line, is the set of points for which the network prediction is $Y_{v,w}(x) = 0$.

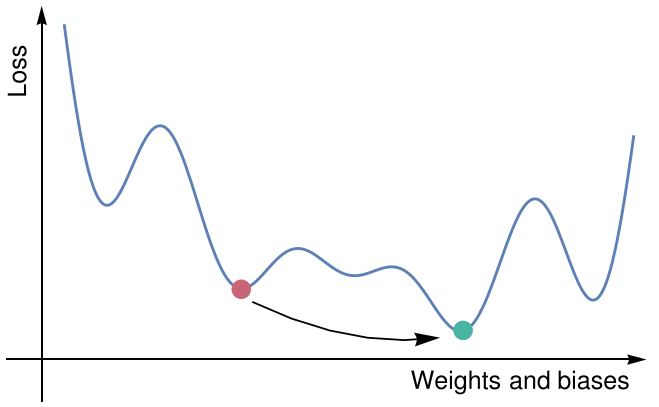

FIG. 5. Schematic representation of the loss function. Depending on the starting point and training method, classical training may end up in any local minima, such as the red one. Quantum training can tunnel from any of these minima to the global one, shown in green.

where t is given in microseconds and $s_q = 0.2$. Since the more expensive part of the computation is the preparation of the annealer for the desired model, it is customary to run it several times once this is done to reduce noise in the output. We perform 300 runs and pick the final state with the least energy. In Fig. 4 we show the decision boundary $Y(x) = 0$ obtained for each dataset using this procedure.

To measure the network's performance, we use the area under the receiver operating characteristic (ROC) curve. To generate the ROC curve, a threshold parameter is introduced. Each output of the NN is then interpreted as a prediction for $y = 1$ (signal) if it is above the threshold and for $y = -1$ (background) if it is below. The ROC curve is then the one described in the space of the true positive rate versus false positive rate, as one varies the threshold. The value of the area under this curve is perfect (100% area under ROC) for the `circles` and `quadrant` datasets, 92% for the `bands`, and 78% (close to the best attainable by other methods [13]) for `ttbar`.

The advantage of quantum training is twofold. First, it can be performed in a single step instead of the incremental gradient-descent procedure used customarily in classical training. Future annealers will be able to represent more extensive networks, which usually require large training times. The quantum annealing procedure will be able to reduce them significantly. Second, the quantum evolution can tunnel through barriers in the loss function to escape local minima, as depicted schematically in Fig. 5. By contrast, classical algorithms must somehow surmount the barriers to find the global minimum. Depending on the size of the barriers and on the hyperparameters of the training, classical methods may quickly become trapped in a local minimum and thus be unable to find the best solution to the problem. This advantage of quantum optimization was studied in Ref. [41], where various nonconvex loss functions were optimized with different methods, including quantum annealing and several classical algorithms. The quantum annealer found the global minimum more consistently than all the other methods.

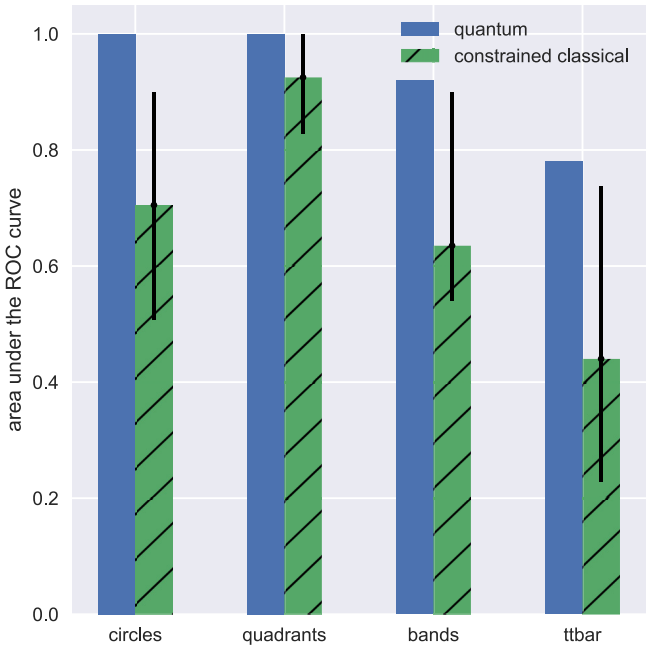

FIG. 6. Comparison of the results of the quantum NN to its classical analog. Each bar gives the median of ten training runs. The error bars show the percentiles 20 and 80.

