Symmetry-guided gradient descent for quantum neural networks

Kaiming Bian,[1,2] Shitao Zhang,[1] Fei Meng,[3,*] Wen Zhang,[4,†] and Oscar Dahlsten[3,1,5,‡]

[1]*Shenzhen Institute for Quantum Science and Engineering, Southern University of Science and Technology, Nanshan District, Shenzhen 518055, China*

[2]*Department of Physics, Southern University of Science and Technology, Nanshan District, Shenzhen 518055, China*

[3]*Department of Physics, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong SAR, China*

[4]*HiSilicon Research, Huawei Technologies Co., Ltd., Shenzhen 518129, China*

[5]*Institute of Nanoscience and Applications, Southern University of Science and Technology, Shenzhen 518055, China*

(Received 23 January 2024; revised 24 June 2024; accepted 11 July 2024; published 5 August 2024)

Many supervised learning tasks have intrinsic symmetries, such as translational and rotational symmetry in image classifications. These symmetries can be exploited to enhance performance. We formulate the symmetry constraints into a concise mathematical form. We design two ways to adopt the constraints into the cost function, thereby shaping the cost landscape in favor of parameter choices, which respect the given symmetry. Unlike methods that alter the neural network circuit *Ansatz* to impose symmetry, our method only changes the classical postprocessing of gradient descent, which is simpler to implement. We call the method symmetry-guided gradient descent (SGGD). We illustrate SGGD in entanglement classification of Werner states and in two classification tasks in a two-dimensional feature space. In both cases, the results show that SGGD can accelerate the training, improve the generalization ability, and remove vanishing gradients, especially when the training data is biased.

DOI: 10.1103/PhysRevA.110.022406

I. INTRODUCTION

Quantum machine learning extends concepts from classical machine learning into the regime of quantum superposition [1,2]. Quantum neural networks can be viewed as quantum generalizations of classical neural networks, amounting to an extension of deep learning to the quantum regime [3–5]. QNNs have shown promising results, for example, in quantum phase recognition [6] and classical classification tasks [7]. QNNs employ parameterized quantum circuits (PQC) [4,8–10], with a classical optimizer updating the parameters during training. There are now several variants including quantum graph neural networks [11,12], quantum convolution neural networks [6,13], and tensor network-based QNNs [14–16].

Leveraging symmetry can improve the performance of both classical and quantum neural networks. Inspired by successful machine learning models, a field known as geometric deep learning investigates the relation between symmetry and learning [17,18]. Recently, such ideas have been extended into the quantum regime, establishing the field of geometric quantum machine learning [19–22]. To leverage symmetry one may modify the quantum circuits being trained to gain significant performance improvements [22–24]. However, the symmetry-preserving modification entails altering the PQC *Ansatz*, which can be experimentally costly. In specific scenarios (see Appendix A) such modifications of quantum circuits even transform local unitaries into global unitaries, which are difficult to implement in experiments. Such potential hardware inefficiencies motivated us to explore further methods to leverage symmetry properties to optimize the performance of quantum neural nets.

We give a mathematically justified method to modify the cost function to guide the parameters toward the space with the desired symmetry, a method which we call symmetry-guided gradient descent (SGGD). We formulate the symmetry constraints as a neat equation so that the constraints can easily be rewritten as a penalty term. Specifically, we rewrite the symmetry constraint into a concise equation and modify the cost function to effectively guide the optimization of parameters, steering the quantum circuit to satisfy the symmetry. We designed two ways to implement SGGD, which involve averaging an observable with a symmetry group, a process called twirling. In one approach, the cost function gains a penalty term that suppresses the appearance of symmetry-breaking circuits by twirling. In the other approach, the original observable is replaced with the twirled one when calculating the cost function.

We illustrate SGGD in numerical experiments with both quantum and classical inputs. For the case of quantum input, we consider the entanglement classification of Werner states [25]. SGGD is shown to prevent biased sampling of input states from leading to poor generalization, dramatically reducing the required training data size. Additionally, it sharpens the slope of the cost curve, which leads to faster training. For the case of classical inputs, we consider classifying a set of two-dimensional (2D) points with rotational symmetry into a

*Contact author: feimeng@cityu.edu.hk
†Contact author: zhangwen20@huawei.com
‡Contact author: oscar.dahlsten@cityu.edu.hk

few categories. The 2D data are encoded into quantum states by rotation gates [26,27]. The encoded states with their labels are fed into a PQC for training. SGGD increases the accuracy from 69.9%–89.6% in classification into two classes, and increases the accuracy from 49.5%–66.1% in classification into three classes when the training data is highly biased. In both cases, our method can improve the generalization ability significantly, especially when confronted with biased training data—a prevalent challenge within high-dimensional spaces [18,28].

Our method, SGGD, is widely applicable to gradient descent over QNNs and other PQCs. The SGGD approach of changing the cost function is more convenient than altering the PQC *Ansatz* because the modification is implemented classically, which does not impose additional workload on quantum devices. As demonstrated in the numerical examples, SGGD successfully reduces the occurrence of circuits that do not satisfy the symmetry, thereby improving the efficiency of the QNN model.

II. RESULTS

QNN setup. In QNN training, the PQC processes input data on a quantum chip while the classical optimizer runs on a classical computer to update parameters [29,30]. The input data needs to be encoded into the quantum state if it is classical [26,31,32]. After the state is evolved by the PQC $U(\theta)$, commonly the output of QNN is determined by measuring an observable O and obtaining its expectation value

$$f_\theta(x) = \mathrm{tr}[(U(\theta)\rho(x)U^\dagger(\theta)O)], \tag{1}$$

where the encoding ρ maps classical data x into the density matrix $\rho(x)$. Now, a classification function h, as classical postprocessing, is applied to map this expectation value to the predicted label. For instance, a step function h may be employed by the model to predict the label of x; if $f_\theta(x) \geqslant 0$, the model predicts the label of x as 1; otherwise, it predicts the label of x as -1.

Symmetry constraint equation. The data label in many supervised learning tasks is invariant under certain symmetries. For example, the number of 1s in a bit string remains unchanged through swapping two bits [see Fig. 1(a)]. Reference [18] gives a formal expression of an $\mathcal{S}$-invariant data label:

$$f(s(x)) = f(x),\ \forall x \in \mathcal{X},\ \forall s \in \mathcal{S}, \tag{2}$$

where $\mathcal{S}$ denotes the group that consists of symmetry operations. We describe the case where the classical postprocessing function h is identity first before examining the case of general h.

If an objective function f_θ could approximate the target function f, there is an optimal parameter θ^* such that $f_{\theta^*}(s(x)) = f_{\theta^*}(x) + \epsilon$, where ϵ is a tolerable error. Combining this condition with Eqs. (1) and (2), the optimal PQC $U(\theta^*)$ that satisfies the symmetry of the data should obey

$$\mathrm{tr}(\rho(x)S^\dagger \widetilde{O}(\theta^*)S) = \mathrm{tr}(\rho(x)\widetilde{O}(\theta^*)) + \epsilon, \tag{3}$$

for all S in a symmetry matrix group, where $\widetilde{O}(\theta)$ is $U^\dagger(\theta)OU(\theta)$. It can be observed that, as long as the encoded states $\{\rho(x)\}$ do not strictly reside within a linear subspace of the entire Hilbert space $\mathcal{H}$, Eq. (3) is equivalent to the following symmetry constraint equation (for the detailed derivation, see Appendix B):

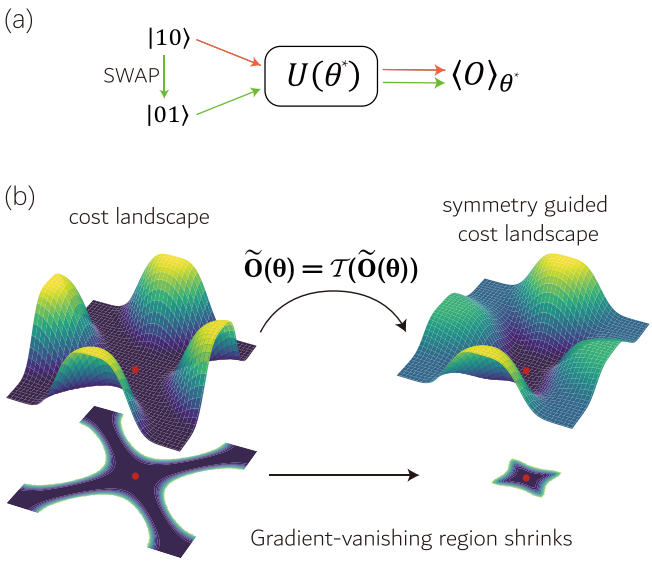


FIG. 1. Imposing symmetry of a task on the cost landscape. (a) Suppose $|1,0\rangle = \mathrm{SWAP}|0,1\rangle$ and $|0,1\rangle$ should have the same label so that SWAP is a task symmetry. A circuit with optimal parameter θ^* gives output $\langle O\rangle_{\theta^*}$ determining the assigned label. (b) The neural net circuit is trained via gradient descent on a cost landscape. Using the symmetry constraint equation (4), we shape the cost landscape given the symmetry of the data labels. In this example, the gradient-vanishing region around the optimal parameter θ^* then shrinks, where θ^* is represented by the red dot. The symmetry guidance method moreover improves training performance given biased data.

$$\mathcal{T}(\widetilde{O}(\theta^*)) = \widetilde{O}(\theta^*) + E, \tag{4}$$

where E is a tolerable-size error matrix, which satisfies $\|E\| < \epsilon$. $\mathcal{T}$ is the twirling operator, which maps an operator O into the symmetry-twirled Hamiltonian

$$\mathcal{T}(O) = \int d\mu(S) SOS^\dagger, \tag{5}$$

where μ is the Haar measure.

Symmetry guided gradient descent (SGGD). Gradient descent uses the gradient of the cost function as the direction for updating parameters. A general method for guiding gradient descent is to include an extra penalty term [33]. We say a gradient descent is guided by symmetry if the cost function is modified to make the parameters satisfy the symmetry constraint equation Eq. (4). We now describe how to create a penalty term for a given symmetry.

