Learning complexity gradually in quantum machine learning models

Erik Recio-Armengol,[1,2] Franz J. Schreiber,[3] Jens Eisert,[3,4,5] and Carlos Bravo-Prieto[3,*]

[1]*ICFO-Institut de Ciencies Fotoniques, The Barcelona Institute of Science and Technology, 08860 Castelldefels (Barcelona), Spain*

[2]*Eurecat, Centre Tecnològic de Catalunya, Multimedia Technologies, 08005 Barcelona, Spain*

[3]*Dahlem Center for Complex Quantum Systems, Freie Universität Berlin, 14195 Berlin, Germany*

[4]*Fraunhofer Heinrich Hertz Institute, 10587 Berlin, Germany*

[5]*Helmholtz-Zentrum Berlin für Materialien und Energie, 14109 Berlin, Germany*

(Received 9 January 2026; accepted 19 May 2026; published 6 July 2026)

Quantum machine learning is an emergent field that continues to draw significant interest for its potential to offer improvements over classical algorithms in certain areas. However, training quantum models remains a challenging task, largely because of the difficulty in establishing an effective inductive bias when solving high-dimensional problems. In this work, we propose a training framework that prioritizes informative data points over the entire training set. This approach draws inspiration from classical techniques such as *curriculum learning* and *hard example mining* to introduce an additional inductive bias through the training data itself. By selectively focusing on informative samples, we aim to steer the optimization process toward more favorable regions of the parameter space. This data-centric approach complements existing strategies such as warm-start initialization methods, providing an additional pathway to address performance challenges in quantum machine learning. We provide theoretical insights into the benefits of prioritizing informative data for quantum models, and we validate our methodology with numerical experiments on selected recognition tasks of quantum phases of matter. Our findings indicate that this strategy could be a valuable approach for improving the performance of quantum machine learning models.

DOI: 10.1103/wnlr-q9xd

I. INTRODUCTION

In light of the impressive success of classical machine learning and the ever-improving experimental means to perform quantum computing, the study of the potential and limitations of *quantum machine learning* (QML) has become a topic of widespread interest. Quantum systems are known to perform learning tasks that are beyond the capabilities of classical computers [1–4]. However, there is growing evidence that the training of quantum learning models is hindered by severe problems not present in classical machine learning. Recent investigations have highlighted that the cost landscape of quantum learning models contains many poor minima and vast regions of vanishing gradients, so-called barren plateaus, which preclude trainability [5–11].

To address these training challenges and enhance the performance of quantum learning models, it is crucial to introduce a favorable inductive bias [12–14]. Indeed, barren plateaus arise from random initialization in high-dimensional problems, where the cost landscape becomes flat, and gradients vanish due to the concentration of measure phenomenon. As such, incorporating a well-advised inductive bias is crucial for achieving reasonable performance. In quantum models, inductive bias has typically been introduced through two main strategies: (1) Designing specific quantum circuit architectures that exploit problem symmetries or structures [15–21], and (2) employing smart initialization techniques to set the model parameters near to optimal values, often referred to as warm starts [22–30]. Both approaches aim to facilitate the optimization process toward favorable regions of the parameter space, thereby improving trainability and convergence.

However, these approaches primarily focus on modifying the model architecture or parameter initialization, while the role of training data in introducing inductive bias has been largely overlooked. Traditionally, training examples are presented to the learner in random order, a practice common in both classical and quantum machine learning. Nonetheless, there is an increasing awareness in classical machine learning that imposing a specific order on the training data can speed up convergence and lead to better local minima. Typically, this ordering is guided by a scoring function, where training starts with examples that have low scores and gradually incorporates data with higher scores as the training progresses. This idea has been explored in different forms, the most prominent examples being *curriculum learning* [31], which was recently applied to quantum models [32], and *hard example mining* [33]. These strategies primarily differ in their approach to devising scoring functions. Hard example mining focuses on presenting the learner with difficult examples, while curriculum learning, inspired by human learning processes, starts

*Contact author: c.bravo.prieto@fu-berlin.de

with easier examples and progressively introduces more challenging ones. Despite their differences, both strategies have demonstrated success across various classical machine learning domains, including natural language processing, computer vision, and others [33–41]. Therefore, adopting an agnostic stance on the design principles is essential, as their quality is ultimately determined by their performance in practice.

Inspired by these powerful classical methods, this work explores an approach to incorporate inductive bias into quantum learning models through the quantum data itself. By carefully selecting, structuring, and presenting the quantum data, we aim to steer the optimization process away from unfavorable regions of the parameter space and guide the model toward better generalization and faster convergence. This data-centric perspective offers a complementary strategy to existing methods focusing on circuit design and parameter initialization. Our contributions include developing a unified and intuitive quantum analog of curriculum learning and hard example mining, tailored to the characteristics of quantum models. Through theoretical analysis and empirical validation, we demonstrate that leveraging the structure and presentation of quantum data can improve the trainability and performance of quantum learning models. This approach not only enhances optimization but also opens avenues for introducing inductive bias in quantum machine learning. More broadly, our work represents a step toward importing sophisticated methods from classical machine learning into the quantum domain, reflecting the growing maturity of the field.

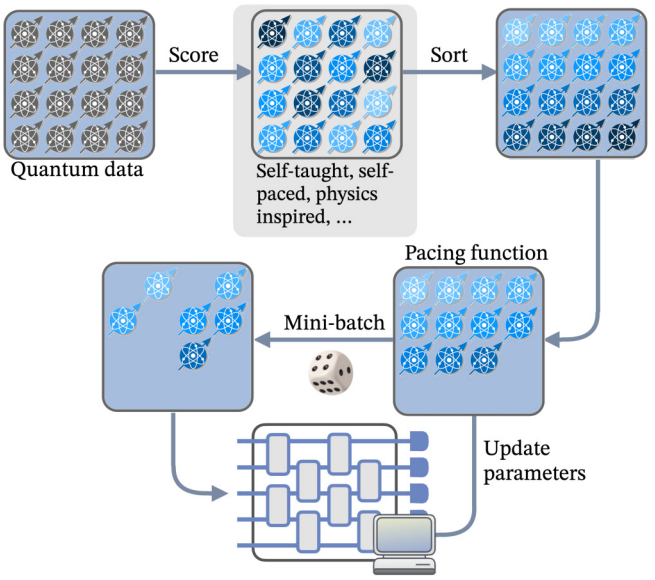

FIG. 1. The training framework begins by scoring quantum data points according to a predefined scoring function (e.g., self-taught, self-paced, or physics inspired). Data points are sorted by score, and the pacing function determines how much of the dataset is accessible at each training epoch. Minibatches are then sampled from the available subset and used to update the model parameters iteratively, enabling a gradual change in learning complexity throughout training.

II. THEORY

In this section, we develop the theoretical underpinning of our framework, starting by laying out a number of preliminaries.

A. Learning setup and training procedure

We consider a supervised learning scenario involving training data consisting of a collection of quantum state vectors $\{|\psi_i\rangle \in \mathcal{H}\}$ and their respective labels $y_i \in \mathcal{Y} \subseteq \mathbb{R}$. Our quantum learning model, represented as $f_{\boldsymbol{\vartheta}} : \mathcal{H} \to \mathcal{Y}$, has trainable parameters $\boldsymbol{\vartheta} = (\vartheta_1, \vartheta_2, \dots)$. The training set is denoted by $\mathbb{X} = \{X_i\}_{i=1}^N = \{|\psi_i\rangle, y_i\}_{i=1}^N$. Depending on the specific learning task, the collection of quantum state vectors $\{|\psi_i\rangle\}$ may naturally arise or encode classical training data through a particular embedding process. Our goal is to identify a set of parameters $\boldsymbol{\vartheta}$ that minimizes a given loss function $\ell(\boldsymbol{\vartheta}; (|\psi\rangle, y))$.

Traditionally, the entire training dataset $\mathbb{X}$ is accessible from the start of the training. A common approach is to randomize the order of $\mathbb{X}$, split it into minibatches, and perform gradient-based optimization. Variations of this setup exist, but typically, the order in which training data are presented to the learner is not prioritized. Here, we deviate from this conventional setup by initially limiting the learner to a subset of the training data, gradually increasing access to the full dataset. This framework can be visualized in Fig. 1.

Practically, this is done by introducing a scoring and pacing function. The scoring function determines the order in which data points are added to the subset accessible to the learner by assigning a score $s(|\psi\rangle)$ between 0 and 1 to each data point $|\psi\rangle$.

Definition 1 (Scoring function). Given a dataset $\{\boldsymbol{x}_i, y_i\}_{i=1}^N$, with $\boldsymbol{x}_i \in \mathcal{X}$, a scoring function is defined as $s : \mathcal{X} \to [0, 1]$.

For two data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, if $s(\boldsymbol{x}_i) \leqslant s(\boldsymbol{x}_j)$, then $\boldsymbol{x}_j$ should not be available to the learner before $\boldsymbol{x}_i$. The pacing function essentially dictates the rate at which new data become available to the learner by assigning to each epoch a fraction of the training data. Importantly, our framework is also immediately compatible with training on classical data. For example, some methods embed classical data into quantum states, while others, such as data reuploading [42,43], encode it directly into quantum circuits. Our framework accommodates such approaches, as the scoring function can operate on either the classical data or their corresponding embedding quantum states without conceptual inconsistency.

Definition 2 (Pacing function). Let $\mathbb{T} := \{1, 2, \dots, T\}$ enumerate the T training epochs. A pacing function is defined as a function $p : \mathbb{T} \to [0, 1]$.

Therefore, in epoch t, the learner has access to the fraction $p(t)$ of the entire dataset based on the scores of $s(|\psi\rangle)$.

B. Scoring and pacing functions

Scoring functions evaluate the training data, often guided according to notions of difficulty or complexity. While we take an agnostic stance on the idea of scoring functions as difficulty measures, it is noteworthy that those inspired by complexity or difficulty, as previously mentioned, have a successful track record in classical machine learning, often capturing data structures that enable better learning.

We briefly review scoring functions used in practice, distinguishing between domain-agnostic and domain-specific methods. Domain-agnostic scoring functions, despite their name, are typically tied to the training set and are often learned. This involves initially training a secondary learner on the data and using the resulting loss values as a scoring metric. For example, a learner $f_{\boldsymbol{\vartheta}}$ may first undergo conventional training, and its hypothesis is subsequently used as a scoring function. Alternative architectures, such as support vector machines, can also be adapted in this way. However, this approach can be computationally intensive due to the additional learner.