To compare the effectiveness in finding the global minimum with classical algorithms, we trained a network with the same structure and activation function, using the ADAM version of the (classical) gradient descent algorithm. When the network parameters are continuous, we obtained results with similar scores for the area under the ROC curve as with the quantum training. However, such a network has much greater freedom than the one implemented in the annealer in which the parameters are only allowed to take two different values. To make the comparison more equitable, we mimicked this by adding

$$\omega \sum_p (p-1)^2 (p+1)^2, \tag{15}$$

to the classical loss function, where the sum is over all parameters (weights and biases) p. When ω is sufficiently large, the weights and biases are forced to adopt the same $p = \pm 1$ values available to the quantum annealer.

We find that classical training constrained in this way fails to reach the optimal solution most of the time with any of the four datasets, a practical manifestation of the situation sketched in Fig. 5. A comparison between the results of the quantum and constrained classical training is shown in Fig. 6. Apart from having a lower score on average, the classical training exhibits a significant variance in the results due to its becoming trapped in different local minima.

A finer discretization of the input data and network weights (that is, an encoding of each variable that uses more spins) will be made possible by future quantum annealers as they are expected to have more qubits with more couplings between them. Moreover, the above comparison suggests that quantum annealers can operate and train NNs, even with coarsely

discretized weights and biases, where classical training would fail. This indicates that quantum training may quite rapidly be able to achieve better accuracy and do so more consistently than classical algorithms. In this regard it is also worth mentioning that there are techniques open to quantum annealers such as diabatic annealing that have the potential to avoid anneal times increasing exponentially with the difficulty of the problem [42,43], an issue that is seen in both adiabatic quantum annealing and in simulated annealing. Such techniques cannot be efficiently simulated classically. A related issue is the possible effects of dissipation. This was studied in, for example, Refs. [42,44], which provided evidence for robustness. Although an in-depth discussion of this issue is beyond the scope of the paper, it is worth mentioning that its effects can, in some circumstances, even be beneficial [45].

We note that the global minimum of the loss function is found more reliably using a quantum approach does not mean that it will be more likely to overfit. Overfitting will become important once larger NNs can be embedded in quantum annealers. The measures to prevent it should be taken at the level of the NN and loss function design. They should be similar to those of classical approaches, with a parameter space and global minimum that avoid the possibility of learning features specific to the training data as much as possible. The usual training, test, or validation split approach should be applied for this purpose. A typical example of an overfitting prevention measure is introducing a regularization term in the loss function that is proportional to the MSE of the weights and biases. This can be trivially implemented in our approach as it becomes a sum of quadratic terms in the spins after the binary encoding has been applied.

V. CONCLUSION

We presented a method for training NNs purely in a quantum annealing device. The result of the quantum training was the optimal set of values for the NN weights, which can then be used for inference with a classical NN. Our method involved encoding the free parameters of the network in terms of binary variables, approximating activation functions through polynomials, and reducing general binary polynomials to quadratic ones. All these techniques are applicable in a general setting, allowing for encoding any NN structure with any activation function and any loss function to be encoded as an Ising-model Hamiltonian.

The current quantum annealing technology only provides a limited number of qubits with sparse connections. This means that only relatively small NNs can be embedded in quantum annealers today. However, we were able to implement our method for the training of such a NN and shown that it learns to classify several datasets in one annealing run.

The advantage of quantum optimization algorithms is that quantum tunneling processes can escape local minima. In the example we consider, we showed that the quantum training with our method consistently performs better than its classical analog. This suggests that, when larger quantum annealing devices are available in future applications, the quantum training algorithm we present may be used to train NNs more robustly than classical algorithms in real-life applications.

ACKNOWLEDGMENTS

We would like to thank Luca Nutricati for helpful discussions. S.A., J.C. and M.S. are supported by the Science and Technology Facilities Council under grant ST/P001246/1.

APPENDIX: ENCODING BINARY HIGHER-DEGREE POLYNOMIALS AS QUADRATIC ONES

We prove here that, for any polynomial in binary spin variables $\sigma_\ell = \pm 1$, there exists a quadratic polynomial in an extended set of variables that includes auxiliary spins, such that the values of σ_ℓ at the global minima in the new system coincide with those in the old system.