We take the mean square error (MSE) as a default cost function $c_0(\theta)$, and construct a penalty term $g(\theta)$ that employs the Hilbert-Schmidt norm to evaluate the difference between $\widetilde{O}(\theta)$ and $\mathcal{T}(\widetilde{O}(\theta))$,

$$\begin{aligned} c_1(\theta) &= c_0(\theta) + \lambda g(\theta) \\ &= \frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}}[f_\theta(x) - f(x)]^2 + \lambda\|\mathcal{T}(\widetilde{O}(\theta)) - \widetilde{O}(\theta)\|, \end{aligned} \tag{6}$$

where $\mathcal{X}$ is the data set and λ controls the intensity of the penalty term. The penalty term $g(\theta)$ would reach the minimum

value 0 when the circuit $U(\theta)$ respects the symmetry constraint equation. Hence, the $g(\theta)$ guides the parameters toward the symmetry-preserving space during gradient descent. As illustrated in Fig. 1(b), the gradient vanishing region shrinks after including the penalty term in the cost function, implying an improvement in parameter training. Moreover, $g(\theta)$ aids in discovering the optimal θ irrespective of the quality or the quantity of data as the penalty constrains θ to the region with symmetry, compensating for the lack of suitable training data.

The SGGD method could be directly applied to other gradient based methods, such as ADAM [34] and mirror gradient descent [20], by adding the penalty term to their cost function. The performance can be further improved by tuning the intensity coefficient λ. Thus, SGGD can be added to gradient based methods rather than being mutually exclusive.

As a second way to incorporate symmetry guidance, we directly substitute the symmetry constraint equation into the objective function f_θ,

$$c_2(\theta) = \frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}}\{\mathrm{tr}[\rho(x)\mathcal{T}(\widetilde{O}(\theta))] - f(x)\}^2. \tag{7}$$

This cost function is guided by symmetry in the sense that only those parameters that lead to variations in $\mathcal{T}(\widetilde{O}(\theta))$ can change the overall cost value c_2.

The two methods differ in terms of what exactly is optimised and in their computational cost. c_1 only cares about preserving symmetry rather than predicting accuracy, while c_2 cares both about symmetry and the accuracy of the QNN. However, c_2 introduces more burden in terms of quantum devices since it makes explicit use of the twirled observable $\tilde{O}$ [35] whereas c_1 can be evaluated via a swap test between the twirled and nontwirled observables. While the twirled quantities can be calculated efficiently with a quantum device, in the numerical experiments in this paper we calculate c_1 and c_2 via naive and inefficient classical simulation. Whether calculated with quantum devices or classical methods, the penalty term only needs to be calculated once per training epoch without looping over training inputs.

To illustrate and explain how and when SGGD is useful, we analyze two examples. We choose examples that are nontrivial yet sufficiently simple to also be analyzed by hand so one can gain a full understanding of when and why SGGD works. In some cases, the symmetry is an invariance of the classical label and in some cases the symmetry is an invariance of the quantum output under a unitary operation. The first example we consider is of the latter type and the second example is of the former.

Example: Entanglement classification. We now analyze the task of entanglement classification of two-qubit Werner states. This example illustrates how our methods alleviate bad performance caused by biased training data.

Consider the task to classify Werner states [36,37] as entangled or separable. Two-qubit Werner states can be expressed as

$$\rho_{\mathrm{W}} = \frac{2-p}{6}I + \frac{2p-1}{6}\mathrm{SWAP}, \tag{8}$$

where the parameter p ranges from -1 to 1. SWAP is a unitary and Hermitian matrix defined by $\mathrm{SWAP}|i,j\rangle = |j,i\rangle$. Werner states are entangled for $p < 0$ and separable for $p \geqslant 0$ [38].

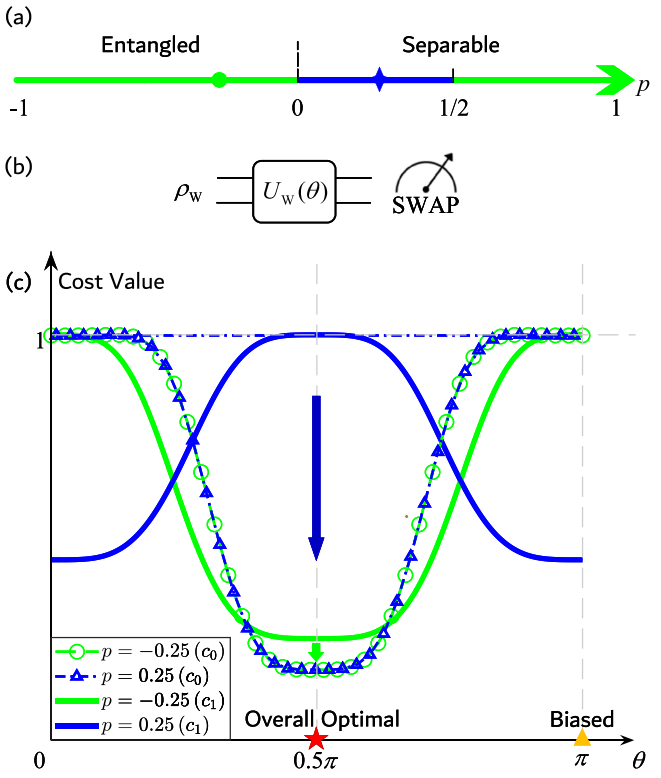


FIG. 2. Settings and loss landscapes comparison for the classification of two-qubit Werner states. (a) illustrates how the entanglement of Werner states changes along with the parameter p; states in different color areas have different kinds of loss landscapes in our case. (b) shows the QNN structure being used. (c) shows the loss curves before (c_0) and after (c_1) applying symmetry guidance for two representative samples from each color area, with $p = -0.25$ and 0.25, respectively. Curves in this figure are rescaled, with each value divided by the maximum value of its corresponding curve.

We take the demanded symmetry matrix group to be all tensor products of single-qubit unitaries, $U \otimes U$, because Werner states are invariant under these operations.

Consider how to employ a QNN for the above task. Without loss of generality, we say a state is classified as entangled if the output of the QNN satisfies $f_\theta(\rho_{\mathrm{W}}) < 0$ and as separable otherwise. The simplest QNN that is composed of the identity circuit followed by measuring the SWAP operator as the observable will output $f_\theta(\rho_{\mathrm{W}}) = p$, and thus classify the state correctly. For simplicity and clarity, we fix the observable as SWAP and parametrize the circuit as a unitary $U_W(\theta)$, where θ is the variational parameter to be trained. We assume a one-dimensional parameter as it helps to visualize the loss landscape easily. We use the following QNN *Ansatz*,

$$U_{\mathrm{W}}(\theta) = \mathrm{CNOT}^{\cos^2(\theta)}, \tag{9}$$

which is simple enough to elaborate the mechanisms of SGGD while containing the unique optimal solution (identity) and a biased solution [controlled-NOT (CNOT)]. The *Ansatz* structure is depicted in Fig. 2(b). The cost landscape highly depends on the training data chosen. As shown in Fig. 2, we find that the whole data set can be split into two subsets

$A = \{p \mid p \in [0, 1/2)\}$ and $\bar{A} = [-1, 1] - A$. Training data from $\bar{A}$ leads to a convex cost landscape that gives the correct solution under gradient descent, and training data from A yields a concave landscape that gives the biased solution. One can verify the above claim by depicting the cost landscape for each data point in A and $\bar{A}$, respectively, as shown in Fig. 2(c) with $p = -0.25$ and $p = 0.25$ as examples.

We find that SGGD alleviates the bias caused by training data in A. The symmetry group G of input Werner states is $\{U \otimes U \mid U \text{ is unitary}\}$. The penalty term given by Eq. (5) is

$$\mathcal{T}(\widetilde{O}(\theta)) = \alpha_1 I + \alpha_2 \mathrm{SWAP}, \tag{10}$$

where $\widetilde{O}(\theta) = U_{\mathrm{W}}^{\dagger}(\theta)\mathrm{SWAP}U_{\mathrm{W}}(\theta)$, $\alpha_1 = \cos^2(\frac{\eta}{2})/2$, $\alpha_2 = \sin^2(\frac{\eta}{2})$, and $\eta = \pi \sin^2\theta$. Substituting the twirled operator of Eq. (10) into Eq. (6) we get the penalty term for c_1 as

$$g(\theta) = \tfrac{3}{4}(\sin^2\eta + 2\cos\eta + 2). \tag{11}$$

Note that minimizing the penalty alone could optimize the parameter θ since $g(\pi/2) = 0$ is the minimal value, and $\theta = \pi/2$ gives the optimal QNN circuit.

Using the SGGD cost function c_1, all data points, no matter in A or $\bar{A}$, can give the optimal solution, as shown by the two dotted curves in Fig. 2(c). This implies that even if the majority of the training data is picked from A, QNN can still be trained correctly. SGGD thus dramatically improves the generalization performance of the QNN trained by biased data. Additionally, the cost curves of data from $\bar{A}$ become steeper, and the training converges faster, indicating that SGGD can accelerate the training of QNNs. Experimental details and alternative circuit *Ansätze* are given in the Appendix E 2.

Example: Classification of two-dimensional classical data. Consider the task of classifying classical data points on a 2D plane [39] into two classes using hardware-efficient PQCs [40]. As shown in Fig. 3(a), training data are sampled in the range $[0, 1]^2$, and data points are classified into two categories depending on if their distance to the center point $(\frac{1}{2}, \frac{1}{2})$ is smaller than 0.2 or not (a case of nonlinear classification). This task can be solved by QNNs with high accuracy when the training data are sampled uniformly from the whole data space [39]. However, models can exhibit poor generalization performance when training samples are not uniformly sampled, e.g., when data is sampled from only the right half of the data space, as shown in Fig. 3(a). We choose one of the effective QNNs in Ref. [39], whose circuit structure is detailed in the Appendix E 3, and compare its prediction performances using the cost function c_0 and the SGGD cost function c_2. SGGD with cost function $_2$ increases the accuracy from 69.9%–89.6%. The reason we choose c_2 here is that c_1 is overly restrictive, imposing symmetry at the level of the quantum output before the classical postprocessing. We do not want the training to rule out models that respect the symmetry of the label but are not symmetric before the postprocessing.

SGGD also performs well on a generalization to the above 2D classification task with three classes and a larger quantum circuit with 16 qubits, as shown in Fig. 4. The data points in $[0, 1]^2$ with coordinates (x, y) are to be classified into three categories based on their distances r to the center point (0.5,0.5). If $r < 0.2$, then the data point belongs to class 0; if $r > 0.4$, then it belongs to class 2; otherwise, it belongs to class 1. This classification can be solved by applying the two-class classifier we obtained above twice; the first classifier tells whether the point is located inside the inner circle, and the second classifier outputs whether the point is located inside the outer circle. Via binary encoding, each coordinate x or y is truncated into a four-digit binary string $\mathbf{x}$ or $\mathbf{y}$ and is represented by four qubits, respectively, resulting in a eight-qubit state $|\mathbf{x}, \mathbf{y}\rangle$ that encodes a truncated data point. Then, we generate two copies of the state $|\mathbf{x}, \mathbf{y}\rangle^{\otimes 2}$ using 16 qubits, where each copy is fed into the two-class classifier quantum circuit, as shown in Fig. 4(b). The upper quantum circuit outputs whether the data point is inside the inner circle or not, and the bottom circuit outputs whether the data point is inside the outer circle or not. Combining the classification outcome of these two parallel circuits, we can classify the data points into one of the three classes. With highly biased training data shown in Fig. 4(a), the accuracy of prediction is only 49.5%, while using SGGD to modify the cost function increases the accuracy to 66.1%.