To mitigate such costs, self-paced learning offers a simpler alternative. Here, the loss $\ell_{\boldsymbol{\vartheta}}$ of the current hypothesis $f_{\boldsymbol{\vartheta}}$ acts as the scoring function ordering the training data of the subsequent epoch. This method preserves the domain-agnostic nature of scoring while avoiding the overhead of training an auxiliary learner. In contrast, domain-specific scoring functions leverage domain expertise to define metrics, often inspired, as mentioned previously, by complexity measures such as input length in text-based tasks or the number of objects in an image.

Unlike scoring functions, pacing functions dictate the rate at which training examples are introduced. They are generally domain agnostic and are chosen to be monotonically increasing. Linear pacing functions are the most common baseline, but nonlinear alternatives, such as the root-p and geometric progression functions [34,39,44], are also employed. These can be categorized by their growth relative to linear functions: Slower-growing functions extend training on high-scoring examples while faster-growing ones prioritize low-scoring data.

III. RESULTS

In this section, we aim to provide theoretical justification for the use of this learning setup and present theoretical results within the framework proposed.

A. Analytical results on the mitigation of barren plateaus

In this subsection, we demonstrate how a learning setup defined by a scoring function s and a pacing function p can address challenges during early training epochs and improve convergence rates. We show that carefully selecting and ordering training data can enhance the training dynamics of parametrized quantum models.

Proposition 1 (Mitigation of barren plateaus). Let $\{|\psi_i\rangle \in \mathcal{H}\}$ be a collection of quantum state vectors. Let $f_{\boldsymbol{\vartheta}} : \mathcal{H} \to \mathcal{Y}$ be a parametrized quantum model with parameters $\boldsymbol{\vartheta} \in \Theta$ and $\mathcal{Y} \subseteq \mathbb{R}$ the label space. Consider a loss function $\ell(\boldsymbol{\vartheta}; (|\psi\rangle, y))$ associated with data points $\{(|\psi_i\rangle, y_i) \in \mathcal{H} \times \mathcal{Y}\}$. Suppose there exists a scoring function $s : \mathcal{H} \to [0, 1]$ such that the expected squared norm of the gradient

$$G(z) = \mathbb{E}[\|\nabla_{\boldsymbol{\vartheta}} \ell(\boldsymbol{\vartheta}; (|\psi\rangle, y)\|^2 \mid s(\psi) = z] \tag{1}$$

is a nonincreasing function of s. Then, this learning setup defined by s increases the expected squared gradient magnitude during early training epochs compared to standard training (uniform access to the entire dataset).

The proof is given in Appendix A. By prioritizing data points with lower scores—corresponding to higher expected squared gradient norms—we mitigate the effects of barren plateaus during the initial stages of training. We later give an explicit example of a physically inspired scoring function s such that Proposition 1 holds.

B. Analytical results on faster convergence

Next, we show that this learning setup could also improve the convergence of the model.

Proposition 2 (Faster convergence). Let $\{|\psi_i\rangle\} \in \mathcal{H}$ be a collection of quantum state vectors. Let $f_{\boldsymbol{\vartheta}} : \mathcal{H} \to \mathcal{Y}$ be a parametrized quantum model with parameters $\boldsymbol{\vartheta} \in \Theta$ and $\mathcal{Y} \subseteq \mathbb{R}$ the label space. Consider a convex loss function with Lipschitz continuous gradients $\ell(\boldsymbol{\vartheta}; (|\psi\rangle, y))$ associated with data points $\{(|\psi_i\rangle, y_i) \in \mathcal{H} \times \mathcal{Y}\}$. Suppose there exists a scoring function $s : \mathcal{H} \to [0, 1]$ such that the scoring function $s(|\psi\rangle)$ prioritizes data points with lower gradient variance at each epoch, and the pacing function p increases monotonically over epochs $t = 1$ to T. Then, this learning setup defined by s and p achieves a theoretical upper bound on its expected empirical risk after T epochs that is less than or equal to the bound for standard training (uniform access to the entire dataset).

The proof is given in Appendix B. This proposition establishes that by prioritizing data points with lower gradient variance, our learning setup achieves a tighter worst-case convergence guarantee. While this does not strictly prove that the expected risk will be lower in every scenario, it provides a strong theoretical motivation for why this data-ordering strategy is beneficial: It bounds the suboptimality more tightly, suggesting more stable and reliable training dynamics. The assumption of Lipschitz continuity for the gradients assumption is reasonable, given that the empirical risk $R(\boldsymbol{\vartheta})$ is generally bounded. The convexity assumption ensures a well-behaved loss landscape, which, while not always realistic, makes the analysis tractable. However, this assumption might be particularly reasonable in scenarios where this learning setup is combined with other warm-start techniques that initialize the parameters near to optimal values. By employing a scoring function that prioritizes such data points and a pacing function that gradually increases the complexity of the training set, the learning process can benefit from reduced variance in gradient estimates, resulting in faster convergence and lower expected risk after a fixed number of epochs.

While these propositions provide a theoretical intuition for the use of hard example mining and curriculum learning in quantum machine learning, empirical validation remains crucial. The theoretical benefits outlined may vary in practice due to factors such as the specific architecture of the quantum model, the characteristics of the data distribution, and the optimization algorithm used. Constructing meaningful scoring and pacing functions remains challenging, and empirical experiments are essential for assessing the practical effectiveness of this learning setup. Such validation can help identify discrepancies between expected and observed performance, allowing adjustments to optimize the approach for specific applications. Interestingly, although Propositions 1 and 2 lay the theoretical ground for what to expect, the numerical

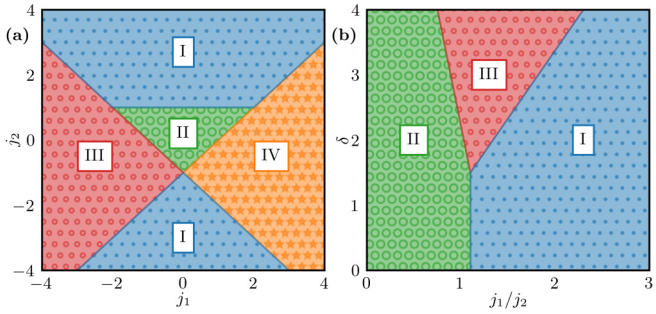

FIG. 2. Ground-state phase diagrams of (a) the generalized cluster Hamiltonian in Eq. (2) and (b) the bond-alternating *XXZ* Hamiltonian in Eq. (3), exhibiting (I) symmetry-protected topological, (II) trivial, (III) antiferromagnetic, and (IV) ferromagnetic phases.

performance in certain cases significantly exceeds these initial insights.

C. Learning setup

The methodology and underlying principles presented here are generally applicable to quantum machine learning tasks. For specificity, however, we focus on phase recognition. This section presents numerical results demonstrating the application of the presented framework to quantum phase recognition, a popular benchmark task in QML [45–50] due to its importance in the study of condensed matter physics [51,52]. We explore its effectiveness on two paradigmatic quantum spin chains: the generalized cluster model and the bond-alternating *XXZ* model, both well-studied quantum many-body Hamiltonians.

The generalized cluster model is described by the following local Hamiltonian:

$$H_{\text{cluster}} = \sum_{j=1}^{n} (Z_j - j_1 X_j X_{j+1} - j_2 X_{j-1} Z_j X_{j+1}), \qquad (2)$$

where n is the number of qubits, j_1 and j_2 are coupling strengths, and X_i and Z_i are Pauli operators acting on the ith qubit. As shown in Refs. [53,54] and illustrated in Fig. 2(a), the ground-state phase diagram exhibits four distinct regimes: (I) a symmetry-protected topological phase, (II) a trivial phase, (III) an antiferromagnetic phase, and (IV) a ferromagnetic phase. For specific choices of parameters, this Hamiltonian is the standard cluster model, the conventional cluster state being the ground state.

The bond-alternating *XXZ* Hamiltonian is slightly more complicated, although it remains geometrically local, and reads

$$H_{XXZ} = j_1 \sum_{j=1}^{n/2} (X_{2j-1}X_{2j} + Y_{2j-1}Y_{2j} + \delta Z_{2j-1}Z_{2j}) + j_2 \sum_{j=1}^{n/2-1} (X_{2j}X_{2j+1} + Y_{2j}Y_{2j+1} + \delta Z_{2j}Z_{2j+1}), \qquad (3)$$

where j_1 and j_2 are coupling strengths, X_i, Y_i, and Z_i are Pauli operators acting on the ith qubit, and δ is the spin anisotropy. As depicted in Fig. 2(b), this model Hamiltonian hosts three distinct phases that can be detected by the many-body topological invariant $\mathcal{Z}_{\mathcal{R}}$ [55]: (I) a topological phase, (II) a trivial phase, and (III) an antiferromagnetic phase.

Our learning tasks involve developing a classification model that can identify the quantum phase given the ground state of a system described by either the *generalized cluster Hamiltonian* (GCH) or the bond-alternating XXZ Hamiltonian. We achieve this by generating a training set, denoted by $S = \{(|\psi_i\rangle, \boldsymbol{y_i})\}_{i=1}^{N}$, where N represents the number of training data points. Each element in S is a pair: $|\psi_i\rangle$ corresponds to the ground-state vector of the Hamiltonian (H_{cluster} or H_{XXZ}) for a specific choice of coupling constants (j_1 and j_2 or j_1/j_2 and δ). The ground states are drawn from a distribution $\mathcal{D}$ that corresponds to sampling these coupling constants uniformly at random, ensuring a balanced representation of all classes. The second element, $\boldsymbol{y_i}$, represents the corresponding one-hot encoded phase label. In this scheme, each label is a vector with a length equal to the number of classes, with only one element set to 1, indicating the corresponding phase.

We leverage the *quantum convolutional neural network* (QCNN) architecture introduced in Ref. [56] for the phase classification tasks. Inspired by classical convolutional neural networks, QCNNs are expected to excel at identifying spatial or temporal patterns, making them well suited for this purpose. A key feature of QCNNs is their alternating structure of convolutional and pooling layers. Convolutional layers apply parametrized, translation-invariant unitary operations to neighboring qubits, acting like filters between feature maps across layers. Pooling layers then reduce the dimensionality of the quantum state while preserving the relevant information. This is achieved by measuring a subset of qubits and applying single-qubit unitaries based on the measurement outcomes. Each pooling layer consistently reduces the number of qubits by a constant factor, leading to quantum circuits with logarithmic depth compared to the initial system size.

