Detecting quantum phase transitions in a frustrated spin chain via transfer learning of a quantum classifier algorithm

André J. Ferreira-Martins [1], Leandro Silva,[1,2] Alberto Palhares [1,3] Rodrigo Pereira [1,3] Diogo O. Soares-Pinto,[2]
Rafael Chaves,[1,4] and Askery Canabarro [2,5,*]

[1]*International Institute of Physics, Federal University of Rio Grande do Norte, 59078-970 Natal, Brazil*
[2]*Instituto de Física de São Carlos, Universidade de São Paulo, CP 369, 13560-970 São Carlos, São Paulo, Brazil*
[3]*Departamento de Física Teórica e Experimental, Federal University of Rio Grande do Norte, 59078-970 Natal, Brazil*
[4]*School of Science and Technology, Federal University of Rio Grande do Norte, 59078-970 Natal, Brazil*
[5]*Grupo de Física da Matéria Condensada, Núcleo de Ciências Exatas - NCEx, Campus Arapiraca, Universidade Federal de Alagoas,
57309-005 Arapiraca, Alagoas, Brazil*

The classification of phases and the detection of phase transitions are central and challenging tasks in diverse fields. Within physics, these rely on the identification of order parameters and the analysis of singularities in the free energy and its derivatives. Here, we propose an alternative framework to identify quantum phase transitions. Using the axial next-nearest-neighbor Ising (ANNNI) model as a benchmark, we show how machine learning can detect three phases (ferromagnetic, paramagnetic, and a cluster of the antiphase with the floating phase). Employing supervised learning, we demonstrate the feasibility of transfer learning. Specifically, a machine trained only with nearest-neighbor interactions can learn to identify a new type of phase occurring when next-nearest-neighbor interactions are introduced. We also compare the performance of common classical machine learning methods with a version of the quantum nearest neighbors (QNN) algorithm.

I. INTRODUCTION

Machine learning (ML) has proven its efficiency and success in many scientific as well as business sectors [1–19]. In essence, we can teach computers to see patterns by progressively exposing them to quality inputs, which is crucial for data-driven solutions given the gigantic and ever-increasing amount of raw data. Within any branch of ML, substantial improvements in state-of-the-art solutions are strongly related to algorithmic and hardware advances. And, although we still need to be careful about setting long-term expectations, recent breakthroughs in the current noisy intermediate-scale quantum (NISQ) era [20–23] put quantum computing among the most promising directions toward significant progress in machine learning. Within this context, there have been a number of different approaches seeking quantum advantages in machine learning ranging from the quantum analog of neural networks [24] to routines for supervised or unsupervised learning [25,26], quantum reinforcement learning [27], and quantum pattern recognition [28] (see Refs. [29–31] for a detailed review).

A field where machine learning has been particularly successful is that of quantum matter and quantum information. Classical machine learning techniques were used to perform quantum state tomography [12], to approximate the ground state of many Hamiltonians of interest [11], for the description of causal networks [32–34], and finding violations of Bell

inequalities [17,35], among many other applications. Importantly, such classical methods have also been proven capable of tackling a central topic in many-body physics, that of classifying phase transitions, a thorny goal especially due to the exponential increase of Hilbert space describing quantum systems. Apart from simple transitions, witnessed by nonanalyticities in order parameters, more general quantum phase transitions require large lattice sizes, a costly computational task for which a variety of classical machine learning techniques provide alternative and reliable approaches [14–16,36]. It seems thus natural to consider whether quantum machine learning can also identify phase transitions. Indeed, machine learning based on hybrid quantum-classical variational circuits has been shown to detect phase transitions in the simple Hamiltonians, such as the transverse field Ising and *XXZ* models [37–40]. Our approach distinguishes itself significantly from others, primarily through the implementation of transfer learning using a quantum classifier algorithm. This algorithm is exclusively trained within a specific segment of the phase diagram while testing on the rest of the phase diagram. We also explore optimized data preprocessing for compatibility with real quantum hardware. This demonstrates the effectiveness of our technique, as discussed in detail herein.

Our aim in this paper is to show that the quantum nearest neighbors (QNN) algorithm [41] also provides a well-founded tool for classifying quantum phase transitions. Moving beyond the models previously considered, we benchmark the QNN performance by employing the axial next-nearestneighbor Ising (ANNNI) model [42,43] used, for instance, to investigate the magnetic order in quasi-one-dimensional

*askery@gmail.com

spin ladder materials [44], quantum melting of crystalline order in Rydberg atom systems [45], interactions between Majorana edge modes in arrays of Kitaev chains [46,47], and quench dynamics and dynamical phase transitions [48–50]. The ANNNI is the simplest model combining the effects of quantum fluctuations and frustrated exchange interactions, a combination from which a rich ground state phase diagram arises [39,51–55]. It thus provides an interesting challenge to the QNN algorithm capabilities.

As we show, even after significantly reducing the input data, to make it compatible with quantum computational requirements, the QNN algorithm still allows for a successful learning of phase transitions. More precisely, we demonstrate the transfer learning in the ANNNI model, as by training the machine with nearest-neighbor interactions only, it also accurately predicts the phase transitions happening at regions including next-nearest-neighbor interactions. Interestingly, the QNN performs better than its classical counterpart, called K nearest neighbors (KNN), when exposed to the same input data, thus providing a proof-of-principle example of a possible quantum advantage in accuracy.

The paper is organized as follows. In Sec. II we describe the ANNNI model. In Sec. III we provide a succinct but comprehensive overview of classification problems in machine learning, also describing the KNN and QNN algorithms. In Sec. IV we detail the data preprocessing required to make the problem amenable to be implemented in near-term quantum devices. In Sec. V we present our results regarding the learning of phase transitions in the ANNNI model. In Sec. VI we discuss our results and point out interesting directions for future research. Finally, in the Appendix we provide technical details about some of the classical machine learning techniques we have employed in the preprocessing of the data.

II. THE ANNNI MODEL

With the goal of analyzing the use of a quantum classifier algorithm to witness phase transitions in a quantum many-body system, we chose the axial next-nearest-neighbor Ising (ANNNI) model. The reason stems from the fact that this model displays a nontrivial and rich phase diagram. As it happens, ANNNI is the simplest model combining quantum fluctuations and competing frustrated exchange interactions. The first is induced by the presence of a transverse field while the latter is due to the fact that even though the interaction is ferromagnetic for nearest neighbors, it becomes antiferromagnetic for next-nearest neighbors.

The Hamiltonian for the ANNNI model is given by [42,43]

$$H = -J \sum_{j=1}^{N} \left(\sigma_j^z \sigma_{j+1}^z - \kappa \sigma_j^z \sigma_{j+2}^z + g \sigma_j^x \right), \quad (1)$$

where σ_j^α ($\alpha = x, y, z$) are Pauli matrices acting on spin-1/2 degrees of freedom at site j of a one-dimensional lattice with N sites and periodic boundary conditions. The parameter $J > 0$ is a coupling constant that sets the energy scale of the problem (we set $J = 1$) and is associated with the nearest-neighbor ferromagnetic exchange interaction. The dimensionless coupling constants κ and g are related to the

next-nearest-neighbor interaction and the transverse magnetic field, respectively.

The ground state phase diagram of the ANNNI model is well understood and known to contain four phases separated by three quantum phase transitions: ferromagnetic, antiphase, paramagnetic, and floating phase. In a nutshell, the ferromagnetic phase is characterized by a uniform spontaneous magnetization, with one of the ground states given by ↑↑↑↑↑↑↑↑. In turn, the antiphase breaks the lattice translational symmetry and has long-range order with a four-site periodicity of the form ↑↑↓↓↑↑↓↓. Distinctively, the paramagnetic phase is disordered and has a unique ground state with spins pointing predominantly along the field direction. Finally, the floating phase is gapless with correlation functions decaying as a power law for large distances, in contrast with the other phases that have a finite energy gap and exponentially decaying correlations.

For $\kappa = 0$, the transverse field Ising model is reproduced, exactly solvable via the mapping to noninteracting spinless fermions. Along the $\kappa = 0$ line, a second-order phase transition occurs at $g = 1$, separating the ferromagnetic phase at $g < 1$ from the paramagnetic phase at $g > 1$. In particular, exactly at the critical point $g = 1$, the energy gap vanishes.

For $g = 0$, there is a transition between the ferromagnetic phase at small κ and the antiphase at large κ occurring at $\kappa = 1/2$. Notice that with $g = 0$ the model becomes classical, since all operators in the Hamiltonian commute with each other. At this classical transition point, any configuration that does not have three neighboring spins pointing in the same direction is a ground state, showing that the degeneration of the ground state increases exponentially with the system size.

For $g \neq 0$ and $\kappa \neq 0$, the critical lines have to be determined numerically since the model is not integrable any longer. For $0 \leqslant \kappa \leqslant 1/2$, the Ising transition between the paramagnetic and ferromagnetic phases extends from the $g = 1$, $\kappa = 0$ until the degenerate point $g = 0$, $\kappa = 1/2$, a multicritical point at which several transition lines coincide. Two other transition lines start at the multicritical point and extend to the high-frustration regime $\kappa > 1/2$. For fixed $g > 0$ and increasing $\kappa > 1/2$, we first encounter a Berezinsky-Kosterlitz-Thouless (BKT) transition from the paramagnetic phase to the floating phase. Detecting the BKT transition is challenging because the correlation length diverges exponentially at the critical point. However, the level spectroscopy method offers a robust approach for identifying a BKT transition point and examining its universality class in one-dimensional quantum systems [56,57]. As we increase κ further, there is a commensurate-incommensurate (CIC) transition from the floating phase to the antiphase. Numerical density matrix renormalization group results for long spin chains [54] show that the floating phase occupies a rather narrow region in the phase diagram, which makes it hard to discern the BKT from the CIC transition for small system sizes.

