Quantum machine learning for multiclass classification beyond kernel methods

Chao Ding [1,2] Shi Wang [1,*] Yaonan Wang,[1] and Weibo Gao [2,3,4,5,†]

[1]*College of Electrical and Information Engineering, Hunan University, Changsha 410082, China*

[2]*Division of Physics and Applied Physics, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore 637371, Singapore*

[3]*School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore*

[4]*Centre for Quantum Technologies, National University of Singapore, Singapore 117543, Singapore*

[5]*National Centre for Advanced Integrated Photonics (NCAIP) Singapore, Nanyang Technological University, Singapore 637371, Singapore*

(Received 16 June 2024; revised 8 April 2025; accepted 28 April 2025; published 6 June 2025)

Quantum machine learning is considered one of the current research fields with great potential. In recent years, Havlíček *et al.* [Nature (London) **567**, 209 (2019)] have proposed a quantum machine learning algorithm with quantum-enhanced feature spaces, which effectively addressed a binary classification problem and offered a potential pathway to achieving quantum advantage. However, a straightforward binary classification algorithm falls short in solving multiclass classification problems. In this paper, we propose a quantum algorithm that rigorously demonstrates that quantum kernel methods enhance the efficiency of multiclass classification in real-world applications, providing quantum-enhanced performance. To demonstrate this, we design six distinct quantum kernels within the quantum algorithm to map input data into quantum state spaces and estimate the corresponding quantum kernel matrices. The results from quantum simulations reveal that the quantum algorithm outperforms its classical counterpart in handling six real-world multiclass classification problems. Furthermore, we leverage a variety of performance metrics to comprehensively evaluate the classification and generalization performance of the quantum algorithm. The results demonstrate that the quantum algorithm achieves superior classification and better generalization performance relative to classical counterparts.

DOI: 10.1103/PhysRevA.111.062410

I. INTRODUCTION

Quantum machine learning (QML), as one of the key applications of quantum computing in recent years [1–3], seeks to explore the fundamental limits of data analysis as permitted by the laws of physics. In particular, QML leverages unique quantum mechanical properties—superposition and entanglement—by encoding data in the inherently high-dimensional structure of quantum state spaces [4], offering a potential pathway beyond classical paradigms. As the number of qubits continues to increase, the demands on quantum hardware and technology also grow, making the design of QML algorithms more complex. In the era of noisy intermediate-scale quantum (NISQ) technology, a range of innovative methods have been developed to address the limitations imposed by constrained quantum hardware resources [5–15]. Most of these methods are centered on the clever integration of parameterized quantum circuits with classical paradigms to achieve quantum-enhanced performance. The resulting quantum-classical hybrid approach is thus considered one of the most promising avenues toward realizing such enhancement.

Supervised learning, as a classical paradigm in machine learning, aims to learn the correlation between input feature vectors and corresponding output class labels [16]. Interestingly, this learning paradigm can address many real-world machine learning tasks, from identifying diseases in medical diagnostics [17–20] to predicting molecular properties in drug discovery [21–25]. The quantum-classical hybrid approach, as an emerging and transformative learning paradigm, leverages parameterized quantum circuits to explore the intricate relationship between inputs and outputs [4,26–31]. Among these methods, those based on quantum kernels have attracted attention for implicitly embedding classical data into exponentially high-dimensional feature spaces using parameterized quantum circuits [4,26,28–30]. This advantage holds promise for achieving more expressive classification boundaries than classical kernel methods in real-world classification tasks.

A support vector machine (SVM) is a well-known supervised learning algorithm designed to find the optimal hyperplane that separates feature vectors into two distinct classes in a feature space [32–34]. It typically utilizes kernel methods [35–37] to implicitly map feature vectors into a higher-dimensional space, making it well suited for addressing classification problems that are not linearly separable. Some previous quantum-classical hybrid approaches to support vector machines [4,26,28], which explore the connection between quantum kernels and classical support vector machines, have made strides on straightforward binary classification problems. Fundamentally, these methods first perform a nonlinear mapping of feature vectors to quantum states through specialized parameterized quantum circuits.