We can describe the action of the QCNN as a quantum channel $\mathcal{C}_{\boldsymbol{\vartheta}}$ parametrized by $\boldsymbol{\vartheta}$. This channel transforms an input state ρ_{in} into an output state ρ_{out} via the mapping $\rho_{\text{out}} = \mathcal{C}_{\boldsymbol{\vartheta}}[\rho_{\text{in}}]$. Following this, a task-specific Hermitian operator is measured on the output state ρ_{out} to obtain an expectation value. In our numerical implementations, the QCNN maps an eight-qubit input state vector $|\psi\rangle$ into a two-qubit output state. The labeling function for the output state utilizes the probabilities obtained by measuring the state in the computational basis, giving rise to outcomes (0, 0), (0, 1), (1, 0), and (1, 1). Specifically, the predicted label $\hat{\boldsymbol{y}}$ corresponds to the vector of the probabilities as defined by

$$|\psi\rangle \mapsto (p_j)_{j\in\{0,1\}^2} \mapsto \hat{\boldsymbol{y}} := (p_{0,0}, p_{0,1}, p_{1,0}, p_{1,1}). \qquad (4)$$

For each experimental run, data are generated from the corresponding distribution $\mathcal{D}$. During training, we employ the mean squared error as loss function

$$\ell(\boldsymbol{\vartheta}; (|\psi\rangle, \boldsymbol{y})) := \sum_{j=1}^{M} (\langle j|(\mathcal{C}_{\boldsymbol{\vartheta}}[|\psi\rangle\langle\psi|])|j\rangle - y_j)^2, \qquad (5)$$

where M is the number of classes. Then, given a training set of size N drawn from the distribution $\mathcal{D}$, we sample a minibatch

of size L, and we minimize the empirical risk

$$\hat{R}_S(\boldsymbol{\vartheta}) = \frac{1}{L}\sum_{i=1}^{L} \ell(\boldsymbol{\vartheta}; (|\psi_i\rangle, \boldsymbol{y}_i)). \tag{6}$$

Let us detail here the experimental setup used throughout the following sections. We generate a training set S containing 50 ground states drawn from the distribution $\mathcal{D}$, and a test set T containing 1000 samples from the same distribution. While the ratio of training to test data is unconventional compared to classical machine learning practices, it is intentionally chosen for this quantum context. Because obtaining ground-state data for quantum many-body systems is resource intensive, demonstrating robust generalization from a minimal number of training examples is a key benchmark for QML models [45,48]. Furthermore, restricting the training set size ensures the learning task is sufficiently demanding to clearly isolate and highlight the performance benefits of our data-ordering strategies. The large test set is maintained to provide high statistical confidence in the evaluated generalization performance.

To ensure a balanced representation of all classes, these states are obtained by sampling the Hamiltonian's coupling constants an equal number of times from within each distinct phase region. We refer to the standard approach of uniformly sampling minibatches of size L from the training set S as the standard method. In this work, we use a fixed minibatch size of $L = 10$. Each learning framework is evaluated over ten independent runs, using newly generated data and random initializations for each run. Next, we define a scoring function $s : S \to \mathbb{R}$ to measure the difficulty of each training example $(|\psi_i\rangle, \boldsymbol{y}_i)$. This function assigns a higher score to harder examples, e.g., $s((|\psi_i\rangle, \boldsymbol{y}_i)) > s((|\psi_j\rangle, \boldsymbol{y}_i))$. The scoring function s incorporates prior knowledge and thus embodies some of the inductive bias of the model. We additionally define a pacing function to determine how the size of the subsets from which minibatches are uniformly sampled is adjusted throughout training. Details regarding the implementation and the specific pacing functions employed are provided in Ref. [57] and Appendix C.

For the quantum circuit simulations, we leverage the `PennyLane` [58] software library running on classical hardware. We utilize the Adam optimizer [59] for parameter updates during training, which is a stochastic gradient descent method that dynamically adjusts learning rates.

D. Self-taught learning

In this first experiment, we order the quantum data based on the loss of a pretrained QCNN model with identical architecture and trained without ordering. This is usually referred to as self-taught learning. At the beginning of the training, we order the training set according to the loss of each data point. We then gradually increase the number of data points used for training throughout the epochs, following a monotonically increasing pacing function. We consider four orderings: (1) *Standard*, which is used as a baseline and corresponds to the standard approach with no specific ordering nor pacing function, all data points are treated equally from the start; (2) *Random*, where data points are randomly ordered, and serves as a control that should behave similar to the Standard approach; (3) *Easy*, which prioritizes data points with lower loss values (easier examples); and (4) *Hard*, which prioritizes data points with higher loss values (harder examples).

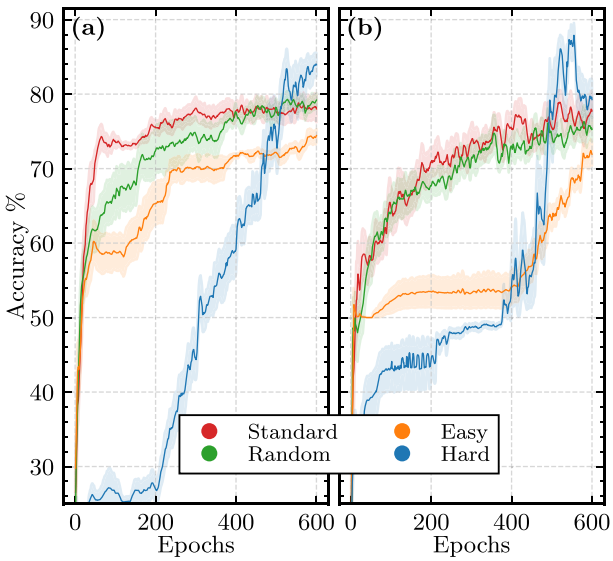

FIG. 3. Average accuracy on the test set as a function of the training epochs for the quantum phase recognition task for the (a) generalized cluster model and (b) the bond-alternating *XXZ* model, using a self-taught strategy. The shaded areas represent the standard error of the mean over ten runs with different random initialization and training data.

Figure 3 shows the average performance over ten runs of these strategies on the generalized cluster and the bond-alternating *XXZ* models. For both spin models, the Standard and Random strategies exhibit comparable performance, achieving the best accuracy nearing 80%. This similarity arises because Random introduces random ordering, mirroring the uniform treatment of data in the Standard approach. Interestingly, the Easy strategy, which prioritizes easier examples, performs worse than both Standard and Random, converging to a lower accuracy and suggesting that early exposure to easier examples might not be optimal. Conversely, the Hard strategy, which focuses on harder examples, despite initially slower convergence, ultimately yields the best performance, approaching 90% accuracy (see Table I). Notably,

TABLE I. Average best accuracy over ten runs for the quantum phase recognition task on the generalized cluster model and the bond-alternating *XXZ* model, using a self-taught strategy. The best-performing method prioritizes hard examples early in the training. Best performance is indicated in bold.

	Generalized cluster		Bond-alternating *XXZ*	
	Training (%)	Test (%)	Training (%)	Test (%)
Standard	79.4	78.2	81.2	78.6
Random	80.6	79.2	78.8	75.8
Easy	75.6	74.4	73.4	72.4
Hard	**86.6**	**83.6**	**88.0**	**86.5**

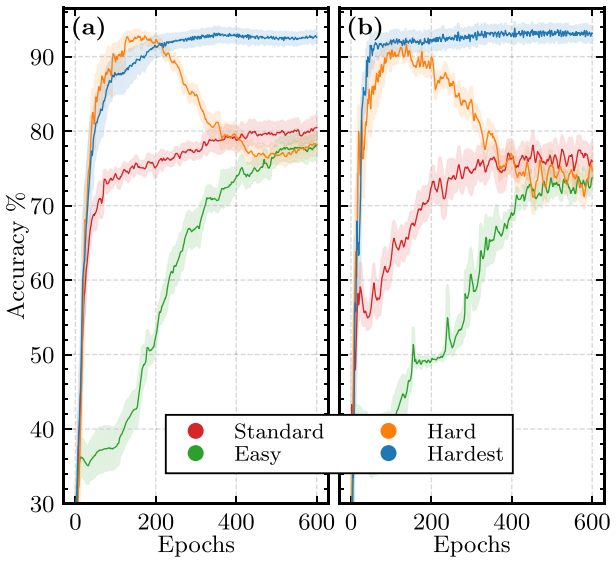

FIG. 4. Average accuracy on the test set as a function of the training epochs for the quantum phase recognition task for the (a) generalized cluster model and (b) the bond-alternating *XXZ* model, using a self-paced strategy. The shaded areas represent the standard error of the mean over ten runs with different random initialization and training data.

for the bond-alternating *XXZ* model shown in Fig. 3(b), the highest accuracy is reached before the end of the training, and then it decreases in the final optimization steps due to the incorporation of the easiest examples in the training batch. This suggests that exposure to noninformative data points might even be detrimental for the performance of the quantum model, and highlights the potential benefit of focusing on complex examples early in the training.

E. Self-paced learning

Self-paced learning dynamically prioritizes training data based on the performance of the current model, unlike the self-taught strategy which scores each point based on the loss with respect to the target hypothesis. In this approach, we rank the training set by the current loss of each data point at each epoch. We refer the reader to Appendix D for an extended discussion on the complexity and overhead of this strategy. We explore four strategies within this framework, using a monotonically increasing pacing function to gradually expose more data points as training progresses: (1) Standard, (2) Easy, and (3) Hard, which we discussed previously, and (4) Hardest, which employs a constant pacing function, consistently training on the ten most difficult examples at each epoch. This means that, although the pacing function is constant, the most difficult examples can vary at each epoch depending on the performance of the model.

The performance of these strategies is illustrated in Fig. 4, showing the average accuracy on the training and test sets for the quantum phase recognition task across the two spin models. Standard and Easy strategies achieve similar performance, with a final accuracy nearing 80%. This indicates that a straightforward or Easy approach does not significantly boost the model's learning capability. The Hard strategy initially shows promising performance but regresses as more data points are introduced due to the increasing pacing function. Despite this regression, it demonstrates the potential benefit of focusing on complex examples early in the training. In striking contrast, the Hardest strategy quickly converges to near-perfect accuracy on the training set and achieves over 90% accuracy on the test set (see Table II). By consistently training on the hardest examples, the model develops a robust understanding of the most challenging aspects of the training data, such as the boundaries of the different quantum phases—as we further explore in Sec. III H—hence significantly enhancing its performance.

TABLE II. Average best accuracy over ten runs for the quantum phase recognition task on the generalized cluster model and the bond-alternating *XXZ* model, using a self-paced strategy. The best-performing method prioritizes the hardest examples in each training epoch. Best performance is indicated in bold.