Using perturbation theory in the regime $\kappa < 1/2$ [43] or by fitting numerical results [54] in the regime $\kappa > 1/2$, one can obtain approximate expressions for the transition lines. For instance, the critical value of g for the Ising transition for
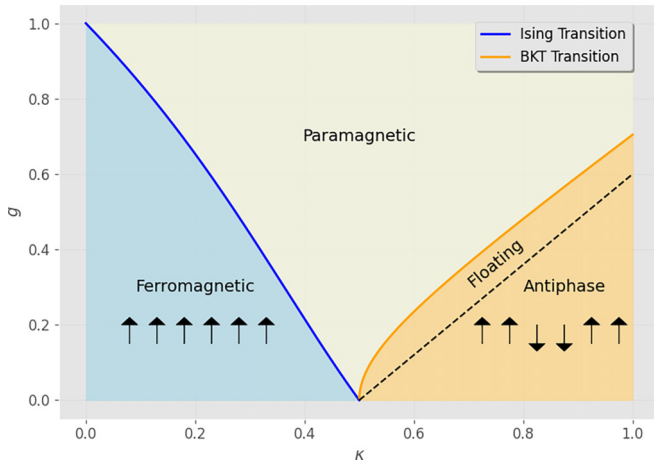
FIG. 1. Illustration of the main phases of ANNNI model. The phase diagram is plotted in terms of the dimensionless coupling constants g and κ.

$0 \leqslant \kappa \leqslant 1/2$ is approximately given by [43]

$$g_\mathrm{I}(\kappa) \approx \frac{1-\kappa}{\kappa}\left(1 - \sqrt{\frac{1-3\kappa+4\kappa^2}{1-\kappa}}\right). \qquad (2)$$

In turn, the critical value of g for the BKT transitions for $1/2 < \kappa \lesssim 3/2$ is approximated by [54]

$$g_\mathrm{BKT}(\kappa) \approx 1.05\sqrt{(\kappa-0.5)(\kappa-0.1)}. \qquad (3)$$

We use these approximations to make benchmark comparisons to our heuristic results.

In Fig. 1, we depict the main phases of the ANNNI model using the approximations given by Eqs. (2) and (3). We have also schematically represented the floating phase, investigated in detail in [54]. It is important to note that this phase is very narrow, making it challenging to observe, particularly in small chains like the ones analyzed in our study.

Our dataset

As will be discussed in more detail throughout the paper, we use the pairwise correlations among all spins in the lattice as the training data for the machine learning algorithms. Given N spins, we have a total of $3 \times C_2 = \binom{N}{2}$ observables for up to (second) nearest neighbors, hence the combination by (2). Thus, the features are given by $\{\langle\sigma_i^x\sigma_j^x\rangle, \langle\sigma_i^y\sigma_j^y\rangle, \langle\sigma_i^z\sigma_j^z\rangle\}$ with $j > i$ and $i = [1, N-1]$, where N is the number of spins in the lattice and $\langle\sigma_i^x\sigma_j^x\rangle = \langle\lambda_0|\sigma_i^x\sigma_j^x|\lambda_0\rangle$ is the expectation value of the spin correlation for the Hamiltonian ground state $|\lambda_0\rangle$ (and similarly for the other expectation values). In our case, we take $N = 12$, a manageable size for both computational and analytical evaluation of the ground state of the ANNNI Hamiltonian. This allows us to efficiently compute a set of 198 pairwise expectation values, which we refer to as "original features," since they could serve as the (raw) input variables for a machine learning algorithm.

It is worth pointing out that even if one only has access to short chains, the Ising transition can still be captured correctly [39]. However, detecting the BKT transitions using standard approaches requires computing observables for significantly

longer chains [54]. Notwithstanding, as we will see below, even though our data regard a quite short chain $N = 12$, the machine learning algorithms, both classical and quantum, will be able to identify not only the Ising but also the antiphase and the paramagnetic phases, lumping the BKT and CIC transitions together.

III. THE QUANTUM NEAREST NEIGHBORS ALGORITHM

The quantum nearest neighbors (QNN) [41] is a quantum classification algorithm that employs the Hamming distance as a distance criterion to compare the training dataset and unclassified observations. Schematically, it consists of three major steps:

(i) First, create a superposition of the training dataset and the input observation.

(ii) Encode the Hamming distance between the input observation and each example in the training set into the amplitude of each state in the superposition.

(iii) Measure the class qudit retrieving the predicted class with the highest probability.

Before the actual quantum algorithm starts, an important classical preprocessing step (whose reason will become clear in what follows) must be performed: the features in the training dataset are represented as bit vectors, so that the feature space becomes $\mathcal{X} = \{0, 1\}^{\otimes n}$. This is achieved via the procedure known as one-hot encoding, which produces the so-called dummy variables [58]. Naturally, such a representation will be discrete (binary, in fact), so that if any original feature is continuous (or even categorical with more than two levels), a prior step of discretization is necessary. Notice that the number of binary features after the encoding may be different from the number of original features, although here we represented both by the same number of bits n. There are several ways to perform this binarization process. However, whatever method is chosen, it is important that the essence of the data topology is maintained—that is, points that are close on the original feature space must remain close on the binarized feature space. In Sec. IV we detail the specifics of the particular procedure we applied to our problem.

Once the training dataset features are binarized, their representation as quantum states is immediate via the basis encoding [59], which accounts for a direct mapping of binary features to the quantum computational basis states: $0 \mapsto |0\rangle$ and $1 \mapsto |1\rangle$. After these two steps, each training set data point is mapped to the quantum state $|x_1^p \cdots x_n^p\rangle \equiv |\boldsymbol{x}^p\rangle$, $x_k^p \in \{0, 1\}$, $p = 1, \ldots, N$, where N is the number of points in the training set. In parallel, in a separate quantum register, we encode the class $y^p \in \{0, \ldots, d-1\}$, and construct, for each observation p, the state

$$\left|x_1^p \cdots x_n^p, y^p\right\rangle \equiv |\boldsymbol{x}^p, y^p\rangle. \qquad (4)$$

If we are dealing with binary classification (which is the case in this work), the respective class is also straightforwardly encoded in a single qubit, as $0 \mapsto |0\rangle$ and $1 \mapsto |1\rangle$. If we have a multiclass problem, qudits are necessary, or one could use more than one qubit to encode integers corresponding to the class (for instance, $|5\rangle = |101\rangle$). In this case, $\lceil \log_2 d \rceil$ qubits are necessary to encode d classes.

Once we have the state corresponding to each one of the training states $|\boldsymbol{x}^p, y^p\rangle$, we construct a training set superposition of all data points, given by

$$|T\rangle = \frac{1}{\sqrt{N}} \sum_{p=1}^{N} |\boldsymbol{x}^p, y^p\rangle. \tag{5}$$

Naturally, with n qubits, one can construct a superposition of 2^n states, representing all possible binarized feature vectors of n features. However, it is possible (and most likely) that in a given training dataset not all possible binary feature vectors will be present. Indeed, in the binarization process, it is likely that multiple observations that are different in the original input space are mapped to the same binary vector so that the transformed training dataset actually has a number of observations quite smaller than the original number of observations (although here we represent both as N). This leads to important details in the implementation of the algorithm in a practical problem, as will be detailed in Sec. IV. Further, notice that in the case in which $N < 2^n$, the superposition in Eq. (5) will have to be prepared with an arbitrary state preparation routine, which is known to be costly [60]. In this work, we perform simulations using Qiskit's [61] arbitrary state preparation routine, which implements the one introduced in [62], which is shown to be optimal for arbitrary superpositions. However, in our case we have a uniform superposition of a subset of the full computational basis superposition, for which there exist much more efficient state preparation routines, namely the one in [63], which introduces a state preparation method that requires an exponential number of CNOTs in the worst case, but was shown to reduce CNOTs significantly for practical benchmarks, while generating an exact representation of quantum states without using free qubits, or a modification of the one in [64], which has a gate complexity and circuit depth of $O(\log N)$, where N is the number of training points, and uses neither ancilla qubits nor any quantum gates with multiple controls. By using one of these procedures, it is possible to efficiently prepare $|T\rangle$ without assuming the existence of a qRAM, or using the inefficient arbitrary state preparation routines, which brings the application of QNN closer to near-term devices.

The next step is to perform the same classical binarization process with the unclassified input vector $\boldsymbol{x}_{\text{in}}$ (the one we wish to classify) and map it to the state $|x_{\text{in},1} \cdots x_{\text{in},n}\rangle \equiv |\boldsymbol{x}_{\text{in}}\rangle$, $x_{\text{in},k} \in \{0, 1\}$. Keep this as the first register of the quantum state. Finally, add an ancilla register $|0\rangle$ as the last register. Such a construction yields an initial state given by

$$|\psi_0\rangle = \frac{1}{\sqrt{N}} \sum_{p=1}^{N} |\boldsymbol{x}_{\text{in}}; \boldsymbol{x}^p, y^p; 0\rangle, \tag{6}$$

which is made up of three registers (or, in fact, blocks of registers): the first containing the input state $|\boldsymbol{x}_{\text{in}}\rangle$, which consists of n qubits, the second containing the superposition $|T\rangle$ (which is the tensor product of the feature vectors $|\boldsymbol{x}^p\rangle$ and the class vectors $|y^p\rangle$), thus consisting of $n + 1$ qubits, and given that we have a binary classification problem, the third contains an ancilla qubit initialized as $|0\rangle$. Therefore, the number of qubits necessary for the algorithm is precisely $2n + 2$.