*Contact author: shi_wang@hnu.edu.cn

†Contact author: wbgao@ntu.edu.sg

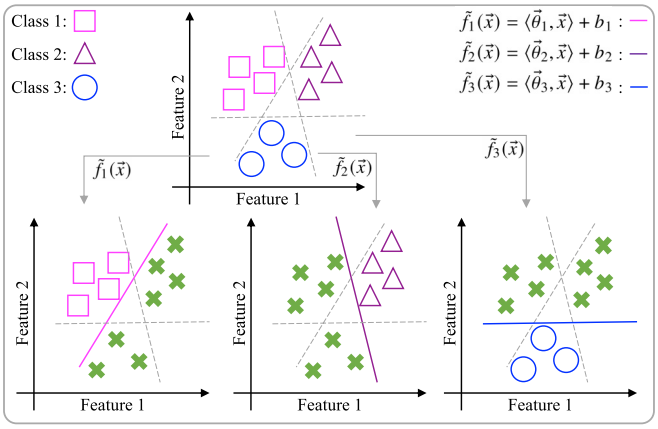


FIG. 1. Illustration of multiclass classification using a one-versus-all approach.

Following this mapping, they evaluate the overlap of pairwise quantum states to construct a kernel matrix. This matrix is then fed into a classical optimizer, which determines the optimal hyperplane by executing a convex quadratic program, effectively dividing the vectors into two classes. Although existing research has predominantly addressed binary classification tasks, multiclass classification problems are more common in real-world applications and pose greater challenges in terms of quantum kernel design, classification effectiveness, and generalization performance.

In this paper, we propose a quantum-enhanced multiclass SVM (QEM-SVM) that leverages the quantum state space [4] as its feature space to tackle real-world multiclass classification tasks and demonstrate quantum-enhanced performance. First, we ensure that quantum kernels, devised through specific parameterized quantum circuits, meet the required standards. In addition, quantum kernel matrices must satisfy the condition of being positive semidefinite matrices. Building on this foundation, we utilize the unique capabilities