	Generalized cluster		Bond-alternating *XXZ*	
	Training (%)	Test (%)	Training (%)	Test (%)
Standard	83.2	77.4	82.2	77.4
Easy	80.8	78.1	76.4	73.5
Hard	98.6	**92.6**	95.0	91.6
Hardest	**99.8**	**92.6**	**97.2**	**93.5**

F. Physics-inspired learning

A key difference between classical and quantum machine learning is, once again, the presence of barren plateaus, where the variation of the loss function over the parameter landscape is exponentially suppressed with the system size [6]. At its core, this is a concentration of measure effect in high dimensions and can be captured in several different ways. One effective approach is through the *dynamical Lie algebra* (DLA) of the circuit ansatz [5,60]. The DLA $\mathfrak{g}$ is defined as the Lie closure over its set of generators,

$$\mathfrak{g} := \langle i\mathcal{G} \rangle_{\text{Lie}}, \tag{7}$$

where the generators are the generators of the parametrized gates of the circuit, $\mathcal{G} := \{H_1, H_2, \ldots, H_K\}$. The DLA $\mathfrak{g}$ is the vector space (over $\mathbb{R}$) that is obtained by taking the span over all the nested commutators that can be formed from the elements of $i\mathcal{G}$. Assuming that the quantum neural network approximates a unitary 2-design [61] over the unitary group generated by the DLA $\mathfrak{g}$, for the training state ρ, the variation of the loss is then given by

$$\text{Var}(\ell(\boldsymbol{\vartheta}; \rho, O)) = \frac{\mathcal{P}_{\mathfrak{g}}(\rho)\mathcal{P}_{\mathfrak{g}}(O)}{\dim(\mathfrak{g})}, \tag{8}$$

where $P_{\mathfrak{g}}$ denotes the $\mathfrak{g}$ purity, defined for Hermitian operators as

$$\mathcal{P}_{\mathfrak{g}}(H) := \sum_{l=1}^{\dim(\mathfrak{g})} |\text{Tr}(B_j^\dagger H)|^2. \tag{9}$$

Here, $\{B_j\}_{j=1}^{\dim(\mathfrak{g})}$ is a Hilbert-Schmidt orthonormal basis of the vector space $\mathfrak{g}_{\mathbb{C}} := \text{span}(\mathfrak{g})$. From Eq. (8), we observe

that barren plateaus occur if $1/\dim(\mathfrak{g}) = \mathcal{O}(b^{-n})$, $\mathcal{P}_{\mathfrak{g}}(O) = \mathcal{O}(b^{-n})$, or $\mathcal{P}_{\mathfrak{g}}(\rho) = \mathcal{O}(b^{-n})$ with $b > 1$.

The only term in Eq. (8) that depends on the training data is the $\mathfrak{g}$ purity given in Eq. (9). Favorable scaling of this quantity is a necessary condition to avoid an exponentially vanishing variance in Eq. (8) and, consequently, the onset of barren plateaus. Since one of the key statistical assumptions of barren plateaus is a random initialization of the trainable parameters, their appearance is most accurate at the beginning of training. Particularly during this initial training phase, one can expect that training data with higher $\mathfrak{g}$ purity enhance the trainability of the model, as these points are associated, on average, with larger gradients in the asymptotic regime. This, however, does not necessarily hold for finite system sizes, as barren plateaus are an intrinsically asymptotic concept. Building on this intuition, we propose the following domain-specific scoring function:

$$s(\rho) = 1 - \mathcal{P}_{\mathfrak{g}}(\rho), \tag{10}$$

where lower scores correspond to training examples with high $\mathfrak{g}$ purity, which are prioritized at the beginning of training. Notice that choosing s this way ensures that Proposition 1 holds. Accordingly, as training progresses and the model converges to more favorable regions of the cost landscape, the statistical assumptions underpinning barren plateaus become less relevant. As this occurs, we gradually incorporate more training data with higher scores, which correspond to lower $\mathfrak{g}$ purity. Ideally, this way, the model can still learn from the complete training set without low gradient data points impeding the training progress.

A potential limitation of using $\mathfrak{g}$ purity as a score function for larger system sizes is the computational difficulty associated with calculating $\mathcal{P}_{\mathfrak{g}}$, which generally scales exponentially with the number of qubits. To address this challenge, it is worthwhile to explore approximation schemes for computing $\mathcal{P}_{\mathfrak{g}}$, which could have broader implications beyond our specific context, particularly for understanding when quantum systems can be efficiently simulated using Lie-algebraic techniques [62]. To this end, we note that there are two sources of computational hardness. First, the Lie algebra itself might be exponentially large, requiring an exponentially sized basis of $\mathfrak{g}_{\mathbb{C}}$. In such cases, training would likely be infeasible anyway, as then $1/\dim(\mathfrak{g}) = \mathcal{O}(b^{-n})$. Second, even if the dimension of the Lie algebra grows only polynomially in the system size, that does not guarantee an efficient method for finding a basis for the space $\mathfrak{g}_{\mathbb{C}}$. Computing the basis elements $\{B_j\}_{j=1}^{\dim(\mathfrak{g})}$ requires computing nested commutators of the generators in $i\mathcal{G}$ and verifying the linear independence of the resulting operators (cf. Algorithm 1 in Ref. [63]). While we have an efficient classical description of the generating set, this does not ensure that the required operations (nested commutators and linear independence checks) can be performed efficiently. Instead, we may need to represent the generators as elements of $\mathbb{C}^{2^n}$ to carry out these computations, making the determination of even a single basis element B_j quickly infeasible. There are, however, special cases in which efficient procedures for computing $\{B_j\}_{j=1}^{\dim(\mathfrak{g})}$ are known [5,62].

To avoid computational bottlenecks in this section, we simplified the QCNN while preserving its structure to a parametrized matchgate circuit composed of fermionic Gaussian unitaries. Specifically, the circuit generators are $\mathcal{G} = \{Z_i\}_{i=1}^{8} \cup \{X_iX_{i+1}\}_{i=1}^{7}$, producing a DLA $\mathfrak{g}$ that remains computationally feasible. This simplification results in a QCNN with fewer variational parameters, leading to reduced performance across all strategies. Nonetheless, this trade-off is acceptable because our goal is not to optimize performance but rather to compare differences between frameworks (e.g., Standard versus alternative methods). We consider four strategies: (1) Standard and (2) Random, previously discussed, and (3) Higher $\mathcal{P}_{\mathfrak{g}}$, and (4) Lower $\mathcal{P}_{\mathfrak{g}}$, which prioritize higher and lower $\mathcal{P}_{\mathfrak{g}}$ states early in the training, respectively.

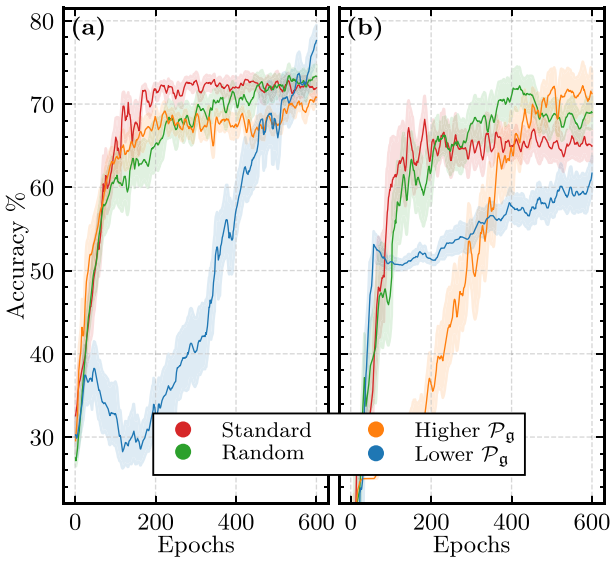

FIG. 5. Average accuracy on the test set as a function of the training epochs for the quantum phase recognition task for the (a) generalized cluster model and (b) the bond-alternating XXZ model, using a physics-inspired strategy ($P_{\mathfrak{g}}$ of the training quantum states). The shaded areas represent the standard error of the mean over ten runs with different random initialization and training data.

Figure 5 shows the accuracy performance of these strategies on the training and test sets for our quantum phase recognition tasks. We observe different behaviors for each spin model under study. For the generalized cluster model, the Standard and Random strategies perform similar, as expected. Interestingly, the Higher $\mathcal{P}_{\mathfrak{g}}$ also performs similarly, though with slightly worse results. Conversely, the Lower $\mathcal{P}_{\mathfrak{g}}$ strategy shows a gradual improvement initially, eventually achieving the highest performance on both test and training sets (see also Table III).

In contrast, for the bond-alternating XXZ model, while the Standard and Random strategies exhibit similar performance to those observed in the generalized cluster model, the $\mathcal{P}_{\mathfrak{g}}$-based strategies behave differently. In this case, the final accuracy of the Higher $\mathcal{P}_{\mathfrak{g}}$ strategy surpasses that of the Lower $\mathcal{P}_{\mathfrak{g}}$ strategy, which significantly underperforms compared to the other strategies. Notably, despite the Higher $\mathcal{P}_{\mathfrak{g}}$ strategy achieving the best final accuracy, the highest accuracy during the training is attained by the Random strategy, indicating

TABLE III. Average best accuracy over ten runs for the quantum phase recognition task on the generalized cluster model and the bond-alternating *XXZ* model, using a physics-inspired strategy. Best performance is indicated in bold.

	Generalized cluster		Bond-alternating *XXZ*	
	Training (%)	Test (%)	Training (%)	Test (%)
Standard	71.6	72.5	70.0	67.9
Random	72.8	72.9	**73.8**	**71.9**
Higher $\mathcal{P}_\mathfrak{g}$	68.6	70.9	73.4	71.6
Lower $\mathcal{P}_\mathfrak{g}$	**78.6**	**77.2**	62.4	60.8

that this specific implementation does not benefit from $\mathcal{P}_\mathfrak{g}$ ordering for this particular task.

These results suggest that ordering the quantum data according to $\mathcal{P}_\mathfrak{g}$ do not consistently yield major improvements in learning performance in this particular setup. While ordering based on $\mathcal{P}_\mathfrak{g}$ guarantees larger gradients on average at the early stages of the training, it does not necessarily lead to better learning outcomes. However, it is evident that $\mathcal{P}_\mathfrak{g}$ captures meaningful properties of the problem, as the performance of the model varies significantly with different orderings. Further research could be beneficial to understand the impact of $\mathcal{P}_\mathfrak{g}$ on training and to develop a scoring function that may leverage its properties in a nontrivial manner.

G. Large-scale implementations

To assess the scalability of our proposed framework, we extend our numerical experiments to larger system sizes. In this section, we investigate the performance of the most promising method identified in our earlier experiments (the self-paced learning strategy) on the generalized cluster model for systems of 16 and 32 qubits.