Once the initial state is prepared, we put the ancilla into a superposition, by applying a Hadamard gate to the last register, i.e., $H = 1 \otimes 1 \otimes 1 \otimes H$, such that

$$|\psi_1\rangle = H |\psi_0\rangle = \frac{1}{\sqrt{N}} \sum_{p=1}^{N} |\boldsymbol{x}_{\text{in}}; \boldsymbol{x}^p, y^p\rangle \otimes \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle). \tag{7}$$

In the next step, we want the Hamming distance components d_k^i between each qubit of the first (input) and second (training) register to replace the qubits in the second register, such that

$$d_k^i = \begin{cases} 0, & \text{if } |x_k^p\rangle = |x_{\text{in},k}\rangle, \\ 1, & \text{else.} \end{cases} \tag{8}$$

This is achieved by simply applying a cNOT$(x_{\text{in},k}, x_k^p)$ gate, which overwrites the entry x_k^p in the second register with 0 if $x_k^p = x_{\text{in},k}$, otherwise with 1:

$$\begin{aligned} \text{cNOT} |00\rangle = |00\rangle, \quad & \text{cNOT} |01\rangle = |01\rangle, \\ \text{cNOT} |11\rangle = |10\rangle, \quad & \text{cNOT} |10\rangle = |11\rangle. \end{aligned} \tag{9}$$

Thus, after this step, the state is then

$$\begin{aligned} |\psi_2\rangle &= \bigotimes_{k=1}^{n} \text{cNOT}(x_k, v_k^p) |\psi_1\rangle \\ &= \frac{1}{\sqrt{N}} \sum_{p=1}^{N} |\boldsymbol{x}_{\text{in}}; \boldsymbol{d}^p, y^p\rangle \otimes \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle), \end{aligned} \tag{10}$$

where the Hamming distance components $|d_1^p \cdots d_n^p\rangle \equiv |\boldsymbol{d}^p\rangle$, $d_k^p \in \{0, 1\}$, $p = 1, \ldots, N$ are now in the second register.

In the third step, we apply the unitary operator

$$U = e^{-i \frac{\pi}{2n} \mathcal{O}}, \quad \mathcal{O} = 1 \otimes \sum_{k=1}^{n} \left(\frac{1 - \sigma_z}{2} \right)_{d_k} \otimes 1 \otimes \sigma_z. \tag{11}$$

This sums the Hamming distance components $\{d_k^p\}$ (thus yielding the actual Hamming distance) between $|\boldsymbol{x}^p\rangle$ and $|\boldsymbol{x}_{\text{in}}\rangle$, $d_H(\boldsymbol{x}_{\text{in}}, \boldsymbol{x}^p) \equiv d_H$, into the phase of the pth state of the superposition. Notice that a relative phase is added, conditioned on the ancilla state. After this step, the state becomes

$$\begin{aligned} |\psi_3\rangle = U |\psi_2\rangle = \frac{1}{\sqrt{2N}} \sum_{p=1}^{N} (e^{-i \frac{\pi}{2n} d_H} |\boldsymbol{x}_{\text{in}}; \boldsymbol{d}^p, y^p; 0\rangle \\ + e^{i \frac{\pi}{2n} d_H} |\boldsymbol{x}_{\text{in}}; \boldsymbol{d}^p, y^p; 1\rangle). \end{aligned} \tag{12}$$

Now we apply another Hadamard to the ancilla. This will generate alternating-sign exponentials associated with each ancilla state, which are easily aggregated into a sine and cosine. The resulting state can be expressed as

$$\begin{aligned} |\psi_4\rangle = H |\psi_3\rangle = \frac{1}{\sqrt{N}} \sum_{p=1}^{N} \Bigg[\cos\left(\frac{\pi d_H}{2n} \right) |\boldsymbol{x}_{\text{in}}; \boldsymbol{d}^p, y^p; 0\rangle \\ + \sin\left(\frac{\pi d_H}{2n} \right) |\boldsymbol{x}_{\text{in}}; \boldsymbol{d}^p, y^p; 1\rangle \Bigg]. \end{aligned} \tag{13}$$

Notice that $0 \leqslant d_H \leqslant n \Rightarrow 0 \leqslant \frac{\pi d_H}{2n} \leqslant \frac{\pi}{2}$. Therefore,

(i) for large d_H, $\cos(\frac{\pi d_H}{2n}) \to 0$ and $\sin(\frac{\pi d_H}{2n}) \to 1$, so that we have higher probability of measuring $|1\rangle$ in the ancilla qubit;

(ii) for small d_H, $\cos(\frac{\pi d_H}{2n}) \to 1$ and $\sin(\frac{\pi d_H}{2n}) \to 0$, so that we have higher probability of measuring $|0\rangle$.

That is, if the input is far away from most training observations, we have a higher probability of measuring the ancilla in the state $|1\rangle$, and if the input is close to many observations, the ancilla is more likely to be measured in $|0\rangle$. Thus, intuitively, since our criterion for classification is to consider the closest observations, by measuring the ancilla in $|0\rangle$, the amplitudes of close observations will be large, while the opposite is true for distant observations. The importance of this fact becomes clear if we rewrite $|\psi_4\rangle$, to show that the different classes appear weighted by their member's distance to the input, such that

$$|\psi_4\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{d-1} |y\rangle \otimes \sum_{l \in y} \left[\cos\left(\frac{\pi d_H}{2n}\right) |\boldsymbol{x}_{\text{in}}; \boldsymbol{d}^l; 0\rangle \right.$$
$$\left. + \sin\left(\frac{\pi d_H}{2n}\right) |\boldsymbol{x}_{\text{in}}; \boldsymbol{d}^l; 1\rangle \right], \quad (14)$$

where l runs over all training vectors classified with the label y. Written like this, it becomes clear that if the ancilla is measured to be in $|0\rangle$, the amplitudes of close observations will be large, which implies that the probability of measuring the respective class qubit of these observations will also be large. And, as we discuss next, this is how the final classification is performed.

As the final step, the ancilla of the state $|\psi_4\rangle$ is measured on the computational basis. According to Eq. (13), it is easy to see that the probability of measuring $|0\rangle$ is

$$P(|0\rangle_a) = |\langle 0|\psi_4\rangle|^2 = \frac{1}{N} \sum_{p=1}^{N} \cos^2\left(\frac{\pi d_H}{2n}\right). \quad (15)$$

The conditional probability to measure a certain class $y \in \{1, \ldots, d\}$, given that we previously measured the ancilla in $|0\rangle$ (and, therefore, the state collapsed to $|\tilde{\psi}_4\rangle = \langle 0|\psi_4\rangle |0\rangle$), is, in terms of the joint probability,

$$P(y \mid |0\rangle_a) = P(y)P(|0\rangle_a)$$
$$= |\langle y|\tilde{\psi}_4\rangle|^2$$
$$= \frac{1}{N} \sum_{l \in y} \cos^2\left(\frac{\pi d_H}{2n}\right), \quad (16)$$

which is easily verifiable using Eq. (14). Indeed, Eq. (16) implies that

$$P(y) = \frac{1}{P(|0\rangle_a)} \frac{1}{N} \sum_{l \in y} \cos^2\left(\frac{\pi d_H}{2n}\right). \quad (17)$$

Thus, the class measured with the highest probability is that whose members are the closest to the input vector, provided that $P(c)$ is only computed after the ancilla is measured in $|0\rangle$, which is precisely why the amplitudes associated to the closest neighbors are considered. Notice that if the measurement returns $|1\rangle$, this run of the algorithm is not taken into account.

In Fig. 7 the full quantum circuit is illustrated for a particular dataset, as detailed in Appendix A 4. The algorithm uses $O(Nn)$ [41] gates, which is completely due to the construction of the training data superposition [described by Eq. (5)], which depends on the number of training samples, thus yielding an $O(Nn)$ complexity [41,65], which is close to the classical KNN algorithm complexity, in which we have to compute the distance between the test observation and all other N training points. However, if one can find a procedure to prepare the training data superposition in a manner independent of the number of samples (perhaps by reading quantum data [66–69]), the QNN algorithms would run in $O(n)$, offering a potentially large advantage over the classical KNN, for which it seems unlikely that there exists an algorithm which is independent of the number of training samples; that is a quite remarkable advantage achieved by the exploitation of the superposition in a quantum algorithm.

We highlight that, in contrast to the classical KNN, the QNN algorithm does not depend on the hyperparameter k. In fact, a superposition of all members of each class is taken into consideration for the classification. This is equivalent to considering all neighbors (that is, $k = N$), which in the classical algorithm is associated with a high bias, since, if the dataset is imbalanced with respect to the target, the majority class will always be predicted. In the quantum algorithm, however, this is not the case: even if the dataset is imbalanced, the input observation will be assigned to the class which is the closest to it, since, as is clear in Eq. (17), the distance of the input to the members of the class explicitly affects the probability.

As a final remark, notice that the probability distribution in Eq. (17) is precisely what is recovered from multiple runs of the algorithm on an actual hardware (or a simulator thereof). The final prediction, as a class, is therefore recovered by choosing the class with the largest observed probability. However, as explained in Sec. IV and illustrated in Fig. 4, in this work we directly use the class probability, i.e., Eq. (17) itself. Fortunately, in contrast to many classical learning models, outputting class probabilities is the most natural choice for the QNN algorithm.

We remark that the Python/Qiskit implementation of the algorithm described above, as well as all the data used in this paper and the corresponding results, is available in an online repository [70].

IV. DATA PREPROCESSING

As described in Sec. III, the classical data loaded into the quantum registers, via the basis encoding strategy, must be in a binary representation. On the other hand, as discussed in Sec. II A, the dataset under study consists of 198 continuous features ("original features"): the pairwise correlations among all spins in a lattice with 12 sites considering boundary conditions and the symmetry it implies. Thus, in order to represent each observation as a binary vector, we must first discretize the continuous features, so that the discrete levels may then be binarized.

Before proceeding with these procedures, however, an important remark is due. As discussed above, the QNN algorithm uses $2n + 2$ qubits, where n is the number of binarized features. Indeed, this implies that if one wants to simulate the

circuit, or even execute it on NISQ hardware, n must be chosen accordingly, to make the execution/simulation feasible.

As will be detailed below, we employed an efficient discretization and binarization procedure that maps each original continuous feature to only two bits. However, given that we start with 198 features, this would imply $n = 396$, thus requiring a circuit consisting of 794 qubits, which is way beyond the limit for a classical simulation as well as current quantum hardware capabilities, both in terms of number of qubits as well as circuit depth. And this is a quite simple discretization one can think of: if one produces more bins per feature, which would be desirable, the resulting number of binary features (and qubits, consequently) would further increase, making the simulation or execution yet more intractable.

Therefore, in order to fit the dataset into current capabilities, we employ a series of preprocessing steps to the original raw features, which starts with a dimensionality reduction procedure implemented via a feature selection routine, in order to pick from the 198 original features, the ones that contribute the most to the classification, with the hope that they are enough to produce a good classifier. The procedure we use for picking the most important features is based on the random forest algorithm [71]—in particular, a modification thereof, known as "extremely randomized trees" [72]. It consists of the calculation of a "feature importance coefficient" which is also known as "mean decrease impurity" or "gini importance" [73]. This coefficient is calculated as the total decrease in each node impurity, averaged over all trees in the forest. The mean is weighted by the probability that the respective node is reached, which is often estimated by the proportion of observations reaching the node. A detailed account of this algorithm can be found in Appendix A 3.