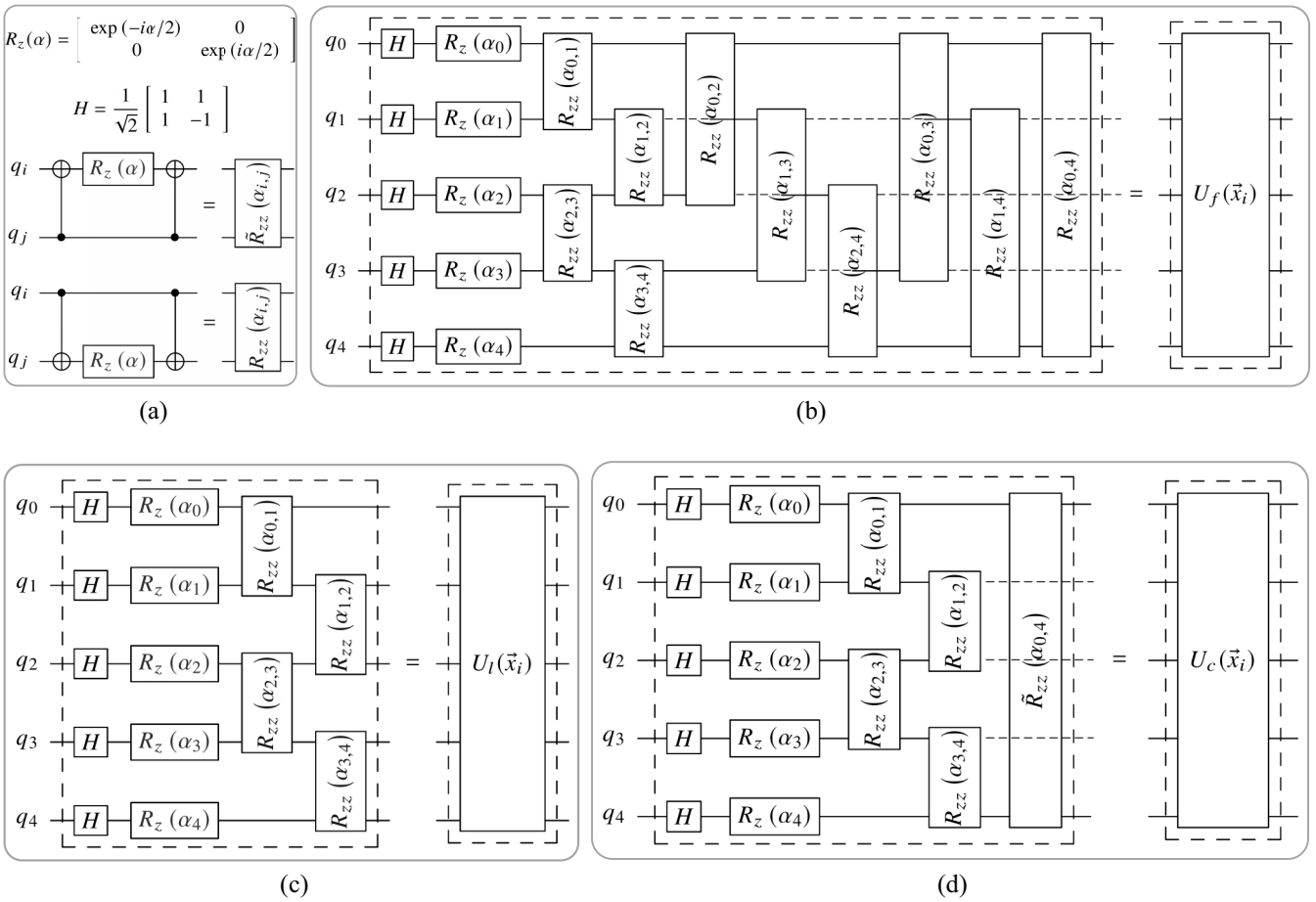


FIG. 2. Quantum circuit example with five qubits in the IQP circuit. (a) The matrix representation of single-qubit gates and the implementation of two-qubit gates. The two-qubit gates are constructed through CNOT and RZ gates. (b) The IQP circuit with a full entanglement layer. For a full entanglement layer with N qubits, where each qubit q_i is entangled with every other qubit q_j, there are a total of $N(N-1)/2$ two-qubit gates and $N(N+1)/2$ training parameters. (c) The IQP circuit with a linear entanglement layer. For a linear entanglement layer with N qubits, where the qubit q_i is entangled with the qubit q_{i+1} and the index i belongs to the set $\{0, 1, \ldots, N-2\}$, there are a total of $N-1$ two-qubit gates and $2N-1$ training parameters. (d) The IQP circuit with a circular entanglement layer. For a circular entanglement layer with N qubits, the qubits are entangled in the same way as in the linear entanglement layer, but with an additional entanglement between the qubit q_0 and the qubit q_{N-1}. There are a total of N two-qubit gates and $2N$ training parameters.