The results of these larger-scale simulations are presented in Fig. 6 and summarized in Table IV. For both the 16- and 32-qubit QCNN, the Hard strategy clearly demonstrates superior performance, reaching a higher test accuracy compared to both the Standard and Easy approaches. This is particularly remarkable in the 32-qubit case: While the absolute accuracy for all methods decreases, this drop is an expected consequence of the increased problem complexity coupled with the simplified, lower-parameter ansatz required to make the classical simulation of 32 qubits computationally feasible. Furthermore, within the restricted number of training epochs shown, this highlights a core advantage of our approach. In practical large-scale quantum machine learning, the computational budget for training steps is often strictly limited. The fact that the performance gap between the Hard strategy and the others remains (and even widens) under these strict constraints confirms that prioritizing difficult examples yields a more efficient optimization path. Thus, even if absolute performance is limited by the ansatz capacity and number of steps, the trainability benefits of data-ordering strategies persist.

Quantitatively, Table IV shows that the Hard strategy achieves a best test accuracy of 71.7% for 16 qubits and 58.7% for 32 qubits. This represents a performance improvement

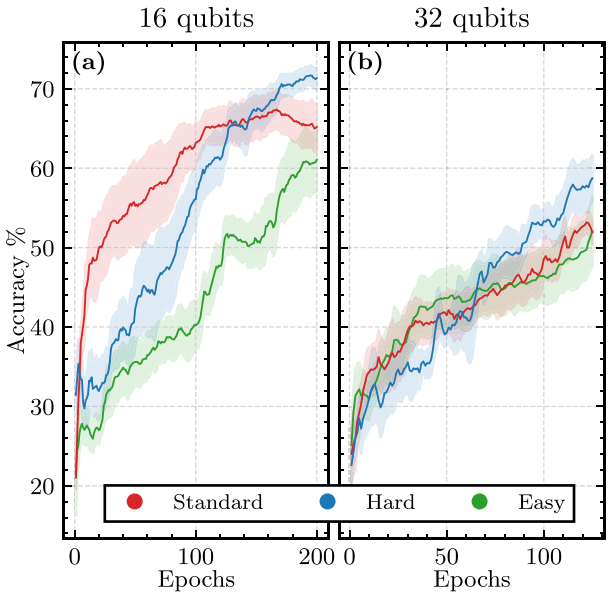

FIG. 6. Average accuracy on the test set as a function of the training epochs for the quantum phase recognition task for the generalized cluster model on (a) 16 qubits and (b) 32 qubits, using a self-paced strategy. The shaded areas represent the standard error of the mean over six runs with different random initialization and training data.

over the Standard approach of 4.3 percentage points for the 16-qubit case and 5.5 percentage points for the 32-qubit case. This widening gap suggests that the inductive bias introduced by focusing on challenging examples might be particularly relevant for navigating the more complex optimization landscapes of larger quantum systems. The Standard method, which samples uniformly, is more likely to be hindered by noninformative gradients, whereas the Hard strategy effectively steers the model toward learning the most decisive features of the data. These findings support the conclusion that a self-paced, difficulty-based curriculum is a scalable and effective approach for improving the performance of quantum machine learning models.

H. Accuracy at the cost of confidence

In this section, we try to understand further the differences between the Standard approach and the best-performing method within the proposed framework: the self-paced Hardest learning strategy. We analyze the probabilities for the

TABLE IV. Average best accuracy over six runs for the quantum phase recognition task on the generalized cluster model using a self-paced strategy, for 16- and 32-qubit QCNNs. Best performance is indicated in bold.

	16 qubits		32 qubits	
	Training (%)	Test (%)	Training (%)	Test (%)
Standard	71.7	67.4	58.7	53.2
Easy	67.0	61.1	58.3	52.1
Hard	**73.3**	**71.7**	**63.3**	**58.7**

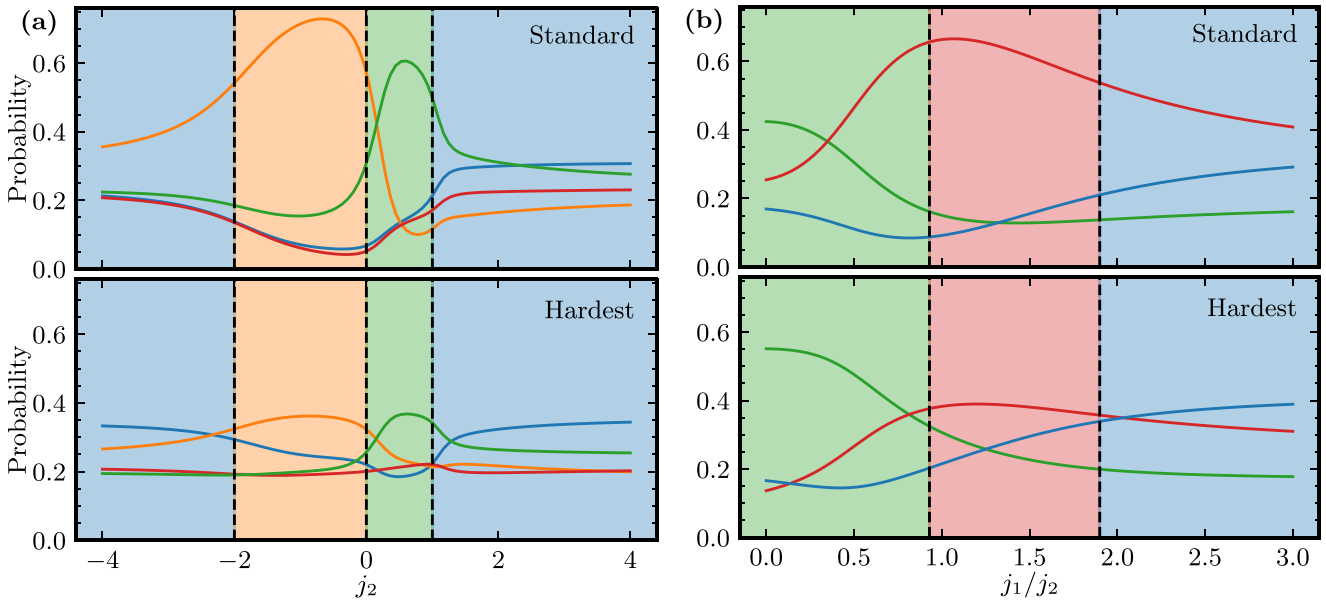

FIG. 7. (a) The generalized cluster Hamiltonian and (b) bond-alternating *XXZ* Hamiltonian for Standard (top) and Hardest (bottom) strategies. The background color of each panel indicates the true quantum phase of the system. A correct classification occurs when the highest probability color matches the background color of the panel. The self-paced Hardest strategy demonstrates superior accuracy both within each phase region and at the phase boundaries.

different phases as determined by the eight-qubit QCNN along a specific cut in the phase diagram in Fig. 2.

We begin by considering the generalized cluster Hamiltonian. We focus on a cut along the j_2 coupling constant, fixing $j_1 = 1$. The model undergoes three phase transitions: (1) from symmetry-protected topological to ferromagnetic at $j_2 = 2$, (2) from ferromagnetic to trivial at $j_2 = 0$, and (3) from trivial to symmetry-protected topological at $j_2 = 1$. Figure 7(a) illustrates the probabilities for the different phases under both the Standard (top) and self-paced Hardest (bottom) strategies. The phase diagram is accurately captured by the QCNN trained with the self-paced strategy, whereas the Standard approach fails to precisely detect the phase boundaries.

Next, we examine the bond-alternating *XXZ* Hamiltonian. Here, we closed-up cut along the j_1/j_2 coupling constants, fixing $\delta = 3$. The model features two phase transitions: (1) from trivial to symmetry-broken antiferromagnetic at $j_1/j_2 \approx 1$ and (2) from symmetry-broken antiferromagnetic to topological at $j_1/j_2 \approx 2$. Figure 7(b) shows the probabilities for the different phases under both the Standard (top) and self-paced Hardest (bottom) strategies. Once again, the QCNN trained with the self-paced strategy accurately captures the phase diagram, while the Standard approach misclassifies a large fraction of the ground states.

Overall, the Hardest approach outperforms the Standard strategy in terms of accuracy, both within individual phases and the phase boundaries. However, it is worth noting that the Hardest approach exhibits less confidence in its predictions, as indicated by the smaller differences between the probabilities of different classes. This suggests a potential trade-off between accuracy and prediction confidence. Despite its superior performance, we still observe slight deviations in the predicted transition points from the actual phase boundary. This discrepancy is expected due to the relatively large correlation length near the phase boundary, which exceeds what the capacity of the eight-qubit QCNN can capture accurately.

IV. CONCLUSIONS

In this work, we have introduced a training approach for quantum machine learning models that leverages the structure of the training data itself to introduce additional inductive bias, thereby improving learning capabilities. Drawing inspiration from classical methods like curriculum learning and hard example mining, we developed a framework tailored to quantum models. We provided some analytical insights and extensive numerical evidence that demonstrates how strategically ordering the quantum data can significantly improve the performance of quantum models.

Analytically, we have established that data ordering can mitigate the effects of barren plateaus (Proposition 1) and tighten the upper bound on the expected empirical risk to enhance convergence (Proposition 2). Our numerical experiments have focused on the task of quantum phase recognition in quantum spin chains, explicitly validated these theoretical insights. In particular, the accelerated convergence and improved final accuracy observed when using data-ordering strategies directly reflect the analytical benefits of reduced gradient variance outlined in Proposition 2. Among all the data-ordering strategies, we have observed that prioritizing harder data points early in the training process leads to superior performance compared to traditional training methods. The self-paced learning strategy, which dynamically adjusts data presentation based on current performance, has proven to be particularly effective, consistently achieving higher

accuracy than other strategies. Crucially, our large-scale simulations on systems of up to 32 qubits have confirmed that this performance advantage is maintained as system size increases. This emphasizes the potential of data ordering in quantum models, not just as a theoretical concept but as a relevant factor in practical applications.

While the framework presented here showcases interesting results, it also opens up several avenues for future research. One possible direction is to extend these methods to tasks beyond quantum phase recognition, such as unitary learning [45,64–67] or quantum error correction [56,68]. Furthermore, exploring different complexity measures—such as those based on notions of entanglement—or combining multiple complexity metrics, including those used in this work, could help refine the scoring and pacing functions. Integrating this data-centric approach with other warm-start techniques, such as those focused on parameter initialization, could further enhance model performance. On a higher level, this work further reinforces the importance of data-centric inductive biases in quantum machine learning models, suggesting that advancing this concept could lead to more sophisticated quantum circuit design strategies that can optimize performance from the outset, while also underscoring the value of leveraging established classical machine learning methods to foster meaningful progress in quantum machine learning.