Having in mind the discussion about the current capabilities of simulation and hardware, we have selected only the four most important features that correspond to the following two-body correlation terms. Physically, we expect that the most important features are the correlation functions $\langle \sigma_i^z \sigma_j^z \rangle$ at the largest available distances. The reason is that this correlation detects long-range order associated with the spontaneous breaking of the $\mathbb{Z}_2$ symmetry $\sigma_j^z \mapsto -\sigma_j^z$ of the Hamiltonian. In the paramagnetic phase, $\langle \sigma_i^z \sigma_j^z \rangle$ decays exponentially to zero for $|i - j|$ larger than the correlation length, while it approaches a nonzero value in the ferromagnetic phase and oscillates with a four-site periodicity in the antiphase. By contrast, the correlation function $\langle \sigma_i^x \sigma_j^x \rangle$ is nonzero for $g \neq 0$ in all phases because the transverse magnetic field induces a finite expectation value for σ_j^x.

We then proceed to the discretization of the features, using a procedure based on the k-means clustering algorithm [74]. More specifically, we use a k-bins discretizer, implemented in scikit-learn [75], which divides continuous data into k intervals or "bins." Essentially, we first cluster observations that are similar to the feature being discretized and use the clusters' centroids as centers of the bins; that is, values in each bin will have the same nearest centroid, as determined by the 1-dimensional k-means algorithm. See Fig. 2 for a schematic illustration of this procedure. For each feature, we created 3 bins. At this point, our dataset is characterized by 12 discrete values, 3 for each one of the 4 features selected by the feature importance procedure.
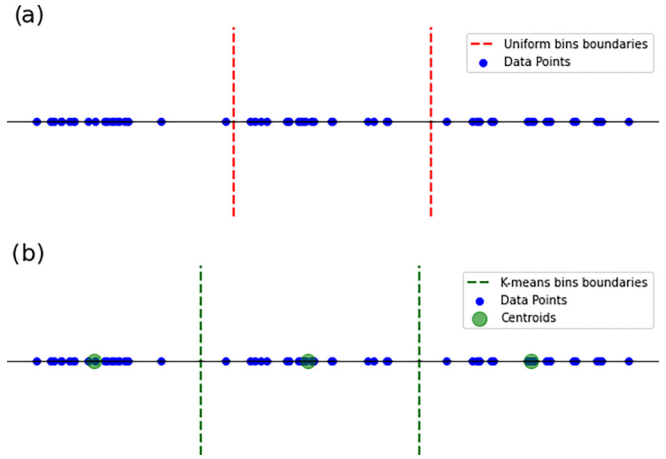
(a)

(b)

FIG. 2. Schematics of the k-bins discretizer using uniform and k-means ($k = 3$) binning strategies. (a) The k-bins discretizer with bins uniformly defined. The vertical red lines represent the bins' limits. Notice that bin widths are uniform in the feature space, but the clustering structure is not respected. (b) The green points represent the clusters' centroids, and the vertical green lines, the bins' limits. Notice how nonuniform bins are created, but the clustering structure is respected.

After discretization, the features are binarized using the one-hot encoding procedure, which consists of creating $l - 1$ independent binary features for each column with l categorical levels, as illustrated in Fig. 3. In our case, since $l = 3$ for each discretized feature, we create $l - 1 = 2$ new binary features each, which then results in $n = 8$ independent binary features. This is the final dimensionality we work with.

Notice that, with 8 binary features, we will need $2 \times 8 + 2 = 18$ qubits for the circuit execution, which is a reasonable number for the circuit simulation or execution—and, most importantly, it is enough to yield good results for the problem we analyze, as will be shown. We could have chosen more than 4 original features or discretized the features in more bins, which could possibly have increased the performance quantum classifier. In Sec. VI we further elaborate on this point.

FIG. 3. One-hot encoding procedure. From a column with $l = 3$ categorical (discrete) levels, we construct $l - 1 = 2$ independent binary columns, which are the binary features. Notice that only $l - 1$ binary columns are necessary because it is possible to represent one of the levels as "00" (in this example, "L3").

Notice that after the feature selection, discretization, and binarization preprocessing described above, some observations which were different in the original space of continuous features may be mapped to the same binary representation. This makes perfect sense, given that the different values of a given feature may fall in the same bin, which is given by a range of values. If this happens with all 4 features of two different observations, they will naturally be represented by the same 8-dimensional feature vector. This is why using a k-means binning strategy is a good idea [instead of the competing strategy "uniform," for example, in which all bins have identical widths, as depicted schematically in Fig. 2(b)]: given that bins are clusters, this strategy groups together similar observations in the same bin, so that it makes sense if they are represented by the same binary feature vector.

After the preprocessing routine, our original training dataset, which had 1000 rows and 198 continuous-valued columns, was reduced to a dataset with 8 binary features and only 10 rows. We can see that as a way of reducing the dataset only to the most representative samples and better explanatory features, which, as we will show below, was enough to yield good results with the quantum classification algorithm.

It is important to remark that the aforementioned feature importance and discretization processes were such that their respective parameters were determined only using the training data. That is, the exact same procedure was merely reproduced for all test datasets, but with parameters already defined in the training data. Now, although this is the correct thing to be done to avoid data leakage, there is a potential problem, especially with the discretization process: given that the feature range varies a lot from training to testing data, it is possible that the resulting bins for testing data features will be concentrated; that is, all observations will fall in the same bin of a given feature. Effectively, this implies that such test observations will be characterized by less than 8 features, which is a problem because the QNN algorithm assumes that the test (input) observation has the same number of features as all training observations. In order to fix this, we pad such input observations with zeros, to guarantee that all binarized testing observations will be 8-dimensional. In practice, different observations will be identified only by the actual features present, and the padding will have no effect in terms of the classification, given that it will be the same for all observations in a given test dataset, as we observed. Indeed, as the results show, such a procedure does not jeopardize the classifier's performance.

As already remarked, remember that QNN is a lazy algorithm, so each test (input) observation is classified at a time. This means that, in principle, we would have to simulate or execute a number of circuits equal to the number of test observations, to have their prediction. Given that we have 10 testing sets, one for each k value, and that we consider $k = 0$ the training point, each one with 1000 observations, the number of circuit simulations or executions would be quite large. However, the aforementioned fact that different training observations in the original feature space may be mapped to the same binary representation is of course also true for the testing data observations (although the exact number of unique binarized testing observations varies among the different test datasets). Given that, we implement a cache:
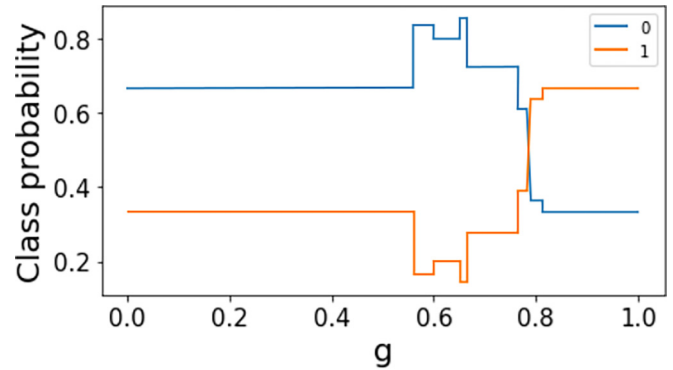

FIG. 4. QNN probability predictions for $\kappa = 0.1$, with training on $\kappa = 0$ as a function of the dimensionless transverse magnetic field coupling parameter g. The machine was trained at $\kappa = 0$ and asked to predict where the transition happens at $\kappa = 0.1$, by considering where the machine is most uncertain, that is, when the probabilities are closest $p_0 = p_1 = 1/2$. Here the ferromagnetic (paramagnetic) phase is labeled as 0 (1).

whenever we see a new testing observation (in terms of its 8 features), we pass it through the quantum circuit, simulating or executing it, and store its prediction. If this observation is repeated (which, again, can happen given the nature of the preprocessing routine), we do not run the circuit again, but instead merely replicate the prediction in the cache. This allows us to have a prediction for each one of the observations in the testing datasets, without having to simulate or execute the quantum algorithm that many times. Indeed, this is very important for resource optimization, in terms of simulation time or hardware calls for execution.

V. MACHINE LEARNING THE PHASE DIAGRAM OF THE ANNNI MODEL WITH QNN

Our aim is to understand whether transfer learning is possible using QNN. More specifically, all our training data consist of $\kappa = 0$ and we use that to predict phases at regions where $\kappa \geqslant 0$. This is particularly relevant since for $\kappa = 0$ the model is analytically solvable, pointing out that a transition occurs at $g \approx 1$. We highlight that for $\kappa = 0$ we have only two phases: either the ferromagnetic ("phase 0") or the paramagnetic ("phase 1"). However, when $\kappa \geqslant 0$ the ANNNI Hamiltonian also leads to a third phase, the antiphase ("phase 2"), not contained in the training data. In particular, for $\kappa \geqslant 0.5$, we are in a region where typically only phases 1 and 2 are noticeable as the floating phase is too narrow for the small chains we investigated, in accordance with [54]. So, the best the classifier algorithm can do is to output 1 if the phase is indeed 1, and not 1 otherwise, which is interpreted as 2, since the classifier is binary.