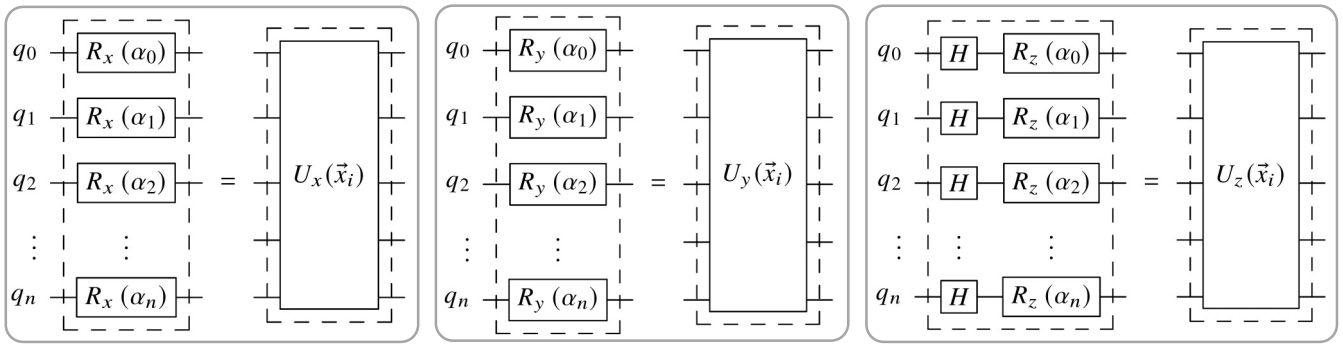


FIG. 3. Quantum circuit example with a trainable single-qubit rotation layer. Each qubit corresponds to each feature of the input classical state $\vec{x}_i$. (a) The trainable single-qubit rotation layer with Pauli-X rotations. A Pauli-X rotation transforms $|0\rangle$ into the quantum state $|\phi_x(x_t)\rangle = \cos(x_t/2)|0\rangle - i\sin(x_t/2)|1\rangle$. This rotation results in a superposition state of $|0\rangle$ and $|1\rangle$, significantly changing the probability amplitude distribution and introducing a phase difference between two basis states. (b) The trainable single-qubit rotation layer with Pauli-Y rotations [46]. A Pauli-Y rotation transforms $|0\rangle$ into the quantum state $|\phi_y(x_t)\rangle = \cos(x_t/2)|0\rangle + \sin(x_t/2)|1\rangle$. Similar to the Pauli-$X$ rotation, this rotation converts $|0\rangle$ into a superposition state of $|0\rangle$ and $|1\rangle$, altering the probability amplitude distribution of the two basis states and introducing a phase difference between them. (c) The trainable single-qubit rotation layer with Pauli-Z rotations. A Hadamard gate and Pauli-Z rotation transform $|0\rangle$ into the quantum state $|\phi_z(x_t)\rangle = \sqrt{2}/2[\cos(x_t/2) - i\sin(x_t/2)]|0\rangle + \sqrt{2}/2[\cos(x_t/2) + i\sin(x_t/2)]|1\rangle$. As with the Pauli-$X$ and Pauli-Y rotations, this operation generates a quantum state where the phase and probability amplitude distribution are both affected.

of high-dimensional data representation provided by instantaneous quantum polynomial circuits [26] to develop three types of quantum kernels: full, linear, and circular. In addition, we utilize parameterized quantum circuits featuring trainable single-qubit rotation layers to develop three types of quantum kernels: Pauli X, Pauli Y, and Pauli Z. To compare the performance of quantum algorithms with various quantum kernels, we introduce six distinct real-world datasets, each with unique feature dimensions and class labels. The results demonstrate that the optimal quantum kernel is significantly contingent upon the distribution and structure of the real-world dataset. Furthermore, a comprehensive suite of performance metrics is employed to evaluate the quantum algorithm. The results demonstrate that the quantum algorithm, when equipped with the optimal quantum kernel, surpasses its classical counterparts in both classification accuracy and generalization performance. Finally, we examine the effects of exponential concentration and hardware noise on the quantum algorithm. The findings reveal no observable impact from exponential concentration, while depolarizing noise affects the algorithm to varying degrees.

The paper is organized as follows: Sec. II presents classical multiclass SVMs and quantum kernel estimation. Section III elaborates on six distinct quantum kernels, and Sec. IV proposes a quantum machine learning algorithm termed QEM-SVM. Section V provides a comprehensive performance analysis of the QEM-SVM, and Sec. VI introduces the effects of exponential concentration and hardware noise on the QEM-SVM. The paper concludes with Sec. VII, encapsulating our findings.

II. PRELIMINARIES

A. Notations

The notations used in this paper are outlined as follows. The dot product is represented as $\langle \vec{a}, \vec{b}\rangle = \vec{a}\cdot\vec{b}$. Consider two quantum states, $|\tau\rangle$ and $|\phi\rangle$, where the inner product is denoted by $\langle\tau|\phi\rangle$, the outer product by $|\tau\rangle\langle\phi|$, and the overlap is

TABLE I. Examples of kernel function representations for input feature vectors $\vec{x}_i$ and $\vec{x}_j$, including classical kernels and quantum kernels.