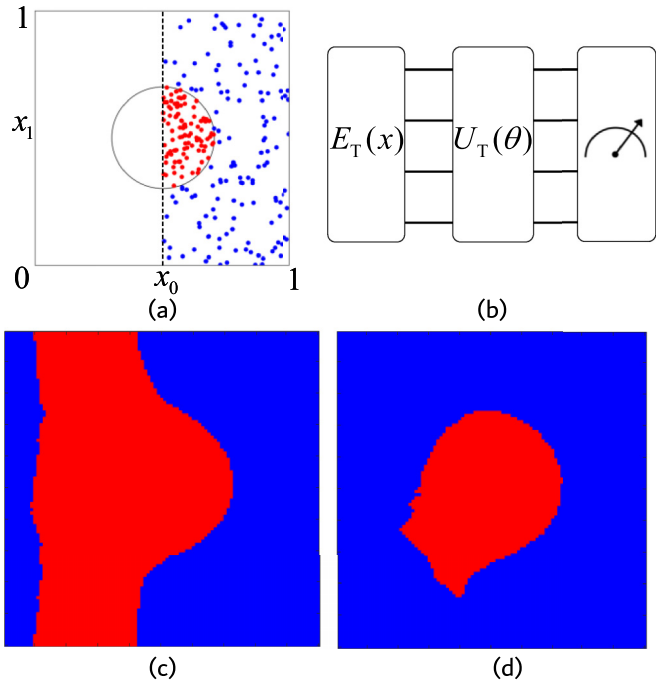


FIG. 3. SGGD enhances 2D classical data classification with two classes using rotation encoding. (a) illustrates the distribution of the training data, which is highly biased. The data points are classified into two categories marked as red and blue. (b) illustrates the QNN structure, where the rotation encoding $E_T(x)$ and circuit *Ansatz* $U_T(\theta)$ are detailed in the Appendix E 3. (c) and (d) demonstrate the predictions on test data without using SGGD (cost function c_0) and using SGGD (cost function c_2), respectively.

Nontrivial classical postprocessing. We now consider nontrivial classical postprocessing where h, a classical function of the output f, is not the identity.

In this case, the requirement for a symmetry-invariant circuit changed from Eq. (3) to $h(f_\theta[s(x)]) = h(f_\theta(x))$. Circuits that adhere to this equation are not necessarily bounded by the symmetry constraint equation (see details in the Appendix B). For instance, consider the labels assigned to $|01\rangle$ and $|10\rangle$,

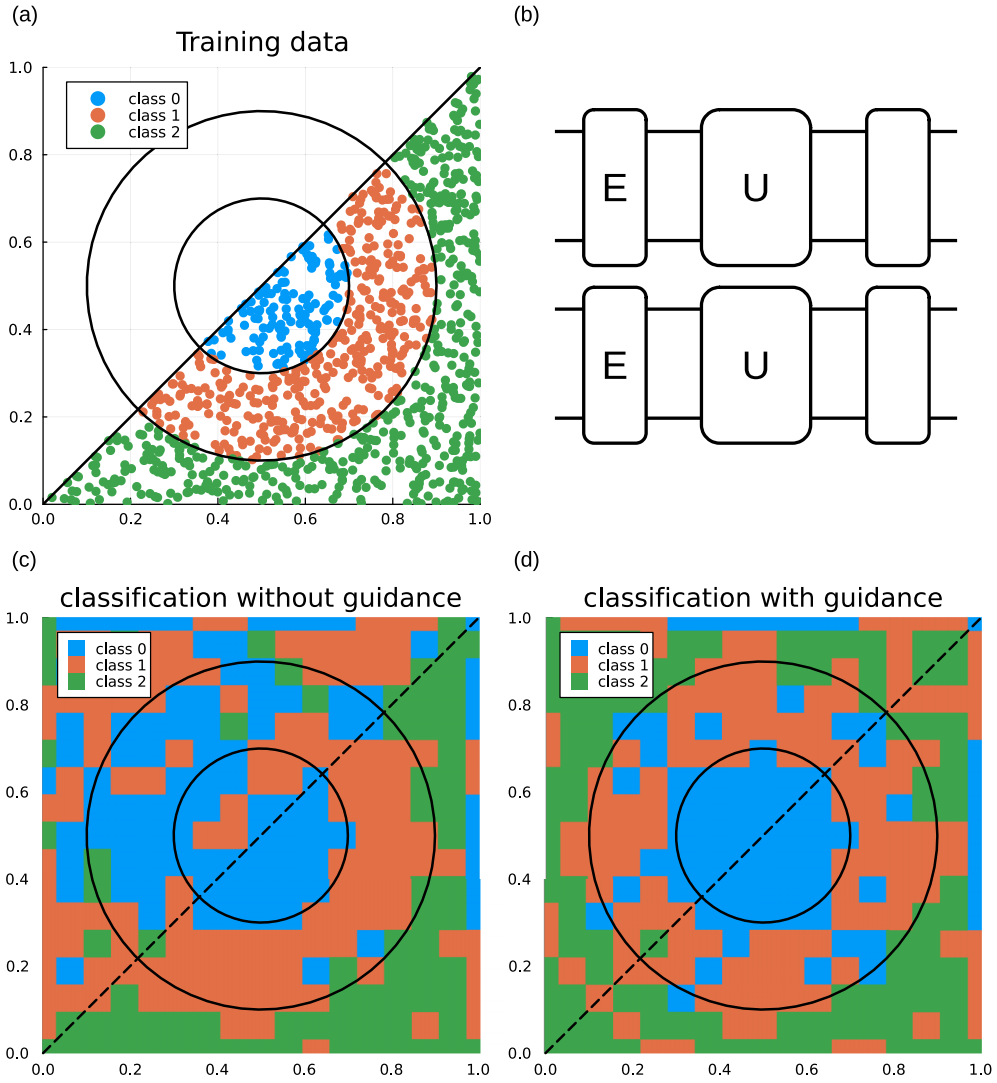


FIG. 4. SGGD enhances 2D classical data classification with three classes using binary encoding. (a) illustrates the highly biased distribution of the training data set. Points are labeled into three classes according to their relative position to the circles. (b) Two identical circuit *Ansätze* for two-class classification are applied to classify whether (i) the input data is inside the inner circle (the top branch) and (ii) inside the outer circle (the bottom branch). Each circuit branch has eight qubits. We use a binary encoder E and the hardware-efficient *Ansatz* U. (c) and (d) demonstrate the predictions on test data without using SGGD (cost function c_0) and using SGGD (cost function c_2), respectively.

both of which are assigned a label of 1. However, their respective probabilities of yielding the label 1 may differ. If we force the parameters or the circuit to obey the symmetry constraint equation, their probabilities to get label 1 are forced to be the same, which is overly restrictive and thus harms the expressive power of the PQCs.

We propose a modified version of the symmetry constraint equation that integrates classical postprocessing. The basic idea of modifying the symmetry constraint equation is considering h as an analytical function that can be expanded using a Taylor series. By truncating the series to the t-order polynomial h_t, the effect of the postprocessing could be formulated into a neat form,

$$h_t(f(x)) = \mathrm{tr}((U(\theta)\rho(x)U^\dagger(\theta))^{\otimes t} O_{h_t}(\theta)). \tag{12}$$

The operator O_{h_t} is determined by the truncated series. Let $\tilde{O}_{h_t}$ denote the operator $U^{\dagger\otimes t}(\theta)O_{h_t}U^{\otimes t}(\theta)$. We then prove that the requirement of $\mathcal{S}$-invariant data label leads to the modified symmetry constraint equation,

$$P(\mathcal{T}(\tilde{O}_{h_t}(\theta^*)) - \tilde{O}_{h_t}(\theta^*)) = 0, \tag{13}$$

where P projects an observable in a Hilbert space $\mathcal{H}$ into the symmetric subspace of $\mathcal{H}$. The proof and the concrete forms of O_{h_t} and P are shown in the Appendix C.

Connection to other results. The effect of SGGD is similar to incorporating the symmetrically transformed data into the training dataset. A frequently used method to utilize symmetry is data enhancement [41,42], which appends the data generated by the symmetry transformation of the training set. In the example of 2D classification, the training data from the right-hand side [as shown in Fig. 3(a)] is in a sense transformed to the upper side since the SWAP symmetry transformation swaps the x_0 and x_1 coordinates of the input data. In Fig. 3(d), the SGGD performs very well in the whole

space except for the left-bottom part. Thus, the result of SGGD is similar to data enhancement in this case.

However, data enhancement can lead to a significant increase in the size of the data set, which introduces computational overhead. In contrast, SGGD does not increase the training set size but adds overhead to the evaluation of the cost function. Since data enhancement creates more training data, the process of training models on the enhanced data may become more challenging [22,28]. An intuitive example is that, as the dimension increases, if we want to maintain a constant data density within a unit volume, the total data volume must grow exponentially with the dimension. The required quantity of data increases exponentially with the dimensions [18]. Linearly increasing the data size is insufficient in high-dimensional spaces. Consequently, resorting to the twirling technique becomes a more viable option in such scenarios because the SGGD achieves similar results without significantly increasing the number of training data points. Balancing the cost overhead of increasing the data set size and the computational cost in the cost function can help to choose a suitable method.

The SGGD approach has a challenge in common with other penalty term approaches and we find a method to deal with this challenge. The challenge is that in principle there is a risk of the modification to the cost function impacting the training results negatively [43,44]. This issue appeared in our numerical experiments of 2D classification. To alleviate this problem, we adjust the weight of the penalty term dynamically during training. We incrementally augment the weight of the penalty term to ensure the parameters are not overguided by the symmetry constraint. This gradual increase in the size of the penalty term results in successful training that outperforms the nonguided training. (See details in the Appendix D.)

III. CONCLUSION AND OUTLOOK

We impose symmetry constraints on the QNN's training, establishing a symmetry-guided gradient descent (SGGD) method. Our results show that SGGD can accelerate the training, improve the generalization ability, and remove vanishing gradients, especially when the training data is biased. Our method contributes understanding of how symmetry can be efficiently exploited in quantum machine learning and is widely applicable to tasks with either classical or quantum input data.