As discussed above, for an observation point, both the classical and quantum classifier algorithms will return a normalized probability vector (p_0, p_1) where the assigned phase will correspond to the component with the largest value. As typical with such algorithms, to determine when we are facing a transition, we plot both the probability components and check when they cross, as shown in Fig. 4. As can be seen in Fig. 5, using this approach, the QNN algorithm recovers
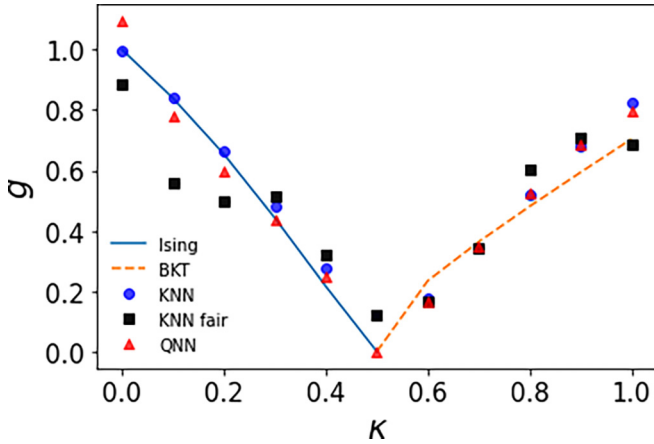
FIG. 5. Phase diagrams (plotted in terms of the dimensionless coupling constants g and κ) produced with diverse (Q)ML algorithms when trained only with $\kappa = 0$: KNN trained with raw data (blue circles); KNN trained with the same preprocessed data as the QNN— fair comparison (black squares); QNN (red triangles), and two different analytical solutions: Ising (solid blue line) and BKT (dashed orange line). All different methods recover the ferro/paramagnetic and paramagnetic/BKT transitions qualitatively well, although, as quantitatively expressed in Table I, the QNN solution yields the smallest MAE and MSE with relation to the analytical approximation, thus being an overall better solution (see main text for more details).

the left part ($\kappa < 0.5$) of the phase diagram, corresponding to the ferromagnetic/paramagnetic transition, very successfully. The precise prediction also holds as we approach the critical point at $\kappa = 0.5$ and $\gamma = 0$ at which a new antiphase appears. However, as we start to increase γ the approximated solutions in (2) and (3) and the QNN predictions start to differ more significantly, even though they remain qualitatively similar.

To benchmark our results we have compared the QNN solution with that obtained by the classical KNN algorithm. As can be seen in Fig. 5, the solution is significantly worse when the classical algorithm is fed with the same preprocessed data as those given to the quantum algorithm. However, if the classical algorithm uses the complete data (not preprocessed) it reaches a similar success in the prediction, even though it is smaller as quantified by shown ahead. Importantly, the quantum classifier performs significantly better at the critical point ($g = 0$, $\kappa = 1/2$).

TABLE I. Performance [average ℓ_1 norm and ℓ_2 norm with relation to the analytical approximations given by Eqs. (2) and (3)] computed for the three main phases and comparing QNN with KNN (using both the preprocessed and complete training data). For the classical KNN we used $k = 7$ and the Euclidean distance, which were found to be the best hyperparameter values using 10-fold cross-validated grid search. The best result is in boldface.

Technique	Average ℓ_1 norm	Average ℓ_2 norm
QNN (preprocessed)	**0.0508(0)**	**0.0036(6)**
KNN (preprocessed)	0.1088(1)	0.0164(1)
KNN (raw data)	0.0509(5)	0.0043(1)

VI. DISCUSSION

The detections of phase and phase transitions of the ANNNI model with both classical as well as quantum heuristic approaches have already been done. In Ref. [36], Canabarro *et al.* showed the possibility of applying both unsupervised and supervised classical techniques to achieve good results. The problem was satisfactorily well solved with unsupervised learning, reducing the use of supervised learning to a validation step. Therefore, they tried using transfer learning of diverse supervised learning algorithms trained solely on nearest-neighbor interactions exhibiting the capacity to discern a novel phase emerging upon the introduction of next-nearest-neighbor interactions. They showed that some of the learners could unveil the Ising as well as the antiphase and the paramagnetic phases. This amalgamation effectively groups the BKT and CIC transitions together, showcasing the robustness of our method. On the other hand, in Ref. [40], Monaco *et al.* used quantum convolutional neural networks by training only on marginal points of the phase diagram represented by integral models. Our approach in this paper is both innovative and complementary. We use a new and simpler quantum machine learning algorithm and apply transfer learning, we test some ideal preprocessing of the data to fit in a real quantum computer, and we train only on $\kappa = 0$, testing on the remaining phase diagram to show the viability of the technique as we discuss here.

We show that with the right preprocessing of the data, the quantum nearest neighbor (QNN) algorithm proposed in [41] allows for the classification of phases in a nontrivial Hamiltonian model. Using two-point correlation functions obtained by exact diagonalization on a small $N = 12$ spins lattice, we could reproduce the main phases of the axial next-nearest-neighbor Ising (ANNNI) model. More precisely, using as training data only the ferromagnetic and paramagnetic phases, we could detect a transition for an antiphase by increasing the interaction strength of the next-nearest neighbor. This is a relevant instance of transfer learning, since using analytical data extracted from the exactly solvable transverse field Ising model, we could explore a nonintegrable region of the Hamiltonian model. This makes the approach computationally cheaper as access to training labels is one of the major bottlenecks for any supervised method.

To benchmark the quality of our quantum machine learning model, we compared it with approximated expressions obtained by various methods. The solution provided by QNN works very well in the ferromagnetic and paramagnetic regions, offering a less precise but still qualitatively reasonable solution as we enter the antiphase. Arguably, however, to assess the quality of a quantum learning method, it is reasonable to compare its performance with that of classical learning algorithms. We performed this comparison, and the results are quite favorable to the quantum approach. Even when we feed the original data (without any preprocessing, a necessary step to reduce the number of qubits in the quantum realization) to classical classifiers, the quantum classifier remains superior, as can be seen in Fig. 5 and Table I. And performing the fairest comparison, obtained when the quantum and classical algorithms see the same preprocessed data, the accuracy of the quantum classifier is significantly higher. Importantly, these

performance comparisons were done on the testing data; that is, we were really evaluating the generalization capability of the different models, which is, after all, what matters the most when one builds a data-driven model. Furthermore, we tuned the classical models' hyperparameters, to guarantee their best performance for the comparison—namely, the values $k = 7$ and Euclidean distance were found to be the best ones via 10-fold cross-validated grid search.

This proof-of-principle (since it was obtained in a simulated or perfect quantum circuit) quantum advantage does not come in terms of algorithmic complexity, but rather in generalization and accuracy, which is of major interest in the context of machine learning. Still, one may interpret the advantage from a different point of view, namely that of sample complexity [76–78]: the quantum algorithm could find the underlying pattern reflected in the dataset with much less information than its classical counterpart, and with better generalization performance. As mentioned before, we can see that as a way of reducing the dataset only to the most representative samples and better explanatory features. Although we focus on a particular Hamiltonian, we believe it leads to relevant general questions: in a statistical learning theoretical sense, how and why is such a sample complexity reduction and consequent quantum advantage achieved? Similar questions have been addressed in recent research [79–81], and further research along this direction, in the particular context of QNN, might lead to new insights. Another clear path is to understand how well the QNN classifier works in real NISQ devices, also considering different Hamiltonian models and increasing the number of qubits and features. In this regard, it would be interesting to consider other data formats, such as the classical shadows [82], efficiently encoding classical data about quantum systems for machine learning purposes [83].

Furthermore, given that the use of machine learning techniques in condensed matter physics and quantum information is a relatively new endeavor, we are still at an exploratory phase, since in most of the literature the main idea is to use models where we already have a good understanding in order to provide a proof of principle that machine learning is a feasible and trustworthy method. Thus, the main idea of our work (and in most of the related works in the literature) is to benchmark the machine learning method rather than learn about the specific model under investigation. Once we are convinced that it works in this case, we can go ahead and apply it to more challenging problems, such as mapping out phase diagrams of models with complex magnetic orders whose nature is not completely settled. This is an approach that has been widely used in the context of classical machine learning, but that, to our knowledge, is much less explored in the context of quantum machine learning. Thus, a proof-of-principle result showing that with the correct preprocessing of data one can achieve an accuracy in classification that is comparable to (even slightly better than) full-power (taking all data into account) classical machine learning is an important step. Moreover, as shown in Fig. 5, when the classical and quantum algorithms take as input the same preprocessed data, the quantum algorithm is significantly better than the classical version, which indicates that in use cases in which there is a limited number of samples and/or limited representation of data points (for instance, *ab initio* data for the training of

machine learning force fields), the quantum algorithm may be superior.

In conclusion, the QNN algorithm [41] was applied to a concrete classification problem in the context of condensed matter physics. By applying this method, we could achieve a quantum model whose generalization performance was superior to its classical counterparts, while using much less information, which represents a quantum advantage in both contexts of generalization and sample complexity. This is the main result of this paper, which opens the way to several discussions concerning the statistical learning theoretical properties of the QNN model.

APPENDIX

To make the paper as self-contained as possible, in this section, we provide a brief description of classification tasks in machine learning [84], followed by a presentation of the classical KNN classifier, and the ensemble methods used in the feature selection procedure.

1. Classification problems

In a classification problem or task, the algorithm has to assign one out of several discrete classes to an observation, according to a rule learned from a set of labeled (previously classified) examples. Exemplary problems include (i) diagnosing a disease given many symptoms; (ii) credit decisions based on features of the applicant; (iii) classifying an email as "spam" or "ham (not-spam)," given its content; (iv) recognition of a handwritten digit; and (v) identification of different phases in many-body systems.

Classification problems are addressed within the supervised learning paradigm, in which a data point p is characterized by an n-dimensional data vector $\boldsymbol{x}^p$, whose components are x_i^p with $i = 1, \ldots, n$ (which we refer to as the *features*, and may be given by binary, integer, or real-valued numbers), as well as by its respective class assignment y^p (which we refer to as the *target*). We say that the feature vector belongs to the *feature space* $\mathcal{X}$, i.e., $\boldsymbol{x}^p \in \mathcal{X}$; and the target belongs to the *target space* $\mathcal{Y}$. Classes, which are discrete, are often encoded by a finite number d of positive integers, that is, $y^p \in \mathcal{Y} = \{0, \ldots, d-1\}$. A particular case of interest is that of binary classification, in which case $d = 2$ and $y^p \in \{0, 1\}$. Within the supervised setting, machine learning

requires that N samples of features and their respective targets be collected—the reason why such data points are often called *observations*. Together, a set of sample observations make up the *training dataset*, $\mathcal{T} = \{(\boldsymbol{x}^p, y^p)\}_{p=1}^N$.