The proof is by induction. Consider a polynomial of degree n, in a system with m spin variables $\sigma_{\ell=1\ldots m}$. This can have many terms, but it is sufficient to focus on the terms of degree n. In general, we can write these terms in the polynomial as follows:

$$P_n(\lambda_\kappa) \equiv \sum_{W_\kappa^n} \lambda_\kappa W_\kappa^n, \tag{A1}$$

where $W_\kappa^n \equiv \sigma_{\ell_1}\sigma_{\ell_2}\ldots\sigma_{\ell_n}$ are all the words (that is monomials in which each spin can appear only once) of length n that can be made from m spins. Since $m \geqslant n$ there are ${}_mC_n$ of them labeled by κ with couplings λ_κ (some of which may be zero).

Consider a single pair of spins, say σ_1 and σ_2. Words containing the product $\sigma_1\sigma_2$ can be reduced by first converting to binary variables with $\sigma_\ell = 2\tau_\ell - 1$, where $\tau_\ell = 0, 1$. Thus all such pairs translate as

$$\sigma_1\sigma_2 \equiv 4\tau_1\tau_1 - 2\tau_1 - 2\tau_2 + 1.$$

The linear and constant terms are contributions to the degree $n-1$ polynomial already and the reduction task is therefore equivalent to reducing the $\tau_1\tau_2$ pair of binary variables.

Consider adding to the polynomial a quadratic term involving the binary variables together with a new auxiliary variable τ_{12}, which is of the form

$$Q(\tau_{12};\tau_1,\tau_2) \;=\; \Lambda[\tau_1\tau_2 - 2\tau_{12}(\tau_1+\tau_2) + 3\tau_{12}], \tag{A2}$$

where the overall coupling Λ is chosen to be sufficiently large and positive. This Hamiltonian can, of course, be translated back into σ_ℓ, however it is easier in QUBO format to check that it forces $\tau_{12} = \tau_1\tau_2$. Indeed if $\tau_1 = \tau_2 = 0$ then $Q = 3\Lambda\tau_{12}$ and if only one of τ_1 or τ_2 is zero then $Q = \Lambda\tau_{12}$. In both cases the minimum is at $\tau_{12} = 0$ and $Q = 0$. Meanwhile if $\tau_1 = \tau_2 = 1$ then $Q = \Lambda(1-\tau_{12})$ and the minimum is at $\tau_{12} = 1$ and $Q = 0$.

Thus adding $Q(\tau_{12};\tau_1,\tau_2)$ allows us to reduce all the words in P_n that contain the product $\tau_1\tau_2$ by replacing the pair of binary variables with the single variable τ_{12}. We note that the combined polynomial

$$\widehat{P}(\tau_{12},\sigma_1,\sigma_2\ldots) \;=\; P_{\widehat{12}} + Q(\tau_{12};\tau_1,\tau_2), \tag{A3}$$

in which $P_{\widehat{12}}$ is the original polynomial with the replacement $\tau_1\tau_2 \to \tau_{12}$ in the W_κ^n words, preserves the global minimum in $\sigma_1,\ldots,\sigma_m$.

We then choose further pairs until the degrees of all the W_κ^n words are reduced, resulting in a degree $n-1$ polynomial with the same global minimum as the original. Note that to

reduce all the words, the same spin may need to appear in more than one pair. This does not disrupt the proof, however, because the minima generated by Q are degenerate and do not favor any value of τ_1 or τ_2 but simply fix τ_{12} to their product, which can then be replaced throughout. Having established that the degree of the polynomial can be reduced by one unit while preserving the global minimum, as required, the final step in the induction is to reduce the words of length $n = 3$ to $n = 2$, which follows in the same manner as for general n.

Finally, let us verify that the reduction works correctly with a simple specific example, namely that provided by the Hamiltonian

$$\begin{aligned} H &= \sigma_1\sigma_2\sigma_3 \\ &\equiv 8\tau_1\tau_2\tau_3 - 4\tau_1\tau_2 - 4\tau_1\tau_3 - 4\tau_2\tau_3 + 2\tau_1 + 2\tau_2 + 2\tau_3, \end{aligned} \tag{A4}$$

where we drop the constant -1 in translating to the binary variables. This example is interesting because the degeneracy of the global minima is lower than in the generic case: there are only four solutions to $\sigma_1\sigma_2\sigma_3 = -1$ (which corresponds in binary language to any one of the τ_ℓ being zero, or all of them), as opposed to the seven solutions to $\tau_1\tau_2\tau_3 = 0$. One might therefore be concerned that new degenerate minima may appear, but this is not the case. As usual we now reduce the trilinear term by trading $\tau_1\tau_2$ for an auxilliary binary τ_{12} by adding the Hamiltonian $Q(\tau_{12}; \tau_1, \tau_2)$, and replacing the pair in H: that is, the new QUBO Hamiltonian becomes

$$\begin{aligned} H &= Q(\tau_{12}; \tau_1, \tau_2) + 8\tau_{12}\tau_3 - 4\tau_{12} - 4\tau_1\tau_3 - 4\tau_2\tau_3 \\ &\quad + 2\tau_1 + 2\tau_2 + 2\tau_3. \end{aligned} \tag{A5}$$