Apart from applying SGGD for enhancing QNN performance, further developments should be investigated. Optimal strategies should be explored for (i) varying the strength of the symmetry guidance during the training, and (ii) the norm used in defining the guidance [45]. It should also be investigated to what extent the method enhances the performance of more sophisticated forms of gradient descent such as the generalisation of gradient descent known as mirror descent [20]. The general method can also be adapted to guide circuit design more generally: one expects that good designs for a given problem should have a small symmetry penalty.

ACKNOWLEDGMENTS

We acknowledge inspiring discussions with Dong Yang and Jin-Long Huang. We acknowledge support from HiSilicon, the National Natural Science Foundation of China (Grants No. 12050410246, No. 1200509, No. 12050410245), and the City University of Hong Kong (Project No. 9610623).

APPENDIX A: HARDWARE INEFFICIENCY OF CIRCUIT *ANSATZ* SYMMETRIZATION

One approach commonly employed to leverage symmetry is modifying the circuits to adhere to the constraints imposed by symmetry, which is referred to as *Ansatz* symmetrization. An illustrative example is the gate symmetrization method described in Ref. [22].

We will first introduce the gate symmetrization procedure and then discuss the overhead that makes such symmetrization hardware inefficient [40]. Assume that the *Ansatz* comprises a set of gates denoted as G, given by

$$G = \{e^{ig_1\theta_1}, e^{ig_2\theta_2}, e^{ig_3\theta_3}, \cdots\}, \tag{A1}$$

where the g_i represents the generators of the *Ansatz*. As an example, let us consider a hardware-efficient *Ansatz* consisting of rotation gates $R_X(\theta)$ and controlled-z gates,

$$\left\{e^{iX\theta}, e^{-iA\frac{\pi}{4}}\right\}, \tag{A2}$$

where $A = I - Z_1 - Z_2 + Z_1Z_2$. The gate symmetrization technique twirls the set of generators to fulfill the symmetry requirement. For the previously mentioned hardware-efficient *Ansatz*, the symmetrized circuit has the generators $\{\mathcal{T}(X), \mathcal{T}(A)\}$. If the output of the quantum circuit is invariant under the $\{I, \text{SWAP}\}$ operation, the twirled generators can be expressed as

$$\mathcal{T}(X_1) = \frac{1}{|\mathcal{S}|}\sum_{S\in\mathcal{S}} SX_1S^\dagger = \frac{1}{2}[X\otimes I + \text{SWAP}(X\otimes I)\text{SWAP}]$$
$$= \frac{1}{2}(X_1 + X_2)$$

and $\mathcal{T}(A) = A$. In the main text, we define that $\mathcal{S}$ is a discrete symmetry group. By utilizing these twirled generators, a new set of gates is constructed,

$$\left\{e^{i(X_1+X_2)\frac{\theta}{2}}, e^{-iA\frac{\pi}{4}}\right\}. \tag{A3}$$

This new gate set is then used to assemble a new quantum circuit that satisfies the required symmetry. Consequently, a quantum circuit that preserves symmetry is constructed through the twirling process. The procedure of constructing a symmetry-preserving QNN from the generators is referred to as *Ansatz* symmetrization.

Now, we will analyze a specific case in which the symmetrization process will introduce an extra burden to the quantum device. Considering we apply a ZZ gate $e^{i\theta Z_1Z_2}$ on the quantum circuit, and the output of QNN is invariant under the permutation operators. The symmetrization of this gate gives the generator

$$\mathcal{T}(Z_1Z_2) = \frac{1}{n!}\sum_{\sigma\in S}\sigma Z_1Z_2\sigma^\dagger = \frac{2}{n(n-1)}\sum_{i\neq j} Z_iZ_j, \tag{A4}$$

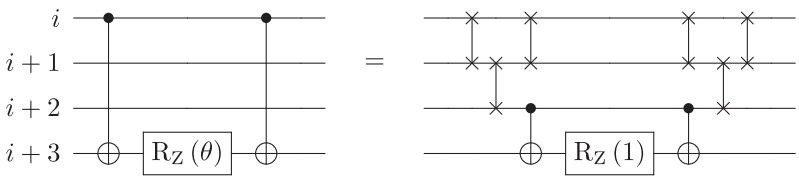
where σ is a permutation operator, and S consists all possible permutations. The realization of variational gates corresponds

to the generator $\mathcal{T}(Z_1Z_2)$ are

$$e^{i\mathcal{T}(Z_1Z_2)\theta} = \prod_{i,j} e^{i\frac{2\theta}{n(n-1)}Z_iZ_j}, \tag{A5}$$

which are pairwise connected ZZ gates over all qubits. Any two qubits in the circuit have a ZZ gate that connects them. Thus, in this example, the twirled gate in Eq. (A5) requires $\frac{n(n-1)}{2}$ gates to implement, which significantly increases the number of gates to implement.

Furthermore, in scenarios where the quantum device is limited to performing only adjacent two-qubit gates, nonadjacent gates must be decomposed into a sequence of adjacent gates. Most of the pairwise ZZ gates are not adjacent gates.



(A7)

To decompose a single Z_iZ_j gate, excluding the case where $|i-j|=1$, we require the addition of $2(2|i-j|-3)$ adjacent SWAP gates. The twirling of one ZZ gate needs $\frac{1}{3}(2n^3-9n^2+7n)$ adjacent gates to achieve,

$$\begin{aligned}&(n-1)+2(n-2)+6(n-3)+10(n-4)\\&\quad+\cdots+(4(n-1)-6)=\frac{1}{3}(2n^3-9n^2+7n)\sim\mathcal{O}(n^3),\end{aligned} \tag{A8}$$

which costs too many gates for the current noisy quantum devices.

In addition to gate symmetrization, other approaches aimed at constructing symmetry-preserving blocks within QNN encounter a similar challenge. Typically, the symmetry operators employed are global unitaries that span multiple qubits. Thus, as we have shown, it becomes evident that only a global gate can maintain global symmetry. A global gate is usually hard to achieve in quantum devices [46,47].

APPENDIX B: SYMMETRY CONSTRAINT EQUATION

In this section, we aim to give the proof of Eq. (4) in the main text. Recall the symmetry requirement

$$\mathrm{tr}(\rho(x)S^\dagger\widetilde{O}(\theta^*)S)=\mathrm{tr}(\rho(x)\widetilde{O}(\theta^*)), \tag{B1}$$

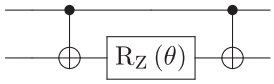
where S are symmetry operations, x is classical input data, and O is the observable. We want to prove that the Eq. (B1) is equivalent to the symmetry constraint equation,

$$\mathcal{T}(\widetilde{O}(\theta^*))=\widetilde{O}(\theta^*), \tag{B2}$$

if the data set $\{\rho(x)\}$ span the space of states.

In fact, Eq. (B1) are many equations because there are many symmetry operations. This equivalence merges many equations into one neat equation, which makes it easy to apply in the cost function.

If we want to apply the gate described in Eq. (A5), we need to decompose nonadjacent ZZ gates to adjacent basic gates. Now, we evaluate the cost of such decomposition. First, we decompose the ZZ gate into basic gates because current quantum devices can not directly apply it,



. (A6)

Then, the ZZ gate on ith and jth qubits need to be decomposed to adjacent gates. We could use SWAP gate to decompose a Z_iZ_j gate to adjacent ZZ gate,

Proof. First, we give the case of discrete groups $\mathcal{S}$. To show that Eq. (B1) and Eq. (B2) are equivalent, we need to show that the solution sets of the two equations are the same. Precisely, we need to prove $A\subset B$ and $B\subset A$, where

$$A:=\{U\mid \mathrm{tr}(\rho(x)S^\dagger\widetilde{O}(\theta^*)S)=\mathrm{tr}(\rho(x)\widetilde{O}(\theta^*))\} \tag{B3}$$

$$B:=\{U\mid \mathcal{T}(\widetilde{O}(\theta^*))=\widetilde{O}(\theta^*)\}. \tag{B4}$$

Recall that $\tilde{O}(\theta):=U^\dagger(\theta)OU(\theta)$.

We begin with the proof of $A\subset B$. $U\in A$ means

$$\sum_{S\in\mathcal{S}}\mathrm{tr}(\rho(x)S^\dagger\tilde{O}(\theta)S)=|\mathcal{S}|\mathrm{tr}(\rho(x)\tilde{O}(\theta))$$

$$\mathrm{tr}(\rho(x)\frac{1}{|\mathcal{S}|}\sum_{S\in\mathcal{S}}S^\dagger\tilde{O}(\theta)S)=\mathrm{tr}(\rho(x)\tilde{O}(\theta))$$

$$\mathrm{tr}\left(\rho(x)\left(\frac{1}{|\mathcal{S}|}\sum_{S\in\mathcal{S}}S^\dagger\tilde{O}(\theta)S-\tilde{O}(\theta)\right)\right)=0,\ \forall\rho(x)\in\mathcal{H}.$$

Because $\{\rho(x)\}$ span the space, $\mathrm{tr}(\rho(x)A)=0\ \forall\rho(x)$ implies $A=0$. Therefore, the following equation holds,

$$\frac{1}{|\mathcal{S}|}\sum_{S\in\mathcal{S}}S^\dagger\tilde{O}(\theta)S-\tilde{O}(\theta)=0. \tag{B5}$$

Thus, Eq. (B2) holds. It means $\forall U\in A$, $U\in B$, thereby $A\subset B$. Now, we prove that $B\subset A$. For any $U\in B$ and any $S_0\in\mathcal{S}$, we have

$$\begin{aligned}\mathrm{tr}(\rho(x)S_0^\dagger\tilde{O}(\theta)S_0)&=\mathrm{tr}\left(\rho(x)S_0^\dagger\left(\frac{1}{|\mathcal{S}|}\sum_{S\in\mathcal{S}}S^\dagger\tilde{O}(\theta)S\right)S_0\right)\\&=\mathrm{tr}\left(\rho(x)\left(\frac{1}{|\mathcal{S}|}\sum_{S\in\mathcal{S}}S_0^\dagger S^\dagger\tilde{O}(\theta)SS_0\right)\right)\\&=\mathrm{tr}\left(\rho(x)\left(\frac{1}{|\mathcal{S}|}\sum_{S\in\mathcal{S}}S^\dagger\tilde{O}(\theta)S\right)\right).\end{aligned} \tag{B6}$$