Type	Name	Kernel function	Hyperparameters
Classical [35]	Linear kernel (LK)	$\kappa_(\vec{x}_i, \vec{x}_j) = \langle \vec{x}_i, \vec{x}_j\rangle$	None
	Polynomial kernel (PK)	$\kappa(\vec{x}_i, \vec{x}_j) = (\varkappa\langle \vec{x}_i, \vec{x}_j\rangle)^{\mathrm{d}}$	$\varkappa > 0$, $\mathrm{d} \geqslant 1$
	Sigmoid kernel (SK)	$\kappa(\vec{x}_i, \vec{x}_j) = \tanh(\varkappa\langle \vec{x}_i, \vec{x}_j\rangle + c)$	$\varkappa > 0$, $c < 0$
	Gaussian kernel (GK)	$\kappa(\vec{x}_i, \vec{x}_j) = \exp(-\varkappa\cdot\|\vec{x}_i - \vec{x}_j\|^2)$	$\varkappa > 0$
Quantum	Full quantum kernel (FQK)	$\kappa(\vec{x}_i, \vec{x}_j) = \mathrm{Tr}[(\|0\rangle\langle 0\|)^{\otimes N}(U_f^\dagger(\vec{x}_j)U_f(\vec{x}_i)(\|0\rangle\langle 0\|)^{\otimes N}U_f^\dagger(\vec{x}_i)U_f(\vec{x}_j))]$	None
	Linear quantum kernel (LQK)	$\kappa(\vec{x}_i, \vec{x}_j) = \mathrm{Tr}[(\|0\rangle\langle 0\|)^{\otimes N}(U_l^\dagger(\vec{x}_j)U_l(\vec{x}_i)(\|0\rangle\langle 0\|)^{\otimes N}U_l^\dagger(\vec{x}_i)U_l(\vec{x}_j))]$	None
	Circular quantum kernel (CQK)	$\kappa(\vec{x}_i, \vec{x}_j) = \mathrm{Tr}[(\|0\rangle\langle 0\|)^{\otimes N}(U_c^\dagger(\vec{x}_j)U_c(\vec{x}_i)(\|0\rangle\langle 0\|)^{\otimes N}U_c^\dagger(\vec{x}_i)U_c(\vec{x}_j))]$	None
	Pauli-X quantum kernel (XQK)	$\kappa(\vec{x}_i, \vec{x}_j) = \mathrm{Tr}[(\|0\rangle\langle 0\|)^{\otimes N}(U_x^\dagger(\vec{x}_j)U_x(\vec{x}_i)(\|0\rangle\langle 0\|)^{\otimes N}U_x^\dagger(\vec{x}_i)U_x(\vec{x}_j))]$	None
	Pauli-Y quantum kernel (YQK)	$\kappa(\vec{x}_i, \vec{x}_j) = \mathrm{Tr}[(\|0\rangle\langle 0\|)^{\otimes N}(U_y^\dagger(\vec{x}_j)U_y(\vec{x}_i)(\|0\rangle\langle 0\|)^{\otimes N}U_y^\dagger(\vec{x}_i)U_y(\vec{x}_j))]$	None
	Pauli-Z quantum kernel (ZQK)	$\kappa(\vec{x}_i, \vec{x}_j) = \mathrm{Tr}[(\|0\rangle\langle 0\|)^{\otimes N}(U_z^\dagger(\vec{x}_j)U_z(\vec{x}_i)(\|0\rangle\langle 0\|)^{\otimes N}U_z^\dagger(\vec{x}_i)U_z(\vec{x}_j))]$	None

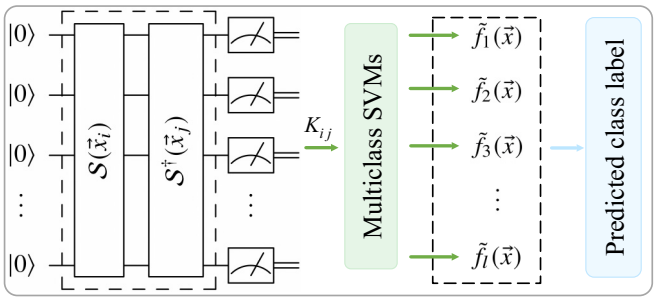


FIG. 4. Schematic diagram of the QEM-SVM, constructed by integrating quantum kernels with classical multiclass SVMs, where $K_{ij} = |\langle 0|^{\otimes N}\mathcal{S}^\dagger(\vec{x}_j)\mathcal{S}(\vec{x}_i)|0\rangle^{\otimes N}|^2$. The quantum feature mapping $\mathcal{S}(\vec{x}_i)$ is constrained to elements of the set $\{U_f(\vec{x}_i), U_l(\vec{x}_i), U_c(\vec{x}_i), U_x(\vec{x}_i), U_y(\vec{x}_i), U_z(\vec{x}_i)\}$.