From the perspective of statistical learning theory [76], a supervised learning problem may be posed as follows. It is assumed that there is a process that relates the features and the target, i.e., that determines y from $\boldsymbol{x}$. We can mathematically describe such a process with the so-called *target function* $\mathcal{F}$: $\mathcal{X} \to \mathcal{Y}$.[1] In principle, at least in a theory-driven approach, it could be possible to construct $\mathcal{F}(\boldsymbol{x})$ from first principles. Most often, however, the problems under consideration are too complicated for a theoretical description to be feasible. In such cases, we resort to a data-driven approach, which is commonly implemented with machine learning techniques. In this case, the so-called *hypothesis function* $f_{\mathcal{H}, \boldsymbol{w}}(\boldsymbol{x})$ is introduced, which is intended to approximate the theoretical process $\mathcal{F}$. One important point is that the hypothesis is parametrized by the *parameter vector* $\boldsymbol{w}$. In this setting, $\mathcal{F}$ is unknown, and our job is to approximate it with $f_{\mathcal{H}, \boldsymbol{w}}$; i.e., we want $f_{\mathcal{H}, \boldsymbol{w}} \approx \mathcal{F}$. In order to do so, we are presented with the training data $\mathcal{T} = \{(\boldsymbol{x}^p, y^p)\}_{p=1}^N$, which we assume to have been generated by $\mathcal{F}$; i.e., $y^p = \mathcal{F}(\boldsymbol{x}^p)$.

An important concept is that of a *hypothesis set* $\mathcal{H}$, which is the set of particular functional dependencies that we allow the hypothesis to take. That is, it is the set of all candidate hypotheses, so that $f_{\mathcal{H}, \boldsymbol{w}}(\boldsymbol{x}) \in \mathcal{H}$. Since the functional dependency among candidates on $\boldsymbol{x}$ is the same, it is clear that different candidate hypotheses differ only by the parameter vector $\boldsymbol{w}$. It is the role of the *learning algorithm* to take a hypothesis set alongside with the training data, and then determine the values of the particular parameters such that on the training dataset the hypothesis is as close as possible to the real process; i.e., $f_{\mathcal{H}, \boldsymbol{w}}(\boldsymbol{x}^p) \approx \mathcal{F}(\boldsymbol{x}^p)$.[2] At the end, a single hypothesis is picked, via the determination of the particular $\boldsymbol{w}$. Together, the hypothesis set $\mathcal{H}$ and the learning algorithm make up the so-called *learning model*.

Therefore, machine learning methods aim at estimating $\mathcal{F}$ based on limited knowledge of it, reflected in $\mathcal{T}$. In other words, the learning algorithms try to capture the underlying pattern of the population, based on what is observed on the sample. Naturally, in such a scenario, one major concern is the *generalization* of such an approach, i.e., how confident we are that $f_{\mathcal{H}, \boldsymbol{w}}$ as trained using data from the training set will perform well as a stand-in for $\mathcal{F}$ on data *that was not seen in training*. This question is extensively addressed via tools of statistical learning theory [76–78].

---

[1]To account for noisy targets, it is in fact necessary to introduce the *target distribution* $P(y \mid \boldsymbol{x})$. In this setting, a noisy target may be given by a deterministic portion $\mathcal{F}(\boldsymbol{x}) = \mathbb{E}(y \mid \boldsymbol{x})$ as well as by the noise $y - \mathcal{F}(\boldsymbol{x})$. If there is no noise, a special case occurs for $P(y \mid \boldsymbol{x}) = 0$ everywhere, except for $y = \mathcal{F}(\boldsymbol{x})$.

[2]Operationally, the determination of the particular $\boldsymbol{w}$ that best fits $f_{\mathcal{H}, \boldsymbol{w}}$ to $\mathcal{T}$ is achieved via the solution of an optimization problem, namely, the optimization of the *cost function*, which is an error function, accounting for the differences between predicted targets $f_{\mathcal{H}, \boldsymbol{w}}(\boldsymbol{x}^p)$ and actual targets y^p. For details, see [77,78].

Pragmatically, solving a problem with a machine learning approach is a good idea if the following hold:

(i) There are data available for the problem at hand; for an empirical modeling approach such as machine learning, data are absolutely essential. Without sample data to learn from, it is simply impossible to apply any machine learning technique.

(ii) A pattern exists and is reflected in the sample; after all, if there is no pattern to be learned (or, pragmatically, if the pattern is not captured in the training dataset), not even the best learning technique will be able to produce a good model.

(iii) A theoretical description is not available; there will always be errors associated with a machine learning model since it is a method to empirically estimate the true process. Therefore, if a full theoretical description is available, its use would be preferable, since such estimation errors would not be present.

The aim of our work is to show that a quantum classification algorithm can perform well in the classification of phases of a nontrivial Hamiltonian, the ANNNI model, a problem that fits the requirements listed above. The training data are given by the correlations between the spins of different sites in the network. Those features are experimentally accessible and form the basis of any attempt to classify phases in many-body systems, thus fulfilling the first two criteria listed above. As for the third criterion, as discussed in Sec. II, there are no analytical tools for witnessing phase transitions in general, a problem that typically relies on perturbation theory, approximations, or exact diagonalization on small chains.

2. *K* nearest neighbors classifier

We now present the fundamentals of the K nearest neighbors (KNN) classifier [84], the classical counterpart of the QNN algorithm described in Sec. III.

Given a training dataset $\mathcal{T} = \{(\boldsymbol{x}^p, y^p)\}_{p=1}^N$ and an unclassified observation $\boldsymbol{x}_{\text{in}}$ encoding n features, the goal is to assign to the input vector $\boldsymbol{x}_{\text{in}}$ a given class in $\mathcal{Y}$, using information from the training data. A possible approach for such a classification is to use some distance measure and then assign to the input vector the class whose members are closest to it. That is the essence of the KNN classifier, with the particularity that only the K nearest neighbors are considered for the classification (see Fig. 6); i.e., the input observation is predicted to belong to the class which is the most representative among its K nearest neighbors.

In short, the KNN algorithm proceeds via the following steps: (1) Set the value for K (the number of neighbors to be considered). (2) For every instance x (query example) in the training set, (i) calculate the distance between the query example and true output, and (ii) add the distance and the corresponding index of the query example to an ordered collection. (3) Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances. (4) Pick the first K entries from the sorted collection. (5) Get the labels of the selected K entries. (6) Return the mode of the K labels (classification tasks).

Apart from K, which is an obviously important hyperparameter, it is also crucial to choose an appropriate distance metric, so that the determination of neighbors' observations is meaningful. Several distance metrics are possible, each
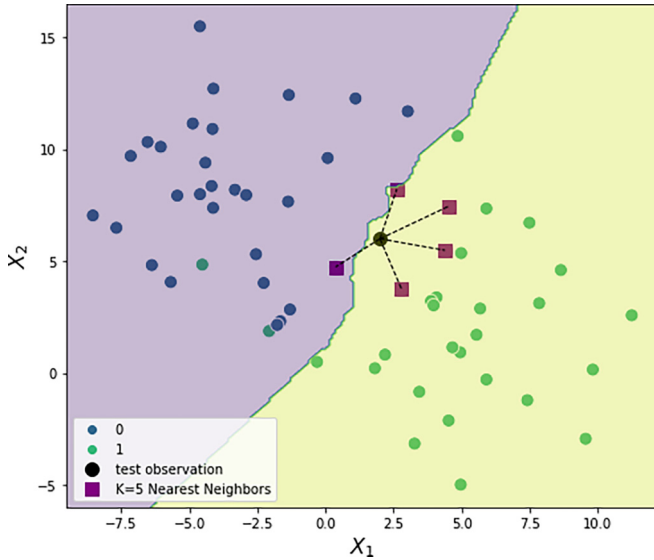
FIG. 6. Pictorial illustration of KNN classification considering $K = 5$. Based on the distances from the five training points closest to the input test observation, this input would be classified as class 1 (green). The decision boundary (surface separating the two classes) is also depicted.

exploring a particular aspect of the dataset structure. In this work, we will consider the Hamming distance, which is used when the features are encoded as binary strings or vectors. It is defined as the number of different bits, or components, among a pair of binary strings. For instance, taking $a = 1011101$ and $b = 1001001$, we have $d_H(a, b) = 2$, since only the third and fifth components differ.

It is worth noticing that the KNN classifier is a "lazy algorithm," given that an explicit training procedure does not take place—which also means that no explicit hypothesis from a hypothesis set will be determined. In fact, each input observation is classified at a time. This is also the case for the quantum nearest neighbors algorithm.

We trained the KNN model using sklearn's implementation *sklearn.neighbors.KNeighborsClassifier* [75]. We ran a cross-validated grid search to select the best values for K, while we used the Euclidean distance for the model trained with the raw features, and Hamming distance for the model trained with the preprocessed, binary data.

3. Random forest and extra trees

Random forest [71] is an ensemble method used for classification tasks, consisting of a multitude of decision trees. In the end, it outputs the most voted class of the individual decision trees, mimicking the ancient idea behind the "wisdom of the crowds." Decision tree learning consists of the construction of a decision tree from labeled training instances. The root is the training data itself. The branches are the output (Boolean output) of tests for the attributes. Finally, the leaf nodes are the estimated class. Therefore, for a given input x, a label y is predicted after percolating from the root up to the leaf. It must be noted, however, that an individual decision tree algorithm is a weak learner since it produces high-bias errors.

To implement a combination of decision trees, we first partition the training set into M smaller subsets $B_1, B_2, \ldots, B_M$ that are randomly chosen and may be repeated. If the subsets are large enough for training a specific learner, they can be aggregated to create an ensemble predictor. For classification tasks, we take a majority vote on all the predictions. This process was introduced in Ref. [73] and is called *bagging*, an acronym for bootstrap aggregation, and was shown to reduce the variance (out-of-sample error) without increasing the bias (in-sample error) [2].