In the first line, we just substituted the Eq. (B2) into the left-hand side. The last line is based on the rearrangement theorem of a group. Again, using Eq. (B2), the Eq. (B6) becomes

$$\begin{aligned}\mathrm{tr}(\rho(x)S_0^\dagger\tilde{O}(\theta)S_0) &= \mathrm{tr}\left(\rho(x)\left(\frac{1}{|\mathcal{S}|}\sum_{S\in\mathcal{S}}S^\dagger\tilde{O}(\theta)S\right)\right)\\ &= \mathrm{tr}(\rho(x)\widetilde{O}(\theta)).\end{aligned} \tag{B7}$$

Thus, for all U in B, U is also in A, which means $B\subset A$. Combining the result $A\subset B$, we get that $A=B$. For continuous groups (Lie groups), the proof is the same because the rearrangement theorem works for both discrete groups and continuous groups. ■

APPENDIX C: CLASSICAL POSTPROCESSING

In this section, we discuss how to derive the penalty term of SGGD to the cost function when there is a nontrivial postprocessing on the output of the QNN. Our analysis presented in this section greatly increases the applicability of SGGD, as in many cases postprocessing cannot be avoided—the output of QNN is not the class label of the data but the probability of the data being in each class. For instance, assuming the input state $|\psi\rangle = |010\rangle$ has a label of 1. A QNN that yields $f(|010\rangle) = 0.9$ signifies that the model assigns high probability to the state $|010\rangle$ correctly corresponding to label 1, rather than label the state $|010\rangle$ as 0.9. Requiring the invariance of the output of QNN with respect to symmetry is too restrictive, as it requires $f(|010\rangle) = f(S|010\rangle) = 0.9$ for all the symmetry operator S. However, if $f(|010\rangle) = 0.9$ but $f(|010\rangle) = 0.8$, then the state $|010\rangle$ will again be assigned label 1, and the symmetry of the data label is respected. Suppose the postprocessing that takes the probability to the label is the step function Θ, where

$$\Theta(x) = \begin{cases} 1 & \text{if } x \geqslant 0 \\ -1 & \text{if } x < 0 \end{cases} \tag{C1}$$

Under this classical postprocessing, the label invariance is expressed as

$$\Theta(f(\rho)) = \Theta(f(S\rho S^\dagger)). \tag{C2}$$

Here, $f(\cdot)$ represents the output of the QNN, which provides the probability of the data belonging to each class.

Imposing symmetry to postprocessing benefits the training and performance of the QNN. As pointed out above, imposing the symmetry constraint on the output of QNN adds more constraints to the parameters than is required. If the circuit *Ansatz* is not capable of completely satisfying the symmetry, requiring the QNN output to be invariant to the symmetry operation may reduce the accessible region of the parameters to a very small set that does not contain the optimal parameter θ^*. This potentially makes a *Ansatz* incapable of satisfying the requirements $f(\rho) = f(S\rho S^\dagger)$. In this case, the optimal parameters θ^* are not in the symmetry-conserved space Ω'. The symmetry penalty term likely produces a negative effect, which we observed in our numerical experiments—the penalty term without postprocessing pushes the convergence to a suboptimal parameter with worse accuracy.

Here, we show that symmetry penalty terms can be designed with classical postprocessing considered. Our method of adding symmetry penalty terms incorporating classical postprocessing greatly increases the applicability of SGGD. Now, we introduce the detailed construction of the symmetry penalty term where there is classical postprocessing. Suppose the postprocessing function h is an analytical function that could be expanded by a Taylor series. Truncate the Taylor series in t order, we get an approximate value of $h(x)$,

$$h(x) = a_0 + a_1x + a_2x^2 + \cdots + a_tx^t + \mathcal{O}(x^{t+1}), \tag{C3}$$

where a_j is the jth order derivative of h, $a_j = \frac{d^jh}{dx^j}(0)$, and N is the order of $\rho(x)$. Applying the postprocessing to the model output $f_\theta(x)$, we get

$$\begin{aligned}h(f[\rho(x)]) = a_0 &+ a_1\mathrm{tr}(U(\theta)\rho(x)U^\dagger(\theta)O)\\ &+ a_2\mathrm{tr}(U(\theta)\rho(x)U^\dagger(\theta)O)^2 + \cdots.\end{aligned} \tag{C4}$$

Note that

$$\begin{aligned}&\mathrm{tr}(U(\theta)\rho(x)U^\dagger(\theta)O)^k\\ &\quad= \mathrm{tr}\big((U(\theta)\rho(x)U^\dagger(\theta))^{\otimes t}\,O^{\otimes k}\bigotimes I^{\otimes t-k}\big),\end{aligned} \tag{C5}$$

the Eq. (C4) becomes

$$h(f[\rho(x)]) = \mathrm{tr}\left((U(\theta)\rho(x)U^\dagger(\theta))^{\otimes t}\sum_{k=0}^{t}a_kO^{\otimes k}\otimes I^{\otimes t-k}\right). \tag{C6}$$

The deduction of this equation is in the Appendices. For simplicity, define the operator $F_{(h,t)}(O)$ as $F_{(h,t)}(O) := \sum_{k=0}^{t}a_kO^{\otimes k}\bigotimes\mathbb{1}^{\otimes t-k}$. While considering the classical postprocessing, the symmetry invariant requirement reads as

$$\begin{aligned}&\mathrm{tr}([U(\theta^*)S\rho S^\dagger U^\dagger(\theta^*)]^{\otimes t}F_{(h,t)}(O))\\ &\quad= \mathrm{tr}([U(\theta^*)\rho U^\dagger(\theta^*)]^{\otimes t}F_{(h,t)}(O))\end{aligned} \tag{C7}$$

for $\forall S\in\mathcal{S}_{\mathrm{Mat}}$ and optimal parameter θ^*. Notice that Eq. (C7) is not equivalent to the condition

$$\mathcal{T}(\tilde{O}_F(\theta^*)) = \tilde{O}_F(\theta^*), \tag{C8}$$

where $\tilde{O}_F(\theta^*) = U^\dagger(\theta^*)^{\otimes t}F_{(h,t)}(O)U(\theta^*)^{\otimes t}$. Here, we present an example to support the statement. Because the observable $A = I\bigotimes X - X\bigotimes I$ make $\mathrm{tr}(\rho(x)\otimes\rho(x)A) = 0$ for $\forall\rho(x)\in\mathcal{H}$, the twirled operator $\mathcal{T}(\tilde{O}_F(\theta^*)) = \tilde{O}_F(\theta^*) + A$ could also satisfy the symmetry invariant condition of Eq. (C7).

This counterexample implies a way of symmetry penalty term adjustment. Rather than the restriction in Eq. (C7), the constraint of optimal operator $\tilde{O}_F(\theta^*)$ turns to

$$\mathcal{T}(\tilde{O}_F(\theta^*)) - \tilde{O}_F(\theta^*) \in \mathcal{A}_t, \tag{C9}$$

where $\mathcal{A}_t := \{O|\mathrm{tr}(\rho(x)^{\otimes t}O) = 0, \forall\rho(x)\in\mathcal{H}\}$ is the solution space. Due to the Schur-Weyl duality [48], the solution space is equivalent to the symmetry subspace of the whole Hilbert space. Precisely, let the projection operator by

$$P = \frac{1}{t!}\sum_{\sigma\in\Xi}\sigma, \tag{C10}$$

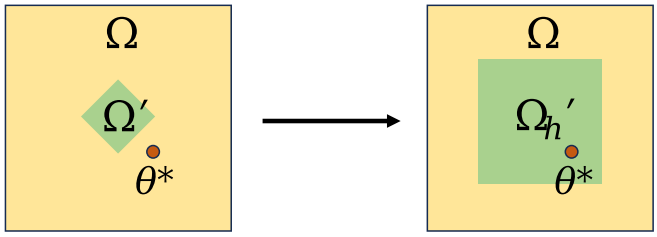


FIG. 5. Illustration of the effect of classical postprocessing. The green area of the left-hand side square means the solution space of the symmetry constraint equation. If non-trivial classical postprocessing is present, the imposed constraint may become excessively stringent such that the optimal parameter could not be located in the green solution space. The modification in Eq. (C11) loses the constraint so that the space with symmetry, the green area on the right-hand side, is enlarged.

where Ξ is the t-element permutation group. The solution space $\mathcal{A}_t$ is equivalent to the image of the projection P. Thus, Eq. (C9) induces that

$$(I - P)(\mathcal{T}(\tilde{O}_F(\theta^*)) - \tilde{O}_F(\theta^*)) = 0. \tag{C11}$$

Equation (C11) draws out the modified symmetry constraint equation. The modified symmetry penalty term is $g_{(h,t)}(\theta) = \|(I - P)(\mathcal{T}(\tilde{O}_F(\theta)) - \tilde{O}_F(\theta))\|$.

Figure 5 shows the effect of the adjustment. Because a solution of equation $g(\theta) = 0$ must be a solution of equation $g_{(h,t)}(\theta)$ while a solution of equation $g_{(h,t)}(\theta)$ may not be a solution of equation $g(\theta) = 0$, the symmetry-preserved parameter space expand after the adjustment. If a QNN is applied to classical postprocessing, adjustments must be made. As we mentioned before, a QNN model may not satisfy Eq. (C8) when the label is $\mathcal{S}$ invariant rather than the model output is $\mathcal{S}$ invariant. In this case, the optimal parameters lie out of Ω', which is the solution space of the symmetry constraint equation (C8). The adjustment of the symmetry penalty term makes the symmetry-preserved space, Ω'_h large enough to contain the optimal parameter, where Ω'_h is the solution space of the modified symmetry constraint equation (C11).

APPENDIX D: ADJUSTING THE COEFFICIENT OF THE SYMMETRY PENALTY TERM TO GAIN BETTER TRAINING PERFORMANCE

In this section, we present a strategy to adjust the coefficient of the symmetry penalty term to achieve better training of a QNN. In the research of quantum supervised learning, one of the frequently chosen cost functions is the mean-squared error between data labels and predictions,

$$c(\theta, \mathcal{X}) = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} (f_\theta(\rho(x)) - f(\rho(x)))^2, \tag{D1}$$

where $\mathcal{X}$ is the data set. We add the symmetry penalty term to this cost function,

$$c_g(\theta, \mathcal{X}) = c(\theta, \mathcal{X}) + \lambda g(\theta). \tag{D2}$$

Here, the λ controls the intensity of the symmetry penalty term.