defined as $|\langle\tau|\phi\rangle|^2$. In addition, we define

$$\delta_{i,p} = \begin{cases} 1, & \text{if } y_i = p \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

where both i and p serve as indices. Finally, three canonical rotations around the axes, corresponding to the Pauli matrices $(\sigma_x, \sigma_y, \sigma_z)$, are denoted as

$$R_x(2\alpha) = \sigma_0 \cos\alpha - i\sigma_x \sin\alpha, \tag{2}$$

$$R_y(2\alpha) = \sigma_0 \cos\alpha - i\sigma_y \sin\alpha, \tag{3}$$

$$R_z(2\alpha) = \sigma_0 \cos\alpha - i\sigma_z \sin\alpha, \tag{4}$$

where

$$\sigma_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad \sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{5}$$

The above summarizes the key notations used in this paper.

B. Multiclass SVMs

One approach to address multiclass classification is by decomposing it into multiple binary classification tasks [38–40]. Multiclass SVMs typically adopt a one-versus-all approach [41–45], illustrated in Fig. 1, wherein three binary classifiers are employed to address the classification task involving three classes. Suppose that there are a set of data points $\{(\vec{x}_i, y_i) : \vec{x}_i \in \mathbb{R}^N, y_i \in \mathbb{Y}\}_{i=1,\ldots,m}$, where $\mathbb{Y} = \{1, \ldots, l\}$, y_i is the class label, and $\vec{x}_i$ is the feature vector. The task of the sth binary classifier is to differentiate data points into two classes: the sth class, which is assigned a positive label, and the combined class of all remaining classes, which receive a negative label. For the sth binary classifier, there exists a normal vector $\vec{\theta}_s \in \mathbb{R}^N$ and a bias term $b_s \in \mathbb{R}$. In this case, we define the decision function $\tilde{f}_s(\vec{x}) = \langle\vec{\theta}_s, \vec{x}\rangle + b_s$, where $\vec{x}$ is the feature vector of unknown data points.

To achieve optimal classification performance, it is essential to address the following optimization problem:

$$\min_{\vec{\theta},b,\vec{\xi}} L(\vec{\theta}, b, \vec{\xi}) = \frac{1}{2}\langle\vec{\theta}_s, \vec{\theta}_s\rangle + C\sum_{i=1}^{m} \xi_i^s, \tag{6}$$

subject to the constraints: $y_i[\langle\vec{\theta}_s, \vec{x}_i\rangle + b_s] \geqslant 1 - \xi_i^s$, $y_i = \pm 1$, $\xi_i^s \geqslant 0$, and $\forall i = 1, \ldots, m$. The dual formulation to this optimization problem can be expressed as

$$\max_{\vec{\alpha}} L(\vec{\alpha}) = \sum_{i=1}^{m} \alpha_i^s - \frac{1}{2}\sum_{i,j} \alpha_i^s \alpha_j^s y_i y_j \langle\vec{x}_i, \vec{x}_j\rangle, \tag{7}$$

TABLE II. Real-world datasets used in experimental result analysis.

Dataset	No. instances	No. features	No. class
Iris [52]	150	4	3
Tae [53]	151	5	3
Penguin [54]	344	5	3
Glass [55]	214	9	6
Ecoli [56]	336	7	8
Vowel [57]	528	10	11

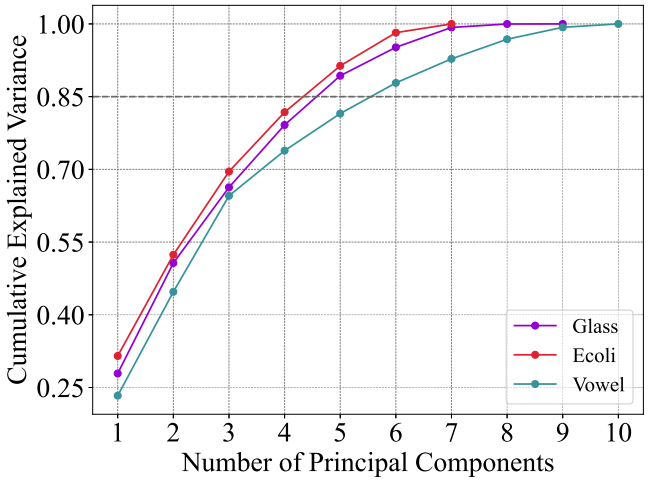


FIG. 5. Cumulative explained variance analysis.