Within this context, the key idea of random forest is to perform subsets of the features. Now, the trees randomly choose a k number of N total features with $k < N$. This bagging of features reduces the correlation between the various decision trees, contributing to better modeling. The code employed in our work was made using the package *sklearn.ensemble.RandomForestClassifier* [75], using parameters given by *max_depth*=None, *n_estimators*=1000.

Operationally, the calculation proceeds as follows. For every tree in the ensemble, we traverse its internal nodes and calculate the impurity reduction that the respective split produces. Naturally, the impurity reduction depends on the criterion used to measure the quality of a split. For classification problems, the scikit-learn implementation supports the Gini impurity and the information gain (entropy) criteria. For this reason, even though the "impurity reduction" term is often used, if we use the entropy criterion, we are not dealing with a proper impurity, and should actually use the term "error reduction."

The impurity reduction Δ_ϵ is calculated as the difference between the impurity before the split and after the split (which is given by the averaged impurities of each one of the two branches); that is, $\Delta_\epsilon = \epsilon_{\text{before split}} - \epsilon_{\text{after split}}$. This impurity reduction is multiplied by the proportion of observations passing through the node, which is how the average becomes weighted. Recall that in each node we have a feature; therefore, the reduction is associated with a given feature. After this procedure, we have a given score for each feature, when we analyze a single tree. To get the final scores, we average the scores over all trees in the ensemble. Notice that the features that more commonly appear in the nodes will naturally have a higher total score, which intuitively makes sense: if a given feature appears in many nodes, and is responsible for a higher impurity decrease, it is a "better" feature, for the purpose of classification.

As mentioned in Sec. IV, we employed a modification of random forest, known as "extremely randomized trees" ("extra trees"). The major difference is that, in the case of extra trees, additional randomness is induced in the way of computing the tree splits: instead of using a fixed better-separating threshold (which is the case for traditional random forest), we randomly generate thresholds for each candidate feature for the split. Introduced in [72], the algorithm yields smaller variance, although the bias is often slightly increased. But, for the purpose of feature selection (which is how we employed this algorithm), it is often a better choice, since the extra randomness allows us to choose the features by their actual discriminative power, avoiding the undesired effect of a fixed threshold.
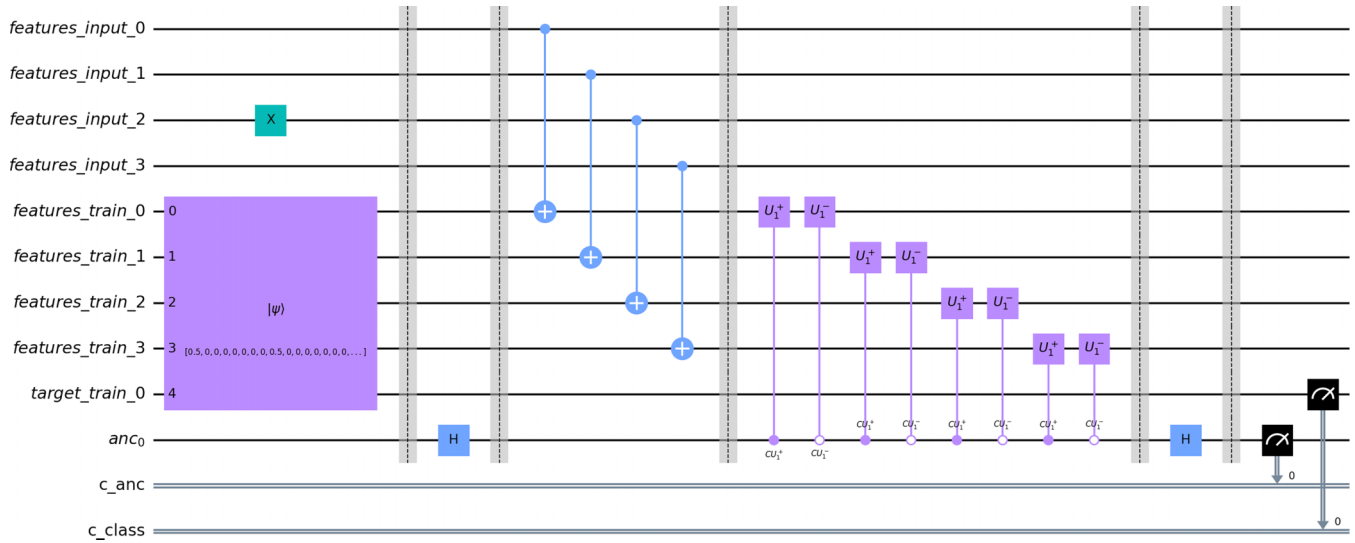
FIG. 7. An illustration of the QNN quantum circuit for a particular dataset of $N = 4$ observations and $n = 4$ features. We plot a smaller circuit here for the sake of space. The test point is given by the binary string 0010. Each barrier in the plot indicates an important step of the quantum algorithm (as described in Sec. III), after which the quantum state is the one given respectively by Eqs. (6), (7), (10), (12), and (14). The measurement yields the probabilities as described in Eqs. (15) and (16).

In practice, once we have an importance score for each feature, we can pick the ones that are the most important. We use the implementation from scikit-learn [75], and chose the 4 most important features, among the 198 original ones.

4. Illustration of the quantum circuit

In Fig. 7 the QNN circuit is illustrated for the following training dataset, consisting of $N = 4$ observations of $n = 4$ binary features:

		X			y
0	0	0	0		0
0	0	0	1		0
1	1	1	0		1
1	1	1	1		1

and the test observation 0010. The superposition state of all the training data points [Eq. (5)] is represented by the black-box state preparation routine given by the *initialize* Qiskit instruction. In practice, this instruction is decomposed into elementary gates during the circuit transpilation.

Notice how the full circuit for this problem instance uses $2n + 2 = 10$ qubits and 2 classical registers and has a depth of 13 (which considers the state preparation routine as depth 1; in practice, the superposition construction takes time $O(Nn)$ [65]). Indeed, that matches the $O(n)$ depth expected for this circuit, which mainly arises from the application of the $\mathcal{U}$ operator [Eq. (11)], which demands $2n$ sequential gates.

Finally, we chose to illustrate the quantum circuit for this simpler problem instance instead of illustrating one of the actual quantum circuits built in the study, because those have 18 qubits, which would be much more difficult to visualize.

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, 2016).

[2] P. Mehta, C.-H. Wang, A. G. R. Day, C. Richardson, M. Bukov, C. K. Fisher, and D. J. Schwab, A high-bias, low-variance introduction to machine learning for physicists, Phys. Rep. **810**, 1 (2019).

[3] N. S. Altman, An introduction to kernel and nearest-neighbor nonparametric regression, Am. Stat. **46**, 175 (1992).

[4] R. S. Olson, N. Bartley, R. J. Urbanowicz, and J. H. Moore, Evaluation of a tree-based pipeline optimization tool for automating data science, in *Proceedings of the Genetic and Evolutionary Computation Conference 2016, GECCO'16* (ACM, New York, 2016), pp. 485–492.

[5] J. D. Kelleher, B. MacNamee, and A. D'Arcy, *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms,* *Worked Examples, and Case Studies* (MIT Press, Cambridge, MA, 2015).

[6] T. Charnock and A. Moss, Supernovae: Photometric classification of supernovae, Astrophysics Source Code Library, https://ascl.net.

[7] R. von Marttens, L. Casarini, N. R. Napolitano, S. Wu, V. Amaro, R. Li, C. Tortora, A. Canabarro, and Y. Wang, Inferring galaxy dark halo properties from visible matter with machine learning, Mon. Not. R. Astron. Soc. **516**, 3924 (2022).

[8] M. Carrillo, J. A. González, M. Gracia-Linares, and F. S. Guzmán, Time series analysis of gravitational wave signals using neural networks, J. Phys.: Conf. Ser. **654**, 012001 (2015).

[9] G. Torlai, G. Mazzola, J. Carrasquilla, M. Troyer, R. Melko, and G. Carleo, Neural-network quantum state tomography, Nat. Phys. **14**, 447 (2018).

[10] R. Iten, T. Metger, H. Wilming, L. del Rio, and R. Renner, Discovering physical concepts with neural networks, Phys. Rev. Lett. **124**, 010508 (2020).

[11] G. Carleo and M. Troyer, Solving the quantum many-body problem with artificial neural networks, Science **355**, 602 (2017).

[12] G. Torlai and R. G. Melko, Learning thermodynamics with Boltzmann machines, Phys. Rev. B **94**, 165134 (2016).

[13] L. M. Ghiringhelli, J. Vybiral, S. V. Levchenko, C. Draxl, and M. Scheffler, Big data of materials science: Critical role of the descriptor, Phys. Rev. Lett. **114**, 105503 (2015).

[14] J. Carrasquilla and R. G. Melko, Machine learning phases of matter, Nat. Phys. **13**, 431 (2017).

[15] P. Broecker, J. Carrasquilla, R. G. Melko, and S. Trebst, Machine learning quantum phases of matter beyond the fermion sign problem, Sci. Rep. **7**, 8823 (2017).

[16] K. Ch'ng, J. Carrasquilla, R. G. Melko, and E. Khatami, Machine learning phases of strongly correlated fermions, Phys. Rev. X **7**, 031038 (2017).

[17] D.-L. Deng, X. Li, and S. D. Sarma, Machine learning topological states, Phys. Rev. B **96**, 195145 (2017).

[18] P. Huembeli, A. Dauphin, and P. Wittek, Identifying quantum phase transitions with adversarial neural networks, Phys. Rev. B **97**, 134109 (2018).

[19] C. B. D. Goes, A. Canabarro, E. I. Duzzioni, and T. O. Maciel, Automated machine learning can classify bound entangled states with tomograms, Quantum Inf. Process. **20**, 99 (2021).

[20] F. Arute *et al.*, Quantum supremacy using a programmable superconducting processor, Nature (London) **574**, 505 (2019).

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

[22] A. Bouland, W. van Dam, H. Joorati, I. Kerenidis, and A. Prakash, Prospects and challenges of quantum finance, arXiv:2011.06492.

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

[24] M. Schuld, I. Sinayskiy, and F. Petruccione, The quest for a quantum neural network, Quantum Inf. Process. **13**, 2567 (2014).