ACKNOWLEDGMENTS

The authors would like to thank Elies Gil-Fuster and Adrián Pérez-Salinas for useful feedback on a previous version of this article. E.A. acknowledges funding from the Government of Spain (Severo Ochoa CEX2019-000910-S, FUNQIP, and European Union NextGenerationEU PRTR-C17.I1), Fundació Cellex, Fundació Mir-Puig, Generalitat de Catalunya (CERCA program), and European Union (PASQuanS2.1, 101113690). E.A. is a fellow of Eurecat's "Vicente López" Ph.D. grant program. F.J.S., J.E., and C.B.-P. were supported by the BMFTR (MUNIQC-Atoms, Hybrid++, QoSol, PasQuops), the BMWK (EniQma), the Munich Quantum Valley (K-8), the Quantum Flagship (PasQuans2, Millenion), the Einstein Foundation (Einstein Research Unit on Quantum Devices), Berlin Quantum, and the DFG (CRC 183, SPP 2514).

DATA AVAILABILITY

The data that support the findings of this article are openly available [57].

APPENDIX A: PROOF OF PROPOSITION 1

In this section, we restate and prove Proposition 1.

Proposition 1 (Mitigation of barren plateaus). Let $\{|\psi_i\rangle \in \mathcal{H}\}$ be a collection of quantum state vectors. Let $f_{\boldsymbol{\vartheta}} : \mathcal{H} \to \mathcal{Y}$ be a parametrized quantum model with parameters $\boldsymbol{\vartheta} \in \Theta$ and $\mathcal{Y} \subseteq \mathbb{R}$ the label space. Consider a loss function $\ell(\boldsymbol{\vartheta}; (|\psi\rangle, y))$ associated with data points $\{(|\psi_i\rangle, y_i) \in \mathcal{H} \times \mathcal{Y}\}$. Suppose there exists a scoring function $s : \mathcal{H} \to [0, 1]$ such that the expected squared norm of the gradient

$$G(z) = \mathbb{E}[\|\nabla_{\boldsymbol{\vartheta}} \ell(\boldsymbol{\vartheta}; (|\psi\rangle, y)\|^2 \mid s(\psi) = z] \tag{A1}$$

is a nonincreasing function of s. Then, this learning setup defined by s increases the expected squared gradient magnitude during early training epochs compared to standard training (random data order).

Proof. Let F be the cumulative distribution function of the scores over the data set given by

$$F(z) = \mathbb{P}[s(\psi_i) \leqslant z]. \tag{A2}$$

The learner has access to the subset $\mathcal{D}_t = \{(|\psi_i\rangle, y_i) \mid s(\psi_i) \leqslant z_t\}$. The expected gradient magnitude at epoch t is defined as

$$\mathbb{E}_{\mathcal{D}_t}[\|g\|^2] = \frac{1}{F(z_t)} \int_0^{z_t} G(z) dF(z), \tag{A3}$$

where g is the gradient of the loss function with respect to the parameters $\boldsymbol{\vartheta}$. For random data presentation, all data points are equally likely to be sampled, regardless of their scores. Therefore,

$$\mathbb{E}_{\text{rand}}[\|g\|^2] = \int_0^1 G(z) dF(z). \tag{A4}$$

Since G is nonincreasing, for $z \leqslant z_t$, $G(z) \geqslant G(z_t)$, and similarly, for $z \geqslant z_t$, $G(z) \leqslant G(z_t)$. Therefore, we can lower bound

$$\mathbb{E}_{\mathcal{D}_t}[\|g\|^2] \geqslant \frac{1}{F(z_t)} G(z_t) \int_0^{z_t} dF(z) = G(z_t) \tag{A5}$$

and upper bound

$$\mathbb{E}_{\text{rand}}[\|g\|^2] \leqslant G(z_t) F(z_t) + G(z_t)(1 - F(z_t)) = G(z_t), \tag{A6}$$

which leads to

$$\mathbb{E}_{\mathcal{D}_t}[\|g\|^2] \geqslant \mathbb{E}_{\text{rand}}[\|g\|^2], \tag{A7}$$

which proves the statement to be shown. ■

APPENDIX B: PROOF OF PROPOSITION 2

In this section, we restate and prove Proposition 2.

Proposition 2 (Faster convergence). Let $\{|\psi_i\rangle\} \in \mathcal{H}$ be a collection of quantum state vectors. Let $f_{\boldsymbol{\vartheta}} : \mathcal{H} \to \mathcal{Y}$ be a parametrized quantum model with parameters $\boldsymbol{\vartheta} \in \Theta$ and $\mathcal{Y} \subseteq \mathbb{R}$ the label space. Consider a convex loss function with Lipschitz continuous gradients $\ell(\boldsymbol{\vartheta}; (|\psi\rangle, y))$ associated with data points $\{(|\psi_i\rangle, y_i) \in \mathcal{H} \times \mathcal{Y}\}$. Suppose there exists a scoring function $s : \mathcal{H} \to [0, 1]$ such that the scoring function $s(|\psi\rangle)$ prioritizes data points with lower gradient variance at each epoch, and the pacing function p increases monotonically over epochs $t = 1$ to T. Then, this learning setup defined by s and p achieves a theoretical upper bound on its expected empirical risk after T epochs that is less than or equal to the bound for standard training (uniform access to the entire dataset).

Proof. In standard training, data points are selected uniformly at random. The variance of the stochastic gradient is

$$\sigma^2_{\text{rand}} = \mathbb{E}_{(|\psi\rangle, y) \sim D}\left[\left\|\nabla \ell\left(f_{\boldsymbol{\vartheta}_t}; (|\psi\rangle, y)\right) - \nabla R(\boldsymbol{\vartheta}_t)\right\|^2\right], \tag{B1}$$

where D is the uniform distribution over the entire dataset. In contrast, for the learning setup defined by s and p, the variance of the stochastic gradient is

$$\sigma^2_{s,p} = \mathbb{E}_{(|\psi\rangle, y) \sim D_t}\left[\left\|\nabla \ell\left(f_{\boldsymbol{\vartheta}_t}; (|\psi\rangle, y)\right) - \nabla R(\boldsymbol{\vartheta}_t)\right\|^2\right], \tag{B2}$$

TABLE V. Summary of the QCNN architecture and parameters for each numerical experiment.

Experiment	Qubits	Two-qubit gate	Gate params.	Layers	Total params.
Self-taught	8	General gates	15	3	75
Self-paced	8	General gates	15	3	75
Physics insp.	8	Match gates	5	3	25
Large-scale	16	Simplified	8	4	61
Large-scale	32	Simplified	8	5	77

where

$$\mathcal{D}_t = \{(|\psi_i\rangle, y_i) \mid s(\psi_i) \leqslant z_t\}. \tag{B3}$$

For convex functions with Lipschitz continuous gradients, the expected suboptimality after T epochs is bounded by [69]

$$\mathbb{E}[R(\boldsymbol{\vartheta}_T) - R(\boldsymbol{\vartheta}^*)] \leqslant \frac{\|\boldsymbol{\vartheta}_0 - \boldsymbol{\vartheta}^*\|^2}{2\eta T} + \frac{\eta}{2T}\sum_{t=1}^{T}\sigma^2(t), \tag{B4}$$

where $\boldsymbol{\vartheta}_0$ are the initial parameters, $\boldsymbol{\vartheta}^*$ are the optimal parameters minimizing $R(\boldsymbol{\vartheta})$, and η is the learning rate (which we assume to be constant for simplicity). Then, for the standard training, we have

$$\mathbb{E}\left[R\left(\boldsymbol{\vartheta}_T^{\text{rand}}\right) - R(\boldsymbol{\vartheta}^*)\right] \leqslant \frac{\|\boldsymbol{\vartheta}_0 - \boldsymbol{\vartheta}^*\|^2}{2\eta T} + \frac{\eta}{2T}\sum_{t=1}^{T}\sigma_{\text{rand}}^2(t), \tag{B5}$$

and for the setup defined by s and t

$$\mathbb{E}\left[R\left(\boldsymbol{\vartheta}_T^{s,p}\right) - R(\boldsymbol{\vartheta}^*)\right] \leqslant \frac{\|\boldsymbol{\vartheta}_0 - \boldsymbol{\vartheta}^*\|^2}{2\eta T} + \frac{\eta}{2T}\sum_{t=1}^{T}\sigma_{s,p}^2(t). \tag{B6}$$

Given that for all t, $\sigma_{s,p}^2(t) \leqslant \sigma_{\text{rand}}^2(t)$, it follows that

$$\sum_{t=1}^{T}\sigma_{s,p}^2(t) \leqslant \sum_{t=1}^{T}\sigma_{\text{rand}}^2(t). \tag{B7}$$

Let K_{rand} and $K_{s,p}$ denote the right-hand sides of the inequalities in Eqs. (B5) and (B6), respectively. These are the upper bounds on the expected suboptimality for our methods and standard training. From Eq. (B7), it is clear that the second term in $K_{s,p}$ is less than or equal to the second term in K_{rand}. Therefore, we can conclude that the overall upper bound for our method is tighter:

$$K_{s,p} \leqslant K_{\text{rand}}. \tag{B8}$$

This proves that our learning setup achieves an equal or tighter theoretical upper bound on the expected empirical risk compared to standard training. ■

APPENDIX C: ADDITIONAL NUMERICAL DETAILS

This Appendix provides a detailed description of the numerical methods, model architecture, and hyperparameters used for the quantum phase classification experiments on the GCH and the bond-alternating *XXZ* Hamiltonian (*XXZ*).

1. Dataset generation

The training and test datasets were generated using a balanced sampling procedure to ensure an equal representation of each quantum phase. For each model, we sample parameter points uniformly at random from the regions of the two-dimensional phase diagram (see Fig. 2) corresponding to each distinct phase. The training set was constructed from 50 total parameter points, resulting in 12–13 samples per phase for the GCH (four phases) and 16–17 samples per phase for the *XXZ* model (three phases). The test set was generated using the same balanced distribution but comprised a larger set of 1000 samples (250 for the large-scale simulations) to ensure a robust evaluation. The final datasets consist of the ground states corresponding to these parameter points.

2. Quantum circuit architecture

We employed a QCNN architecture with periodic boundary conditions. The circuit consists of alternating convolutional and pooling layers, where all gates within a given layer share the same trainable parameters. The final two qubits are measured in the computational basis to produce a probability vector over the four outcomes {00, 01, 10, 11}, which serves as the model's output for classification.

The specific implementation of the QCNN, including the number of qubits, layers, and the type of variational gates, was adapted for each set of experiments. A comprehensive summary of these configurations is provided in Table V. For most experiments, the convolutional and pooling layers were constructed from general two-qubit unitary gates. However, for the physics-inspired learning experiments (Sec. III F), the computational cost of calculating the $\mathfrak{g}$ purity necessitated the use of a more restricted ansatz based on matchgate circuits. Similarly, for the large-scale simulations, we employed a simplified gate ansatz with fewer parameters due to computational constraints. In particular, we utilized the matrix product state backend from `TensorCircuit` [70] to simulate the large-scale quantum circuits. A bond dimension of $\chi = 20$ was employed for the simulations of 16- and 32-qubit QCNNs.