With cost function c_g, the data is updated in the following way during gradient descending,

$$\theta^{(i+1)} = \theta^{(i)} + \frac{dc(\theta, \mathcal{X})}{d\theta}\bigg|_{\theta^{(i)}} \eta + \lambda \frac{dg(\theta)}{d\theta}\bigg|_{\theta^{(i)}} \eta, \tag{D3}$$

where $\theta^{(i)}$ is the parameter of the ith iteration, and η is the learning rate. The extra term $\lambda \frac{dg(\theta)}{d\theta}|_{\theta^{(i)}} \eta$ contributes a force that leads to the symmetry-preserved space Ω'. What is more, the symmetry penalty term $g(\theta)$ would reach the minimum value 0 when parameters θ are in the Ω'. Hence, the $g(\theta)$ guides the parameters toward Ω', and it vanishes when $\theta \in \Omega'$.

The controlling coefficient λ should be chosen carefully for practical applications. Notice that the symmetry penalty term only cares about symmetry rather than if the model makes the correct prediction. If the symmetry penalty is assigned with a large weight at the beginning, then the symmetry may misguide the neural net to a wrong parameter configuration. As shown in Fig. 6, if the symmetry-preserved space Ω'

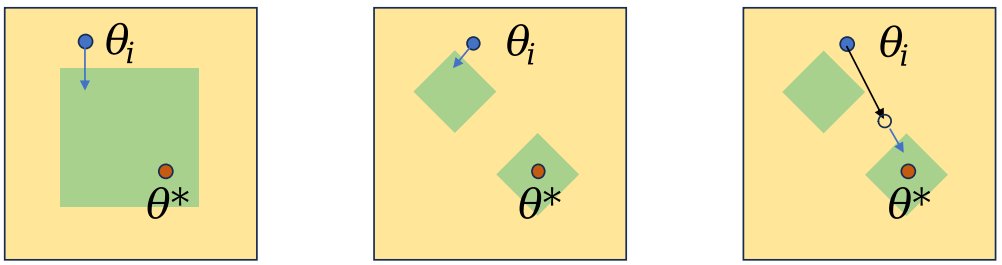


FIG. 6. The misleading problem of the symmetry penalty term can be relieved by adjusting the weight of the penalty term dynamically. The yellow square represents the whole parameter space, and the green area represents the space with symmetry. (a) shows a working case that the symmetry can successfully guide the parameter θ_i closer to the optimal point θ^*, where the subscript i means the parameter of ith iteration. (b) shows that the symmetry guides the parameter into the wrong place. In this case, there are two unconnected symmetry-preserving spaces, and the symmetry penalty term will attract the training to the wrong side if the initial configuration is close to the upper left green region. (c) shows the case when we apply the dynamic coefficient. The parameter will update without symmetry first, whose process is labeled by the black arrow. Then, the parameter will be updated with the help of symmetry. The dynamical coefficient could solve the misleading issue if we choose a proper strategy of changing the coefficient λ.

is not connected, there are several islands in Ω'. When λ makes $\lambda\frac{dg(\theta)}{d\theta} \gg \frac{dc(\theta,\mathcal{X})}{d\theta}$, the parameters tend to be trapped in one of the isolated regions where symmetry is satisfied. However, once the parameters are initialized close to a region that does not contain the optimal parameter, the training will be trapped in a local minimum, and performance will be bad. In this sense, we say that the symmetry misleads parameters.

However, the symmetry misleading phenomenon could be relieved by a proper coefficient λ picking strategy. Several efficient strategies are given here. The strategy uses dynamic intense controlling coefficient, which is shown in Fig. 6. We could let the λ be small at the beginning and let it become larger as the number of iterations increases. This strategy could be interpreted as one kind of pretraining. In the beginning, the λ is negligible while $\frac{dc(\theta,\mathcal{X})}{d\theta}$ is dominant. At this time, the original cost function c leads the parameters close to the optimal parameters θ^*. After the parameters θ close to θ^*, the λ becomes larger and larger, and the symmetry penalty term begins to have an impact that cannot be ignored. Then, the misleading phenomenon disappears in this case because the model parameters are close enough to the optimal one.

Another strategy is fixing an appropriate λ. If the lambda is bound by the following inequality in most places, the cost function will be the dominant term under most circumstances,

$$\lambda \leqslant \left|\frac{dc}{dg}\right|. \tag{D4}$$

Because the bounded λ makes $\lambda\frac{dg(\theta)}{d\theta} \leqslant \frac{dc(\theta,\mathcal{X})}{d\theta}$ for most $\theta \in \Omega$. In this case, the symmetry penalty term just assists the original cost function c. Although its hard to select a subset $\Upsilon \subset \Omega$ that satisfies $\mu(\Upsilon) = (1-\epsilon)\mu(\Omega)$, where μ is a measure in Ω and ϵ is a small number, we could select the λ empirically. In the numerical experiments we did, taking $\lambda = \frac{1}{3}\max(|\frac{dc}{dg}|)$ is small enough to avoid such misleading.

Both strategies have their advantages. The utilization of an unsuitable dynamic coefficient can have detrimental effects on performance, compromising the effectiveness of the approach. On the other hand, developing a well-designed dynamic algorithm necessitates a considerable investment of time and effort. While the second strategy presents practical convenience in its implementation, it is important to note that the efficacy of a fixed parameter λ employed in this approach falls short when compared to the performance achieved by a dynamically adjusted parameter λ that exhibits favorable behavior.

APPENDIX E: SUPPLEMENTAL NUMERICAL RESULTS

In the following, we first show another example that utilizes the first kind of symmetry-guided gradient descent c_1 to classify two-qubit bitstrings into two categories, which we call the cat-dog example. Then, we provide two two-parameter QNNs for the Werner state classification case in our main content, along with a detailed calculation process. We also present the experiment details for the 2D classification task.

1. Cat-dog example

a. Task formulation

Consider categorizing all pure quantum states under the computational basis into two categories, which means states corresponding bit strings with only 1s are labeled as cats while others as dogs. Then, we choose the QNN as $f(|\psi\rangle) = \mathrm{tr}(|\psi\rangle\langle\psi|\mathrm{Z})$, where Z is chosen as the observable as we only need to consider state amplitudes. Finally, we select 0 as the threshold to identify these two classes, which means states with $f(|\psi\rangle) \leqslant 0$ would be classified as cats while others as dogs.

This task could be considered from the view of classical logic circuits, where we process classical bit strings. These tasks are transformed into implementing logical expressions using logical circuits, such as $Y = \overline{A}B + A\overline{B}$ for the two-bit case, where Y is the logical circuit output, A and B are bits inputs, and $\overline{A}$ is the inverse of A. This means we can always find suitable quantum circuits to solve these tasks.

As only the number of 0s or 1s would influence their category, the general form of the symmetry group here could be formulated as

$$H_{\mathrm{cat}} = \left\{ \prod_{(i,j)} \mathrm{SWAP}_{ij} \,|\, i \neq j,\ 1,\ j = 1, 2, \ldots, n \right\}. \tag{E1}$$

b. Two-qubit case

We first consider the simplest $n = 2$ case. According to the two-bit case classical logical expression, which corresponds to the functional expression of an XOR gate, we could use circuits with only two qubits and detect the output value of the second qubit.

The classification task could be achieved by X gates and CNOT gates in two qubits since f is a binary function. Thus, the quantum circuit that achieves the task must consist of X gates and CNOT gates. We use $\mathrm{X}^{\cos(\theta)^2}$ and $\mathrm{CNOT}^{\cos(\theta)^2}$ to switch the gates. If $\theta = 0$, the gates are open, else if $\theta = \frac{\pi}{2}$, the gates are closed. The QNN we used here is

$$U(\theta, \phi) = (\mathrm{I} \otimes \mathrm{X})^{\cos^2(\phi)} \mathrm{CNOT}_{\mathrm{A}\to\mathrm{B}}^{\cos^2(\theta)}, \tag{E2}$$

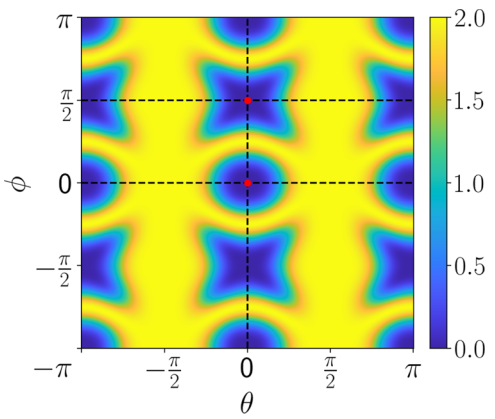


FIG. 7. Symmetry penalty term landscape of two-qubit cat-dog example. The highlighted region signifies the area subjected to a substantial penalty. The red dots are the minimum of the penalty term.

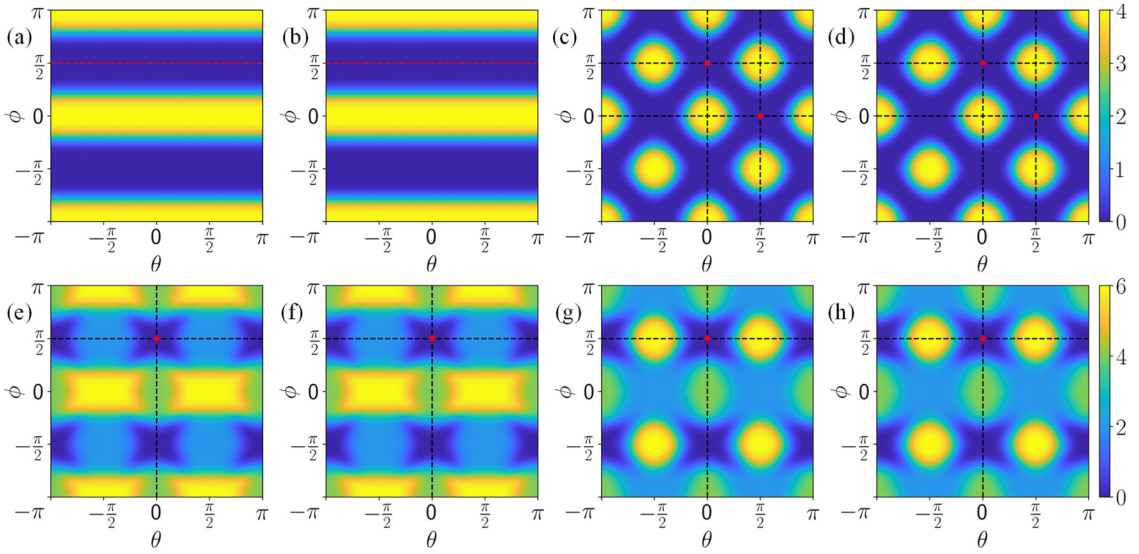


FIG. 8. Loss landscapes before and after adding symmetry penalty term under all inputs.

where A, B denotes the first and second qubit in our circuit, and $\rightarrow$ means A controls B. After the input state evolves by the QNN, we measure the second qubit in Z basis.