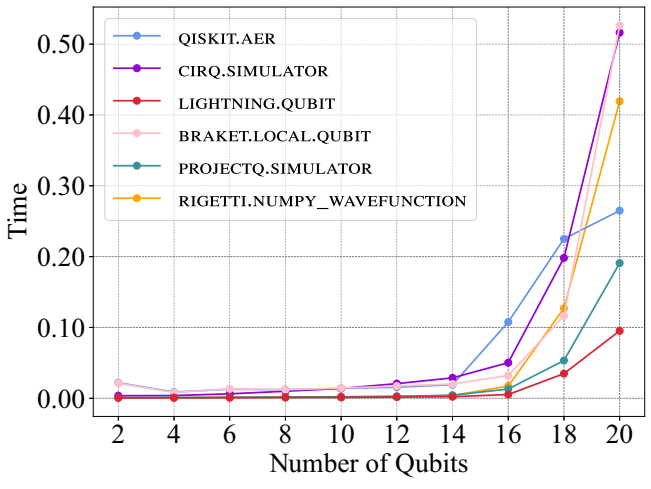


FIG. 6. Runtime comparison of different quantum simulators.

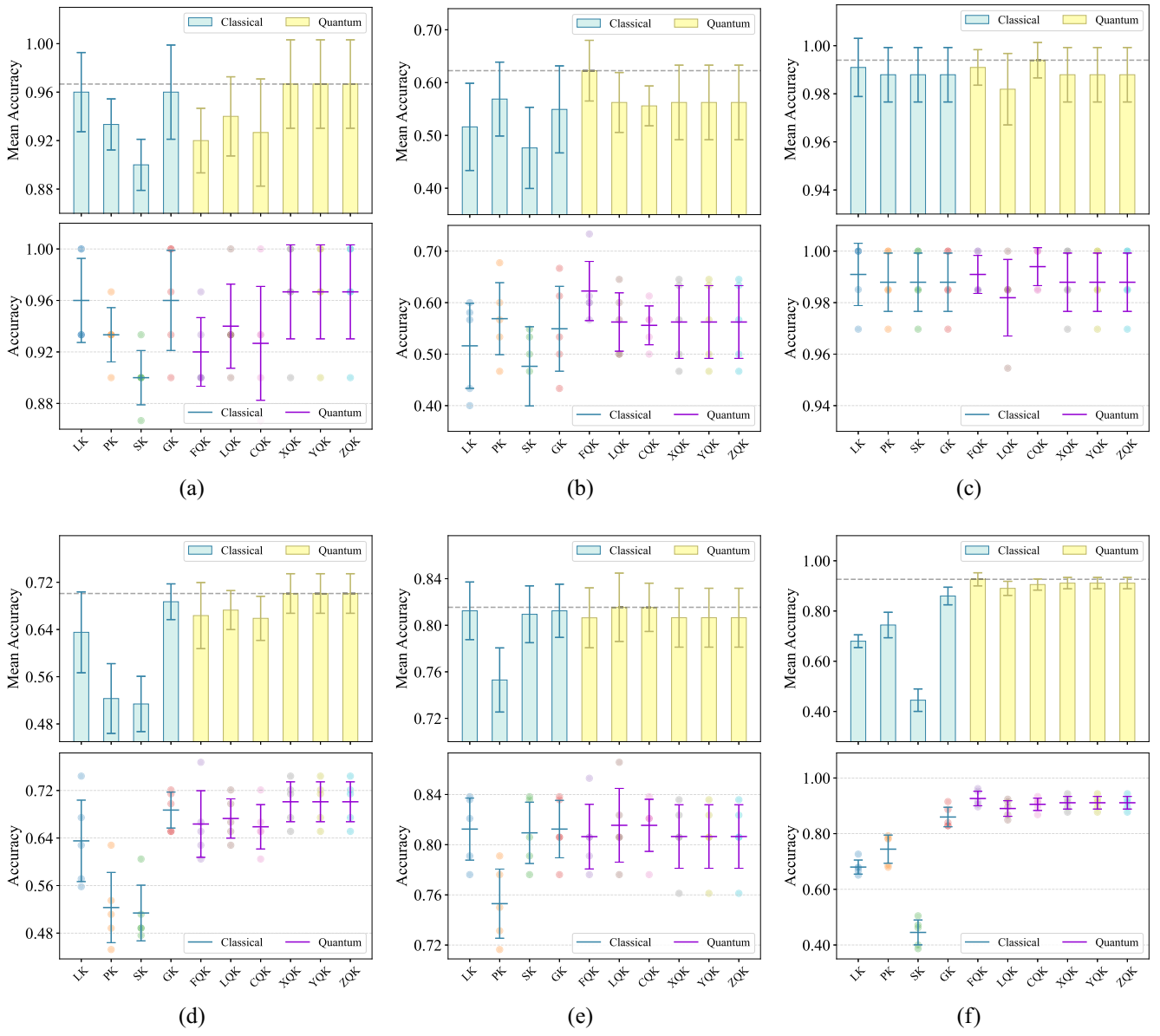


FIG. 7. Performance comparison across datasets: (a) Iris, (b) Tae, (c) Penguin, (d) Glass, (e) Ecoli, and (f) Vowel. The bar chart shows the mean accuracy comparison between classical and quantum kernels, and the scatter plot illustrates the accuracy distribution for each cross-validation iteration.

subject to the constraints $C \geqslant \alpha_i^s \geqslant 0$, and $\sum_{i=1}^m y_i \alpha_i^s = 0$. The Lagrange multipliers α_i^s here are used to determine support vectors, and the index set of the support vectors is denoted by Ω. Therefore, the decision function of the sth binary classifier is redefined as $\tilde{f}_s(\vec{x}) = \sum_{i\in\Omega} \alpha_i^s y_i \langle \vec{x}_i, \vec{x}\rangle + b_s$. As there are l binary classifiers, the corresponding l decision functions are expressed as follows: $\tilde{f}_1(\vec{x}) = \langle \vec{\theta}_1, \vec{x}\rangle + b_1$, …, $\tilde{f}_l(\vec{x}) = \langle \vec{\theta}_l, \vec{x}\rangle + b_l$. This yields the resulting decision function:

$$\tilde{f}(\vec{x}) = \arg \max_{s=1,\ldots,l} [\langle \vec{\theta}_s, \vec{x}\rangle + b_s]. \tag{8}$$

The class with the highest decision function is selected.

C. Quantum kernel estimation