3. Training hyperparameters

The model parameters were optimized using the Adam algorithm with a learning rate of 0.01 and hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The parameters were initialized by sampling from a Gaussian distribution with a mean of zero and a standard deviation of $1/\sqrt{n}$, where n is the number of qubits. All reported results are averaged over ten independent runs (six for the large-scale simulations), each with a different random data sample and parameter initialization, to ensure statistical robustness.

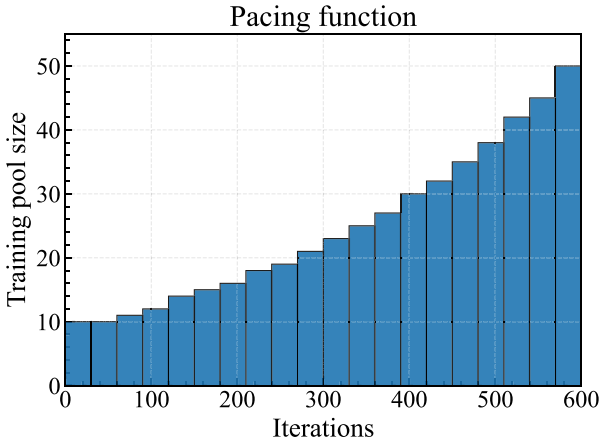

FIG. 8. The exponential pacing function used for a 600 iterations training run. The function dictates the size of the data pool from which minibatches are sampled at each epoch.

4. Curriculum learning implementation

The curriculum is defined by a pacing function, which controls the size of the data pool accessible to the learner at each training epoch. We experimented with linear, logarithmic, and exponential pacing functions. The exponential function, illustrated in Fig. 8 as an example, consistently yielded the best performance and was therefore used for all reported curriculum learning results.

The curriculum starts with an accessible pool of ten samples, which grows exponentially to include all 50 training samples by the end of the training. It is important to note that each epoch, a minibatch of ten samples was drawn uniformly at random from the currently accessible data pool, not from the entire dataset.

APPENDIX D: COMPLEXITY ANALYSIS AND OVERHEAD OF THE SELF-PACED STRATEGY

The self-paced learning strategy requires continuously rescoring and reordering the training data based on the model's evolving hypothesis. While dynamically evaluating the loss over the entire dataset introduces a computational overhead, this cost must be directly contextualized against the dominant resource requirements of gradient estimation when executing on physical quantum hardware.

For a dataset of size N, scoring the data requires N forward passes through the quantum circuit. Subsequently, sorting the scores incurs a classical computational cost of $\mathcal{O}(N \log_2 N)$. In contrast, the optimization process on quantum hardware relies heavily on the parameter-shift rule for exact gradient calculation [71,72]. For a parametrized quantum circuit with P trainable parameters and a minibatch of size L, computing the gradient for a single optimization step requires $\mathcal{O}(P \times L)$ circuit executions.

In our numerical implementations, rescoring was performed at every optimization step. Even under this frequent update schedule, the $\mathcal{O}(P \times L)$ cost of gradient estimation vastly dominates the N forward passes required for scoring, as P is typically large and grows as models scale to achieve better expressivity. Furthermore, our framework is designed to support periodic rescoring to optimize performance. By updating scores only every few iterations, the additional circuit execution budget can be reduced to a negligible fraction of the total training cost.

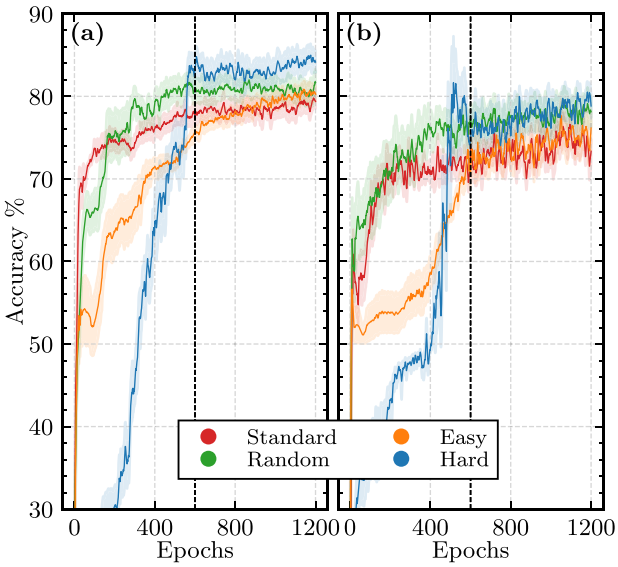

FIG. 9. Average accuracy on the test set as a function of the training epochs (extended to 1200 epochs) for the quantum phase recognition task for the (a) generalized cluster model and (b) the bond-alternating *XXZ* model, using a self-taught strategy. The vertical dashed line at epoch 600 indicates the completion of the pacing function schedule, after which the model is trained on the full dataset. The shaded areas represent the standard error of the mean over ten runs with different random initialization and training data.

Finally, the classical sorting overhead of $\mathcal{O}(N \log_2 N)$ remains negligible compared to the time required for quantum circuit executions. Therefore, the overhead introduced by the self-paced strategy represents only a small fraction of the total execution budget. By significantly improving the trainability and convergence rate of the model within a fixed number of steps, this method effectively amortizes its own scoring cost, proving to be a highly resource-efficient strategy for near-term quantum devices.

APPENDIX E: EXTENDED NUMBER OF ITERATIONS IN THE SELF-TAUGHT STRATEGY

To observe the asymptotic behavior of the self-taught methods, we extended the training to 1200 epochs for both the generalized cluster and the bond-alternating *XXZ* models on ten different independent runs. Figure 9 illustrates the test accuracy over this extended training window. The vertical dashed line at epoch 600 marks the end of the pacing function schedule.

It is important to emphasize that the core objective of these data-ordering strategies is not necessarily to reach asymptotic saturation over an arbitrarily large number of epochs. Rather, the primary goal is to improve model performance and accelerate convergence within a fixed, and often strictly limited, number of training steps. This is particularly critical in the context of near-term quantum hardware, where circuit

executions are expensive and optimization budgets are highly constrained. Nevertheless, analyzing the performance after the completion of the pacing function schedules provides valuable insights into the stability of the optimization trajectory.

As shown in Fig. 9(a) for the generalized cluster model, the Hard strategy successfully steers the optimization process away from poor local minima and toward a more favorable region of the parameter space early in the training. Once the model is situated within this region, subsequent training on the complete dataset preserves this initial advantage, resulting in a higher stabilized final accuracy compared to the other orderings.

For the bond-alternating *XXZ* model in Fig. 9(b), the extended training clarifies the sharp accuracy drop observed near epoch 600. This decrease in the final optimization steps directly coincides with the introduction of the easiest, least informative examples into the training batches. Intriguingly, this suggests that in certain quantum machine learning scenarios, exposure to noninformative data points might actively degrade the model's performance, and it might be beneficial to permanently remove the least informative samples to improve overall accuracy. Following this transient drop, the Hard strategy stabilizes alongside the other methods. Table VI summarizes the best test and training accuracies for all methods evaluated.

TABLE VI. Average best accuracy over ten runs for the quantum phase recognition task on the generalized cluster model and the bond-alternating *XXZ* model, using a self-taught strategy (extended to 1200 epochs). The best-performing method prioritizes hard examples early in the training. Best performance is indicated in bold.

	Generalized cluster		Bond-alternating *XXZ*	
	Training (%)	Test (%)	Training (%)	Test (%)
Standard	79.2	76.6	79.0	76.6
Random	82.8	81.9	81.4	79.5
Easy	81.8	80.8	78.4	77.7
Hard	**85.9**	**85.0**	**82.1**	**81.7**

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

[2] Y. Liu, S. Arunachalam, and K. Temme, A rigorous and robust quantum speed-up in supervised machine learning, Nat. Phys. **17**, 1013 (2021).

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

[4] N. Pirnay, R. Sweke, J. Eisert, and J.-P. Seifert, Superpolynomial quantum-classical separation for density modeling, Phys. Rev. A **107**, 042416 (2023).

[5] M. Ragone, B. N. Bakalov, F. Sauvage, A. F. Kemper, C. Ortiz Marrero, M. Larocca, and M. Cerezo, A Lie algebraic theory of barren plateaus for deep parameterized quantum circuits, Nat. Commun. **15**, 7172 (2024).

[6] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, Barren plateaus in quantum neural network training landscapes, Nat. Commun. **9**, 4812 (2018).

[7] M. Larocca, S. Thanasilp, S. Wang, K. Sharma, J. Biamonte, P. J. Coles, L. Cincio, J. R. McClean, Z. Holmes, and M. Cerezo, A review of barren plateaus in variational quantum computing, Nat. Rev. Phys. **7**, 174 (2024).

[8] E. R. Anschuetz and B. T. Kiani, Quantum variational algorithms are swamped with traps, Nat. Commun. **13**, 7760 (2022).

[9] A. A. Mele, A. Angrisani, S. Ghosh, S. Khatri, J. Eisert, D. S. França, and Y. Quek, Noise-induced shallow circuits and absence of barren plateaus, Nat. Phys. **22**, 751 (2026).

[10] N. A. Nemkov, E. O. Kiktenko, and A. K. Fedorov, Barren plateaus are swamped with traps, Phys. Rev. A **111**, 012441 (2025).

[11] S. Thanasilp, S. Wang, N. A. Nghiem, P. Coles, and M. Cerezo, Subtleties in the trainability of quantum machine learning models, Quantum Mach. Intell. **5**, 21 (2023).

[12] J. M. Kübler, S. Buchholz, and B. Schölkopf, The inductive bias of quantum kernels, in *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS'21 (Curran Associates Inc., Red Hook, NY, 2021), pp. 12661–12673.

[13] M. Cerezo, G. Verdon, H.-Y. Huang, L. Cincio, and P. J. Coles, Challenges and opportunities in quantum machine learning, Nat. Comput. Sci. **2**, 567 (2022).

[14] J. Bowles, V. J. Wright, M. Farkas, N. Killoran, and M. Schuld, Contextuality and inductive bias in quantum machine learning, arXiv:2302.01365.

[15] M. Larocca, F. Sauvage, F. M. Sbahi, G. Verdon, P. J. Coles, and M. Cerezo, Group-invariant quantum machine learning, PRX Quantum **3**, 030341 (2022).