It is easy to check that provided $\Lambda > 2$ the original four degenerate solutions hold in the new combined Hamiltonian as required.

[1] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, The theory of variational hybrid quantum-classical algorithms, New J. Phys. **18**, 023023 (2016).

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

[3] N. Killoran, T. R. Bromley, J. M. Arrazola, M. Schuld, N. Quesada, and S. Lloyd, Continuous-variable quantum neural networks, Phys. Rev. Research **1**, 033063 (2019).

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

[5] A. Blance and M. Spannowsky, Unsupervised event classification with graphs on classical and photonic quantum computers, J. High Energy Phys. 08 (2021) 170.

[6] A. Mari, T. R. Bromley, J. Izaac, M. Schuld, and N. Killoran, Transfer learning in hybrid classical-quantum neural networks, Quantum **4**, 340 (2020).

[7] V. S. Ngairangbam, M. Spannowsky, and M. Takeuchi, Anomaly detection in high-energy physics using a quantum autoencoder, Phys. Rev. D **105**, 095004 (2022).

[8] K. Terashi, M. Kaneda, T. Kishimoto, M. Saito, R. Sawada, and J. Tanaka, Event classification with quantum machine learning in high-energy physics, Computing and Software for Big Science **5**, 2 (2021).

[9] R. Sweke, J.-P. Seifert, D. Hangleiter, and J. Eisert, On the quantum versus classical learnability of discrete distributions, Quantum **5**, 417 (2021).

[10] J. Y. Araz and M. Spannowsky, Classical versus Quantum: comparing Tensor Network-based Quantum Circuits on LHC data, arXiv:2202.10471.

[11] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, A variational eigenvalue solver on a photonic quantum processor, Nat. Commun. **5**, 4213 (2014).

[12] E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization algorithm, arXiv:1411.4028.

[13] A. Blance and M. Spannowsky, Quantum machine learning for particle physics using a variational quantum classifier, J. High Energy Phys. 02 (2021) 212.

[14] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, Quantum computation by adiabatic evolution, arXiv:quant-ph/0001106.

[15] H. Neven, V. S. Denchev, M. Drew-Brook, J. Zhang, W. G. Macready, and G. Rose, Nips 2009 demonstration: Binary classification using hardware implementation of quantum annealing, Quantum 4 (2009), https://static.googleusercontent.com/media/www.google.com/en//googleblogs/pdfs/nips_demoreport_120709_research.pdf.

[16] B. O'Gorman, R. Babbush, A. Perdomo-Ortiz, A. Aspuru-Guzik, and V. Smelyanskiy, Bayesian network structure learning using quantum annealing, Eur. Phys. J.: Spec. Top. **224**, 163 (2015).

[17] A. Mott, J. Job, J. R. Vlimant, D. Lidar, and M. Spiropulu, Solving a Higgs optimization problem with quantum annealing for machine learning, Nature (London) **550**, 375 (2017).

[18] A. Zlokapa, A. Mott, J. Job, J.-R. Vlimant, D. Lidar, and M. Spiropulu, Quantum adiabatic machine learning by zooming into a region of the energy surface, Phys. Rev. A **102**, 062405 (2020).

[19] M. Sasdelli and T.-J. Chin, *Quantum annealing formulation for binary neural networks*, in *2021 Digital Image Computing: Techniques and Applications (DICTA)* (IEEE, New York, 2021), pp. 1–10.

[20] A. B. Finnila, M. A. Gomez, C. Sebenik, and J. D. Doll, Quantum annealing: A new method for minimizing multidimensional functions, Chem. Phys. Lett. **219**, 343 (1994).

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

[22] J. Brooke, D. Bitko, T. F. Rosenbaum, and G. Aeppli, Quantum annealing of a disordered magnet, Science **284**, 779 (1999).