The symmetry group in this task is $H_1 = \{\text{SWAP}, \text{I}\}$ because the number of 1s is invariant under permutation. With the symmetry group, we could calculate the symmetry penalty term

$$g(\theta,\phi) = \|U^\dagger(\theta,\phi)Z_1U(\theta,\phi) - \mathcal{T}(U^\dagger(\theta,\phi)Z_1U(\theta,\phi))\|. \tag{E3}$$

The heat map of the symmetry penalty term $g(\theta,\phi)$ is shown in Fig. 7. Notice that $\cos^2(\theta) = \cos^2(\theta+\pi)$. There are two inequivalent minima $(\theta,\phi) = (0,\pi/2)$ and $(0,0)$ of the penalty term. If $|01\rangle$ and $|10\rangle$ are labeled 1, the optimal parameter is (0,0); else if $|01\rangle$ and $|10\rangle$ are labeled 0, the optimal parameter is $(0,\pi/2)$. Thus, the two minima of the symmetry penalty term correspond to the two labeling strategies.

Generally, there would be a discrepancy between theoretical and empirical errors, as losses or training would be biased under different training samples, which would then lead to overfitting. This could be verified by the heat maps in our case, as shown in Fig. 8, where we plot loss heat maps under different single inputs. In this case, under the default loss function, samples 00 and 01 would only train the ϕ parameter, while using either 10 or 11 would have two possible local minima, where one leads to the optimal point but another does not. But after adding a symmetry penalty term with a suitable guidance weight, we could see that all landscapes would have only one local minimum, and they all are exactly located at the optimal point. We summarize the above results in Table I.

This is an example showing that SG could help us deal with overfitting. As for more qubits cases, it would be hard to show improvement easily, and the circuit shows some properties that could be further investigated.

2. Entanglement classification

a. Task formulation and analysis

Let us consider the example using QNN for Werner state entanglement witness. Werner states could be classified into entangled or separable states according to a given parameter p,

$$\rho_{\text{W}} = \frac{2-p}{6}\mathbf{I} + \frac{2p-1}{6}\text{SWAP}. \tag{E4}$$

We formulate the task as finding a suitable observable that could detect whether a Werner state is separable or entangled according to its expectation value. In this task, we construct the objective function as $f(\rho) = \text{tr}(U(\theta)\rho U(\theta)^\dagger O)$. Then, we classify states according to their trace outputs, where we take states with $f(\rho) \leqslant 0$ as separable states and states with $f(\rho) < 0$ as entangled states.

Here, we work on the two-qubit case for illustration. A two-qubit Werner state could be constructed by $\rho_{\text{Werner}} = t\frac{I}{4} + (1-t)|\phi^-\rangle\langle\phi^-|$, where the factor t ranges from 0 to 4/3. 2/3 is the split point of two kinds of states when $t \leqslant 2/3$ provides separable states and $t > 2/3$ provides entangled states.

This time, the symmetry group should be the whole set of single-qubit unitary operators. We leverage tools provided in [49] (see Eq. (48) in Ref. [49]) to calculate the twirled observable. This corresponds to calculating the second moment of a given observable O. In the two-qubit case, $\mathcal{T}(O)$ could be

TABLE I. Comparison among different cost function settings for two-qubits cat-dog example.

	Symmetry penalty term	Loss function	Loss function + penalty term
Full Data	Half × Half ✓	✓	✓
Half Data	Half × Half ✓	Half × Half ✓	✓

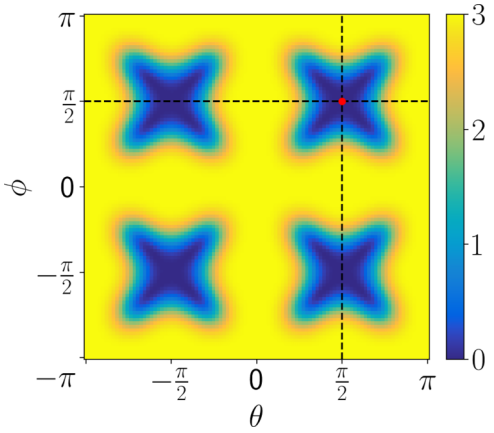


FIG. 9. Symmetry penalty term landscape of two-qubit Werner example 1.

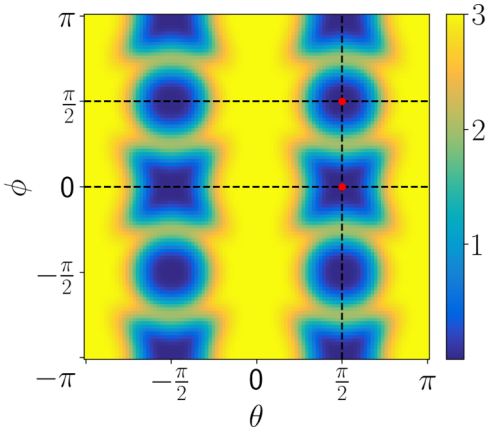


FIG. 11. Symmetry penalty term landscape of two-qubit Werner example 2.

calculated as

$$\mathcal{T}(O) = c_{\mathbb{I},O} I + c_{\mathbb{F},O} \mathbb{F}, \tag{E5}$$

where $\mathbb{F} = \text{SWAP}$, $c_{\mathbb{I},O} = \frac{\text{tr}(O) - 2^{-1}\text{tr}(\mathbb{F}O)}{2}$, and $c_{\mathbb{F},O} = \frac{\text{tr}(\mathbb{F}O) - 2^{-1}\text{tr}(O)}{3}$.

b. Two-qubit case 1

First, we try the circuit form $U(\theta, \phi) = \text{CNOT}_{\text{B}\to\text{A}}^{\cos^2(\phi)} \text{CNOT}_{\text{A}\to\text{B}}^{\cos^2(\theta)}$. Then, we could get the landscape of symmetry penalty term where only one local minimum exists, as shown in Fig. 9. These samples could be separated into four groups based on the shape of their loss landscapes, though different samples having similar shapes would show exactly the same values. We select one sample from each class and show their loss landscapes in Fig. 10. These four classes are entangled states ($0 \leqslant t < 2/3$ states), $t = 1$ state, separable states with $t < 1$, and separable states with $t > 1$. There is a split in separable states, as when gates in the given gates are not all off, the trace output would be a constant value of $1/2$.

From their loss landscapes, we could see that when using the default loss function, we need to select samples to train the model perfectly, as using samples with shapes in (c) and (d) would not lead to the overall optimal point. However, after adding the symmetry guidance term, we could train the model using any sample we like. This result is similar to the above two-qubit cat-dog case and shows that the symmetry penalty term could help biased samples less influence the optimization. But there is only one local minimum in this case,

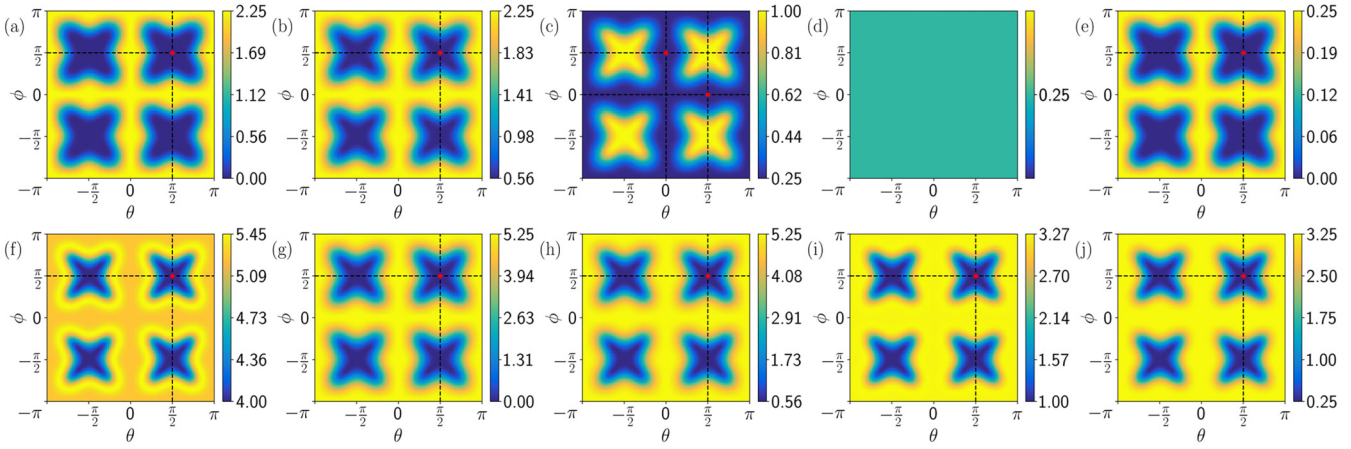


FIG. 10. Loss landscapes before and after adding symmetry penalty term under selected input samples.

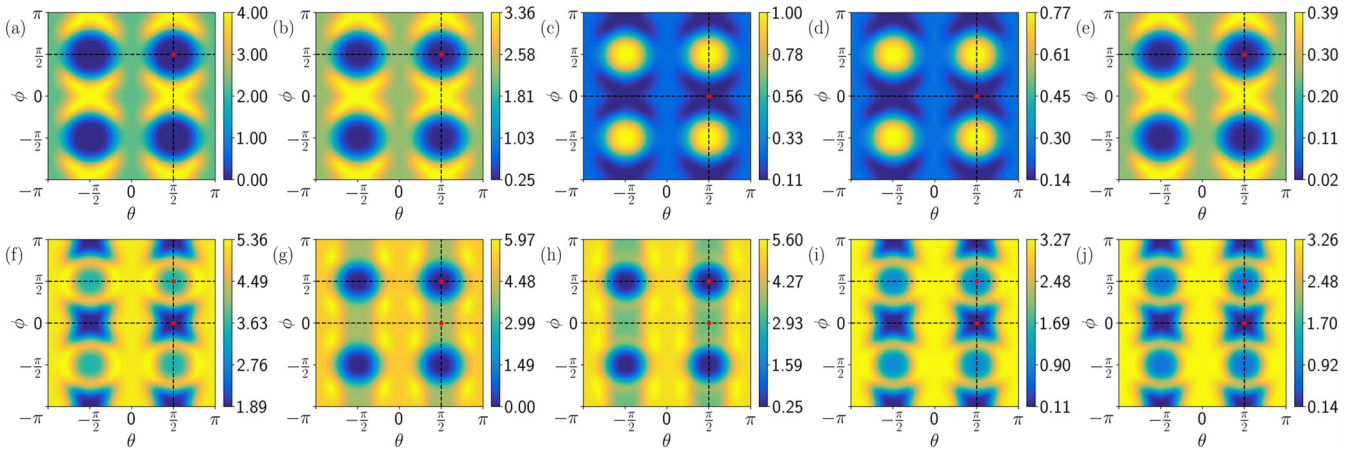


FIG. 12. Loss landscapes before and after adding symmetry penalty term under selected input samples.

which is rare in normal tasks, so we consider a more complex one in the following.

c. Two-qubit case 2

This time, we use the same circuit in the above cat-dog example. We find that the model would output $1-0.5t$ when a single IX gate turns on, while all other circuits expect the identity circuit to output a constant value of $1/2$.