[16] J. J. Meyer, M. Mularski, E. Gil-Fuster, A. A. Mele, F. Arzani, A. Wilms, and J. Eisert, Exploiting symmetry in variational quantum machine learning, PRX Quantum **4**, 010328 (2023).

[17] J. Bowles, D. Wierichs, and C.-Y. Park, Backpropagation scaling in parameterised quantum circuits, Quantum **9**, 1873 (2023).

[18] C.-Y. Park and N. Killoran, Hamiltonian variational ansatz without barren plateaus, Quantum **8**, 1239 (2024).

[19] S. Jerbi, C. Gyurik, S. C. Marshall, R. Molteni, and V. Dunjko, Shadows of quantum machine learning, Nat. Commun. **15**, 5676 (2024).

[20] L. Schatzki, M. Larocca, Q. T. Nguyen, F. Sauvage, and M. Cerezo, Theoretical guarantees for permutation-equivariant quantum neural networks, npj Quantum Inf. **10**, 12 (2024).

[21] P. Rodriguez-Grasa, Y. Ban, and M. Sanz, Training embedding quantum kernels with data re-uploading quantum neural networks, Phys. Rev. Res. **7**, 023269 (2025).

[22] K. Zhang, L. Liu, M.-H. Hsieh, and D. Tao, Escaping from the barren plateau via Gaussian initializations in deep variational quantum circuits, Adv. Neural Inf. Process. Syst. **35**, 18612 (2022).

[23] H. R. Grimsley, G. S. Barron, E. Barnes, S. E. Economou, and N. J. Mayhall, Adaptive, problem-tailored variational quantum eigensolver mitigates rough parameter landscapes and barren plateaus, npj Quantum Inf. **9**, 19 (2023).

[24] J. Dborin, F. Barratt, V. Wimalaweera, L. Wright, and A. G. Green, Matrix product state pre-training for quantum machine learning, Quantum Sci. Technol. **7**, 035014 (2022).

[25] M. S. Rudolph, J. Miller, D. Motlagh, J. Chen, A. Acharya, and A. Perdomo-Ortiz, Synergistic pretraining of parametrized quantum circuits via tensor networks, Nat. Commun. **14**, 8367 (2023).

[26] A. A. Mele, G. B. Mbeng, G. E. Santoro, M. Collura, and P. Torta, Avoiding barren plateaus via transferability of smooth solutions in a Hamiltonian variational ansatz, Phys. Rev. A **106**, L060401 (2022).

[27] R. Puig, M. Drudis, S. Thanasilp, and Z. Holmes, Variational quantum simulation: A case study for understanding warm starts, PRX Quantum **6**, 010317 (2025).

[28] Y. Wang, B. Qi, C. Ferrie, and D. Dong, Trainability enhancement of parameterized quantum circuits via reduced-domain parameter initialization, Phys. Rev. Appl. **22**, 054005 (2024).

[29] C.-Y. Park, M. Kang, and J. Huh, Hardware-efficient ansatz without barren plateaus in any depth, arXiv:2403.04844.

[30] X. Shi and Y. Shang, Avoiding barren plateaus via Gaussian mixture model, New J. Phys. **27**, 104501 (2025).

[31] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, Curriculum learning, in *Proceedings of the 26th Annual International Conference on Machine Learning* (Association for Computing Machinery, Montreal, Quebec, Canada, 2009), pp. 41–48.

[32] Q. H. Tran, Y. Endo, and H. Oshima, Quantum curriculum learning, Phys. Rev. A **112**, 032431 (2025).

[33] A. Shrivastava, A. Gupta, and R. Girshick, Training region-based object detectors with online hard example mining, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, Piscataway, NJ, 2016), pp. 761–769.

[34] X. Wang, Y. Chen, and W. Zhu, A survey on curriculum learning, IEEE Trans. Pattern Anal. Mach. Intell. **44**, 4555 (2021).

[35] P. Soviany, R. T. Ionescu, P. Rota, and N. Sebe, Curriculum learning: A survey, Int. J. Comp. Vis. **130**, 1526 (2022).

[36] R. Tudor Ionescu, B. Alexe, M. Leordeanu, M. Popescu, D. P. Papadopoulos, and V. Ferrari, How hard can it be? Estimating the difficulty of visual search in an image, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, Piscataway, NJ, 2016), pp. 2157–2166.

[37] S. Guo, W. Huang, H. Zhang, C. Zhuang, D. Dong, M. R. Scott, and D. Huang, CurriculumNet: Weakly supervised learning from large-scale web images, in *Proceedings of the European Conference on Computer Vision (ECCV)*, edited by V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Lecture Notes in Computer Science, Vol. 11214 (Springer, Cham, 2018), pp. 139–154.

[38] L. Jiang, D. Meng, T. Mitamura, and A. G. Hauptmann, Easy samples first: Self-paced reranking for zero-example multimedia search, in *Proceedings of the 22nd ACM international conference on Multimedia* (Association for Computing Machinery, Orlando, FL, 2014), pp. 547–556.

[39] E. A. Platanios, O. Stretcu, G. Neubig, B. Poczos, and T. Mitchell, Competence-based curriculum learning for neural machine translation, in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (Association for Computational Linguistics, 2019), pp. 1162–1172.

[40] T. Matiisen, A. Oliver, T. Cohen, and J. Schulman, Teacher–student curriculum learning, IEEE Trans. Neural Netw. Learning Syst. **31**, 3732 (2019).

[41] X. Zhang, G. Kumar, H. Khayrallah, K. Murray, J. Gwinnup, M. J. Martindale, P. McNamee, K. Duh, and M. Carpuat, An empirical exploration of curriculum learning for neural machine translation, arXiv:1811.00739.

[42] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, and J. I. Latorre, Data re-uploading for a universal quantum classifier, Quantum **4**, 226 (2020).

[43] 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).

[44] G. Penha and C. Hauff, Curriculum learning strategies for IR: An empirical study on conversation response ranking, in *Advances in Information Retrieval* (Springer-Verlag, Lisbon, Portugal, 2020), pp. 699–713.

[45] 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).

[46] Y. Wu, B. Wu, J. Wang, and X. Yuan, Quantum phase recognition via quantum kernel methods, Quantum **7**, 981 (2023).

[47] Y.-J. Liu, A. Smith, M. Knap, and F. Pollmann, Model-independent learning of quantum phases of matter with quantum convolutional neural networks, Phys. Rev. Lett. **130**, 220603 (2023).

[48] E. Gil-Fuster, J. Eisert, and C. Bravo-Prieto, Understanding quantum machine learning also requires rethinking generalization, Nat. Commun. **15**, 2277 (2024).

[49] C. Umeano, A. E. Paine, V. E. Elfving, and O. Kyriienko, What can we learn from quantum convolutional neural networks? Adv. Quantum Tech. **8**, 2400325 (2025).

[50] P. Zapletal, N. A. McMahon, and M. J. Hartmann, Error-tolerant quantum convolutional neural networks for symmetry-protected topological phases, Phys. Rev. Res. **6**, 033111 (2024).

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

[52] S. Sachdev, *Quantum Phases of Matter* (Cambridge University Press, Cambridge, MA, 2023).

[53] R. Verresen, R. Moessner, and F. Pollmann, One-dimensional symmetry protected topological phases and their transitions, Phys. Rev. B **96**, 165124 (2017).

[54] N. Schuch, D. Pérez-García, and I. Cirac, Classifying quantum phases using matrix product states and projected entangled pair states, Phys. Rev. B **84**, 165139 (2011).

[55] A. Elben, J. Yu, G. Zhu, M. Hafezi, F. Pollmann, P. Zoller, and B. Vermersch, Many-body topological invariants from randomized measurements in synthetic quantum matter, Sci. Adv. **6**, eaaz3666 (2020).

[56] I. Cong, S. Choi, and M. D. Lukin, Quantum convolutional neural networks, Nat. Phys. **15**, 1273 (2019).

[57] E. Recio-Armengol and C. Bravo-Prieto, Learning complexity gradually in QML models repository, 2025, https://github.com/erik-armengol/learning-complexity-gradually

[58] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, S. Ahmed, V. Ajith, M. S. Alam, G. Alonso-Linaje, B. AkashNarayanan, A. Asadi, *et al.*, Pennylane: Automatic differentiation of hybrid quantum-classical computations, arXiv:1811.04968.

[59] D. P. Kingma, Adam: A method for stochastic optimization, arXiv:1412.6980.

[60] G. Aguilar, S. Cichy, J. Eisert, and L. Bittel, Full classification of Pauli Lie algebras, arXiv:2408.00081.

[61] D. Gross, K. M. R. Audenaert, and J. Eisert, Evenly distributed unitaries: On the structure of unitary designs, J. Math. Phys. **48**, 052104 (2007).

[62] M. L. Goh, M. Larocca, L. Cincio, M. Cerezo, and F. Sauvage, Lie-algebraic classical simulations for variational quantum computing, Phys. Rev. Res. **7**, 033266 (2025).

[63] M. Larocca, P. Czarnik, K. Sharma, G. Muraleedharan, P. J. Coles, and M. Cerezo, Diagnosing barren plateaus with tools from quantum optimal control, Quantum **6**, 824 (2022).

[64] L. Cincio, Y. Subaşı, A. T. Sornborger, and P. J. Coles, Learning the quantum algorithm for state overlap, New J. Phys. **20**, 113022 (2018).

[65] K. Beer, D. Bondarenko, T. Farrelly, T. J. Osborne, R. Salzmann, D. Scheiermann, and R. Wolf, Training deep quantum neural networks, Nat. Commun. **11**, 808 (2020).

[66] L. Cincio, K. Rudinger, M. Sarovar, and P. J. Coles, Machine learning of noise-resilient quantum circuits, PRX Quantum **2**, 010324 (2021).

[67] Z. Yu, X. Zhao, B. Zhao, and X. Wang, Optimal quantum dataset for learning a unitary transformation, Phys. Rev. Appl. **19**, 034017 (2023).

[68] D. F. Locher, L. Cardarelli, and M. Müller, Quantum error correction with quantum autoencoders, Quantum **7**, 942 (2023).

[69] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms* (Cambridge University Press, Cambridge, 2014).

[70] S.-X. Zhang, J. Allcock, Z.-Q. Wan, S. Liu, J. Sun, H. Yu, X.-H. Yang, J. Qiu, Z. Ye, Y.-Q. Chen, *et al.*, Tensorcircuit: A quantum software framework for the NISQ era, Quantum **7**, 912 (2023).

[71] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, Evaluating analytic gradients on quantum hardware, Phys. Rev. A **99**, 032331 (2019).

[72] D. Wierichs, J. Izaac, C. Wang, and C. Y.-Y. Lin, General parameter-shift rules for quantum gradients, Quantum **6**, 677 (2022).