[25] S. Lloyd, M. Mohseni, and P. Rebentrost, Quantum algorithms for supervised and unsupervised machine learning, arXiv:1307.0411.

[26] N. Wiebe, A. Kapoor, and K. M. Svore, Quantum nearest-neighbor algorithms for machine learning, Quantum Inf. Comput. **15**, 318 (2015).

[27] D. Dong, C. Chen, H. Li, and T.-J. Tarn, Quantum reinforcement learning, IEEE Trans. Syst. Man Cybern. Part B: Cybern. **38**, 1207 (2008).

[28] D. Horn and A. Gottlieb, Algorithm for data clustering in pattern recognition problems based on quantum mechanics, Phys. Rev. Lett. **88**, 018702 (2001).

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

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

[31] J. Carrasquilla and G. Torlai, Neural networks in quantum many-body physics: A hands-on tutorial, arXiv:2101.11099.

[32] T. Kriváchy, Y. Cai, D. Cavalcanti, A. Tavakoli, N. Gisin, and N. Brunner, A neural network oracle for quantum nonlocality problems in networks, npj Quantum Inf. **6**, 1 (2020).

[33] N. D'Alessandro, B. Polacchi, G. Moreno, E. Polino, R. Chaves, I. Agresti, and F. Sciarrino, Machine-learning-based device-independent certification of quantum networks, Phys. Rev. Res. **5**, 023016 (2023).

[34] E. Polino, D. Poderini, G. Rodari, I. Agresti, A. Suprano, G. Carvacho, E. Wolfe, A. Canabarro, G. Moreno, G. Milani *et al.*, Experimental nonclassicality in a causal network without assuming freedom of choice, Nat. Commun. **14**, 909 (2023).

[35] A. Canabarro, S. Brito, and R. Chaves, Machine learning nonlocal correlations, Phys. Rev. Lett. **122**, 200401 (2019).

[36] A. Canabarro, F. F. Fanchini, A. L. Malvezzi, R. Pereira, and R. Chaves, Unveiling phase transitions with machine learning, Phys. Rev. B **100**, 045129 (2019).

[37] A. V. Uvarov, A. S. Kardashin, and J. D. Biamonte, Machine learning phase transitions with a quantum processor, Phys. Rev. A **102**, 012415 (2020).

[38] M. Lazzarin, D. E. Galli, and E. Prati, Multi-class quantum classifiers with tensor network circuits for quantum phase recognition, Phys. Lett. A **434**, 128056 (2022).

[39] P. R. C. Guimarães, J. A. Plascak, F. C. Sá Barreto, and J. Florencio, Quantum phase transitions in the one-dimensional transverse Ising model with second-neighbor interactions, Phys. Rev. B **66**, 064413 (2002).

[40] S. Monaco, O. Kiss, A. Mandarino, S. Vallecorsa, and M. Grossi, Quantum phase detection generalization from marginal quantum neural network models, Phys. Rev. B **107**, L081105 (2023).

[41] M. Schuld, I. Sinayskiy, and F. Petruccione, Quantum computing for pattern classification, in *PRICAI 2014: Trends in Artificial Intelligence*, edited by D. N. Pham and S. B. Park, Lecture Notes in Computer Science, Vol. 8862 (Springer, Cham, 2014), pp. 208–220.

[42] W. Selke, The ANNNI model—theoretical analysis and experimental application, Phys. Rep. **170**, 213 (1988).

[43] S. Suzuki, J. Inoue, and B. K. Chakrabarti, *Quantum Ising Phases and Transitions in Transverse Ising Models* (Springer, Berlin, 2013).

[44] J.-J. Wen, W. Tian, V. O. Garlea, S. M. Koohpayeh, T. M. McQueen, H.-F. Li, J.-Q. Yan, J. A. Rodriguez-Rivera, D. Vaknin, and C. L. Broholm, Disorder from order among anisotropic next-nearest-neighbor Ising spin chains in SrHo$_2$O$_4$, Phys. Rev. B **91**, 054424 (2015).

[45] E. Sela, M. Punk, and M. Garst, Dislocation-mediated melting of one-dimensional Rydberg crystals, Phys. Rev. B **84**, 085434 (2011).

[46] F. Hassler and D. Schuricht, Strongly interacting Majorana modes in an array of Josephson junctions, New J. Phys. **14**, 125018 (2012).

[47] A. Milsted, L. Seabra, I. C. Fulga, C. W. J. Beenakker, and E. Cobanera, Statistical translation invariance protects a topological insulator from interactions, Phys. Rev. B **92**, 085139 (2015).

[48] C. Karrasch and D. Schuricht, Dynamical phase transitions after quenches in nonintegrable models, Phys. Rev. B **87**, 195104 (2013).

[49] J. H. Robertson, R. Senese, and F. H. L. Essler, A simple theory for quantum quenches in the ANNNI model, SciPost Phys. **15**, 032 (2023).

[50] A. Haldar, K. Mallayya, M. Heyl, F. Pollmann, M. Rigol, and A. Das, Signatures of quantum phase transitions after quenches in quantum chaotic one-dimensional systems, Phys. Rev. X **11**, 031062 (2021).

[51] J. Villain and P. Bak, Two-dimensional Ising model with competing interactions: Floating phase, walls and dislocations, J. Phys. France **42**, 657 (1981).

[52] D. Allen, P. Azaria, and P. Lecheminant, A two-leg quantum Ising ladder: A bosonization study of the ANNNI model, J. Phys. A: Math. Gen. **34**, L305 (2001).

[53] H. Rieger and G. Uimin, The one-dimensional ANNNI model in a transverse field: Analytic and numerical study of effective Hamiltonians, Z. Phys. B: Condens. Matter **101**, 597 (1996).

[54] M. Beccaria, M. Campostrini, and A. Feo, Evidence for a floating phase of the transverse ANNNI model at high frustration, Phys. Rev. B **76**, 094410 (2007).

[55] A. Nagy, Exploring phase transitions by finite-entanglement scaling of MPS in the 1D ANNNI model, New J. Phys. **13**, 023015 (2011).

[56] K. Okamoto and K. Nomura, Fluid-dimer critical point in $S = 12$ antiferromagnetic Heisenberg chain with next nearest neighbor interactions, Phys. Lett. A **169**, 433 (1992).

[57] K. Nomura, Correlation functions of the 2D sine-Gordon model, J. Phys. A: Math. Gen. **28**, 5451 (1995).

[58] N. R. Draper and H. Smith, "Dummy" variables, in *Applied Regression Analysis* (Wiley, 1998), Chap. 14.

[59] M. Schuld, Supervised quantum machine learning models are kernel methods, arXiv:2101.11020.

[60] V. V. Shende, S. S. Bullock, and I. L. Markov, Synthesis of quantum-logic circuits, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **25**, 1000 (2006).

[61] Qiskit: An open-source framework for quantum computing (2021), doi:10.5281/zenodo.2573505.

[62] V. V. Shende, S. S. Bullock, and I. L. Markov, Synthesis of quantum logic circuits, in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 25 (IEEE, 2006), pp. 1000–1010.

[63] F. Mozafari, H. Riener, M. Soeken, and G. D. Micheli, Efficient Boolean methods for preparing uniform quantum states, IEEE Trans. Quantum Eng. **2**, 1 (2021).

[64] A. Shukla and P. Vedula, An efficient quantum algorithm for preparation of uniform quantum superposition states, Quantum Inf. Process. **23**, 38 (2024).

[65] D. Ventura and T. Martinez, Quantum associative memory, arXiv:quant-ph/9807053.

[66] H.-Y. Huang, R. Kueng, and J. Preskill, Information-theoretic bounds on quantum advantage in machine learning, Phys. Rev. Lett. **126**, 190505 (2021).

[67] D. Aharonov, J. Cotler, and X.-L. Qi, Quantum algorithmic measurement, Nat. Commun. **13**, 887 (2022).

[68] L. Schatzki, A. Arrasmith, P. J. Coles, and M. Cerezo, Entangled datasets for quantum machine learning, arXiv:2109.03400.

[69] M. Chehimi and W. Saad, Quantum federated learning with quantum data, arXiv:2106.00005.

[70] André J. Ferreira-Martins and L. Silva, QML for the ANNNI model, https://github.com/askery/qml_ANNNI (2023).

[71] L. Breiman, Random forests, Mach. Learn. **45**, 5 (2001).

[72] P. Geurts, D. Ernst, and L. Wehenkel, Extremely randomized trees, Mach. Learn. **63**, 3 (2006).

[73] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, 1st ed. (Chapman and Hall/CRC, New York, 1984).

[74] J. MacQueen, Some methods for classification and analysis of multivariate observations, in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics* (University of California Press, Berkeley, 1967), pp. 281–297.

[75] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. **12**, 2825 (2011).

[76] V. Vapnik, *The Nature of Statistical Learning Theory*, Information Science and Statistics (Springer, New York, 1999).

[77] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*, Adaptive Computation and Machine Learning Series (MIT Press, Cambridge, MA, 2018).

[78] R. F. Mello and M. A. Ponti, *Machine Learning: A Practical Approach on the Statistical Learning Theory* (Springer International Publishing, 2018).

[79] M. C. Caro, H.-Y. Huang, M. Cerezo, K. Sharma, A. Sornborger, L. Cincio, and P. J. Coles, Generalization in quantum machine learning from few training data, Nat. Commun. **13**, 4919 (2022).

[80] M. C. Caro, E. Gil-Fuster, J. J. Meyer, J. Eisert, and R. Sweke, Encoding-dependent generalization bounds for parametrized quantum circuits, Quantum **5**, 582 (2021).

[81] M. Schuld, R. Sweke, and J. J. Meyer, Effect of data encoding on the expressive power of variational quantum-machine-learning models, Phys. Rev. A **103**, 032430 (2021).

[82] J. Smith and J. Johnson, Classical shadows: Efficiently encoding classical data for quantum machine learning, Quantum Inf. Process. **19**, 102 (2020).

[83] M. Rath and H. Date, Quantum data encoding a comparative analysis of classical-to-quantum mapping techniques and their impact on machine learning accuracy, arXiv:2311.10375 (2023).

[84] T. Cover and P. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory **13**, 21 (1967).