The symmetry penalty term now changed to have two optimal points, as shown in Fig. 11, though we only want the $(\pi/2, \pi/2)$, or the all-gates-off one. Samples could also be categorized into four examples as above.

In this case, as samples and their loss landscapes are shown in Fig. 12, we can see that we still need to select samples for training perfectly in the default case. After adding the guidance term, though the improvement is not as significant as in the above two examples, new local minima were added for samples that would not train the model well in the default case. This means adding the guidance terms would help us alleviate sample-biased overfitting in a probabilistic way.

3. Circuit and experiment details for 2D classification

The circuit we used for 2D classical data classification is shown in Fig. 13.

The coding is implemented via the PENNYLANE library [50]. The gradient descent uses the ADAM optimizer [34] with a learning rate of 0.01 for both training with c_0 and c_2. Training data are randomly generated samples from each class, with 140 samples drawn from each area, as shown in the main text, Fig. 3(a). After the training, we evaluate the performances on samples uniformly selected from the whole data space. The final overall accuracy increased from $69.93 \pm 4.51\%$ to $89.62 \pm 4.61\%$ after adopting our c_2 function in 21 runs of experiments. The classification results are shown in the main text, Figs. 3(c) and 3(d). Codes and data for our numerical experiments can be found on Github [51].

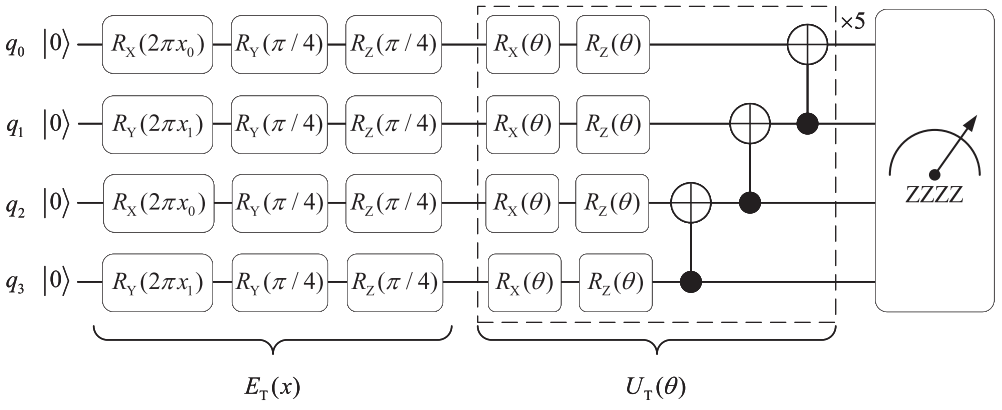


FIG. 13. QNN for 2D classical data classification.

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

[2] 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).

[3] W. Li, Z.-D. Lu, and D.-L. Deng, Quantum neural network classifiers: A tutorial, SciPost Phys. Lect. Notes 61 (2022).

[4] K. H. Wan, O. Dahlsten, H. Kristjánsson, R. Gardner, and M. Kim, Quantum generalisation of feedforward neural networks, npj Quantum Inf. **3**, 36 (2017).

[5] J. Romero, J. P. Olson, and A. Aspuru-Guzik, Quantum autoencoders for efficient compression of quantum data, Quantum Sci. Technol. **2**, 045001 (2017).

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

[7] 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).

[8] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, Parameterized quantum circuits as machine learning models, Quantum Sci. Technol. **4**, 043001 (2019).

[9] J. G. Vidal and D. O. Theis, Calculus on parameterized quantum circuits, arXiv:1812.06323.

[10] S. Sim, P. D. Johnson, and A. Aspuru-Guzik, Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms, Adv. Quantum Technol. **2**, 1900070 (2019).

[11] G. Verdon, T. McCourt, E. Luzhnica, V. Singh, S. Leichenauer, and J. Hidary, Quantum graph neural networks, arXiv:1909.12264.

[12] X. Ai, Z. Zhang, L. Sun, J. Yan, and E. Hancock, Towards quantum graph neural networks: An ego-graph learning approach, arXiv:2201.05158.

[13] S. Oh, J. Choi, and J. Kim, A tutorial on quantum convolutional neural networks (QCNN), in *Proceedings of the 2020 International Conference on Information and Communication Technology Convergence (ICTC)* (IEEE, New York, 2020), pp. 236–239.

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

[15] S. Y. C. Chen, C. M. Huang, C. W. Hsing, and Y. J. Kao, Hybrid quantum-classical classifier based on tensor network and variational quantum circuit, arXiv:2011.14651.

[16] Z.-Z. Sun, C. Peng, D. Liu, S.-J. Ran, and G. Su, Generative tensor network classification model for supervised machine learning, Phys. Rev. B **101**, 075135 (2020).

[17] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, Geometric deep learning: Going beyond euclidean data, IEEE Signal Process. Mag. **34**, 18 (2017).

[18] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, arXiv:2104.13478.

[19] 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).

[20] F. M. Sbahi, A. J. Martinez, S. Patel, D. Saberi, J. H. Yoo, G. Roeder, and G. Verdon, Provably efficient variational generative modeling of quantum many-body systems via quantum-probabilistic information geometry, arXiv:2206.04663.

[21] P. Mernyei, K. Meichanetzidis, and I. I. Ceylan, Equivariant quantum graph circuits, in *Proceedings of the International Conference on Machine Learning* (PMLR, Maryland, 2022), pp. 15401–15420.

[22] 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).

[23] C. Lyu, X. Xu, M.-H. Yung, and A. Bayat, Symmetry enhanced variational quantum spin eigensolver, Quantum **7**, 899 (2023).

[24] K. Seki, T. Shirakawa, and S. Yunoki, Symmetry-adapted variational quantum eigensolver, Phys. Rev. A **101**, 052340 (2020).

[25] R. F. Werner, Quantum states with Einstein-Podolsky-Rosen correlations admitting a hidden-variable model, Phys. Rev. A **40**, 4277 (1989).

[26] M. Schuld and F. Petruccione, *Machine Learning with Quantum Computers* (Springer, Berlin, 2021).

[27] M. Weigold, J. Barzen, F. Leymann, and M. Salm, Data encoding patterns for quantum computing, in *Proceedings of the 27th Conference on Pattern Languages of Programs* (The Hillside Group, USA, 2020), pp. 1–11.

[28] W. Bi, H. Li, and J. Huang, Data augmentation for text generation without any augmented data, arXiv:2105.13650.

[29] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio *et al.*, Variational quantum algorithms, Nat. Rev. Phys. **3**, 625 (2021).

[30] S. Endo, Z. Cai, S. C. Benjamin, and X. Yuan, Hybrid quantum-classical algorithms and quantum error mitigation, J. Phys. Soc. Jpn. **90**, 032001 (2021).

[31] M. Schuld and F. Petruccione, *Supervised Learning with Quantum Computers* (Springer, Berlin, 2018), Vol. 17.

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

[33] F. Rossi, P. Van Beek, and T. Walsh, *Handbook of Constraint Programming* (Elsevier, Amsterdam, 2006).

[34] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.

[35] G. Tóth and J. J. García-Ripoll, Efficient algorithm for multiqudit twirling for ensemble quantum computation, Phys. Rev. A **75**, 042311 (2007).

[36] W. Dür and J. I. Cirac, Classification of multiqubit mixed states: Separability and distillability properties, Phys. Rev. A **61**, 042314 (2000).

[37] X. Gao and L.-M. Duan, Efficient representation of quantum many-body states with deep neural networks, Nat. Commun. **8**, 662 (2017).

[38] R. Unanyan, H. Kampermann, and D. Bruß, A decomposition of separable Werner states, J. Phys. A: Math. Theor. **40**, F483 (2007).

[39] T. Hubregtsen, J. Pichlmeier, P. Stecher, and K. Bertels, Evaluation of parameterized quantum circuits: On the relation between classification accuracy, expressibility, and entangling capability, Quantum Mach. Intell. **3**, 9 (2021).

[40] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets, Nature (London) **549**, 242 (2017).

[41] P. D. Faris, W. A. Ghali, R. Brant, C. M. Norris, P. D. Galbraith, M. L. Knudtson, A. Investigators *et al.*, Multiple imputation

versus data enhancement for dealing with missing data in observational health care outcome analyses, J. Clin. Epidemiol. **55**, 184 (2002).

[42] B. R. Frieden, Image enhancement and restoration, in *Picture Processing and Digital Filtering*, edited by T. S. Huang, Topics in Applied Physics Vol. 6 (Springer, Berlin, Heidelberg, 1979).

[43] S. Bubeck *et al.*, Convex optimization: Algorithms and complexity, Found. Trends Mach. Learn. **8**, 231 (2015).

[44] A. D. Belegundu and T. R. Chandrupatla, *Optimization Concepts and Applications in Engineering* (Cambridge University Press, Cambridge, 2019).

[45] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, On the surprising behavior of distance metrics in high dimensional space, in *Proceedings of the 8th International Conference: Database Theory–ICDT 2001* (Springer, Berlin, 2001), Vol. 8, pp. 420–434.

[46] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver, A quantum engineer's guide to superconducting qubits, Appl. Phys. Rev. **6**, 021318 (2019).

[47] H.-L. Huang, D. Wu, D. Fan, and X. Zhu, Superconducting quantum computing: A review, Sci. China Inf. Sci. **63**, 180501 (2020).

[48] P. I. Etingof, O. Golberg, S. Hensel, T. Liu, A. Schwendner, D. Vaintrob, and E. Yudovina, *Introduction to Representation Theory* (AMS, Providence, 2011), Vol. 59.

[49] A. A. Mele, Introduction to Haar measure tools in quantum information: A Beginner's tutorial, Quantum **8**, 1340 (2024).

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

[51] https://github.com/Fragecity/SymmetryQNN