[23] N. G. Dickson *et al.*, Thermally assisted quantum annealing of a 16-qubit problem, Nat. Commun. **4**, 1903 (2013).

[24] T. Lanting, A. J. Przybysz, A. Y. Smirnov, F. M. Spedalieri, M. H. Amin, A. J. Berkley, R. Harris, F. Altomare, S. Boixo, P. Bunyk, N. Dickson, C. Enderud, J. P. Hilton, E. Hoskinson, M. W. Johnson, E. Ladizinsky, N. Ladizinsky, R. Neufeld, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, S. Uchaikin, A. B. Wilson, and G. Rose, Entanglement in

a Quantum Annealing Processor, Phys. Rev. X **4**, 021041 (2014).

[25] T. Albash, W. Vinci, A. Mishra, P. A. Warburton, and D. A. Lidar, Consistency tests of classical and quantum models for a quantum annealer, Phys. Rev. A **91**, 042314 (2015).

[26] T. Albash and D. A. Lidar, Adiabatic quantum computing, Rev. Mod. Phys. **90**, 015002 (2018).

[27] S. Boixo, V. N. Smelyanskiy, A. Shabani, S. V. Isakov, M. Dykman, V. S. Denchev, M. H. Amin, A. Y. Smirnov, M. Mohseni, and H. Neven, Computational multiqubit tunnelling in programmable quantum annealers, Nat. Commun. **7**, 10327 (2016).

[28] N. Chancellor, S. Szoke, W. Vinci, G. Aeppli, and P. A. Warburton, Maximum–entropy inference with a programmable annealer, Sci. Rep. **6**, 22318 (2016).

[29] 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, Phys. Rev. A **94**, 022308 (2016).

[30] S. Muthukrishnan, T. Albash, and D. A. Lidar, Tunneling and Speedup in Quantum Optimization for Permutation-Symmetric Problems, Phys. Rev. X **6**, 031010 (2016).

[31] A. Cervera-Lierta, Exact ising model simulation on a quantum computer, Quantum **2**, 114 (2018).

[32] T. Lanting, The d-wave 2000q processor, presented at AQC 2017 (2017).

[33] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Syst. **2**, 303 (1989).

[34] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Networks **4**, 251 (1991).

[35] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, The expressive power of neural networks: A view from the width, Advances in neural information processing systems **30** (2017).

[36] P. Kidger and T. Lyons, Universal approximation with deep narrow networks, in *Conference on learning theory*, in Proceedings of Thirty Third Conference on Learning Theory, PMLR, Vol. 125, (PMLR, 2020), pp. 2306–2327.

[37] G. Altarelli, B. Mele, and M. Ruiz-Altaba, Searching for new heavy vector bosons in $p\bar{p}$ colliders, Z. Phys. C **45**, 109 (1989); **47**, 676 (1990).

[38] S. Chatrchyan *et al.* (CMS Collaboration), Search for anomalous $t\bar{t}$ production in the highly-boosted all-hadronic final state, J. High Energy Phys. 09 (2012) 029.

[39] M. Aaboud *et al.* (ATLAS Collaboration), Search for heavy particles decaying into top-quark pairs using lepton-plus-jets events in proton–proton collisions at $\sqrt{s} = 13$TeV with the ATLAS detector, Eur. Phys. J. C **78**, 565 (2018).

[40] M. Aaboud *et al.* (ATLAS Collaboration), Search for heavy particles decaying into a top-quark pair in the fully hadronic final state in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector, Phys. Rev. D **99**, 092004 (2019).

[41] S. Abel, A. Blance, and M. Spannowsky, Quantum Optimisation of Complex Systems with a Quantum Annealer, arXiv:2105.13945.

[42] H. Goto and T. Kanao, Quantum annealing using vacuum states as effective excited states of driven systems, Commun. Phys. **3**, 235 (2020).

[43] E. Crosson and D. Lidar, Prospects for quantum enhancement with diabatic quantum annealing, Nat. Rev. Phys. **3**, 466 (2021).

[44] M. Keck, S. Montangero, G. E. Santoro, R. Fazio, and D. Rossini, Dissipation in adiabatic quantum computers: lessons from an exactly solvable model, New J. Phys. **19**, 113029 (2017).

[45] G. Passarelli, G. D. Filippis, V. Cataudella, and P. Lucignano, Dissipative environment may improve the quantum annealing performances of the ferromagnetic p-spin model, Phys. Rev. A **97**, 022319 (2018).