Suppose that a feature vector $\vec{x}$ is mapped to a quantum state $|\phi(\vec{x})\rangle = \mathcal{S}(\vec{x})|0\rangle^{\otimes N}$, the density matrix of $|\phi(\vec{x})\rangle$ is represented as $\rho(\vec{x}) = |\phi(\vec{x})\rangle\langle\phi(\vec{x})| \in \mathbb{C}^{2^N\times 2^N}$. For $|\phi(\vec{x}_i)\rangle$ and $|\phi(\vec{x})\rangle$, we define the quantum kernel [46–50] using their overlap, i.e., $\kappa(\vec{x}_i, \vec{x}) = |\langle\phi(\vec{x}_i)|\phi(\vec{x})\rangle|^2$. Given a set of data points $\{(\vec{x}_i, y_i) : \vec{x}_i \in \mathbb{R}^N, y_i \in \mathbb{Y}\}_{i=0,\ldots,N-1}$, we generalize the quantum kernel $\kappa(\vec{x}_i, \vec{x}_j)$ to a quantum kernel matrix $K \in \mathbb{R}^{N\times N}$, with the individual elements given by $K_{ij} = |\langle\phi(\vec{x}_i)|\phi(\vec{x}_j)\rangle|^2$. Therefore, we simulate and compute each element K_{ij} of the quantum kernel matrix K on an NISQ device. Furthermore, in accordance with Mercer condition [51], it is imperative to establish that the quantum kernel matrix K satisfies the criteria of being a positive semidefinite matrix. Here we demonstrate

TABLE III. Data description following a 70–30 training-testing split.

	Dataset					
	Iris	Tae	Penguin	Glass	Ecoli	Vowel
N_{train}	105	105	233	149	235	369
N_{test}	45	46	100	65	101	159

that the quantum kernel $\kappa(\vec{x}_i, \vec{x}_j)$ is an admissible kernel based on the straightforward inference from $K_{ij} = K_{ji}$ and $K_{ij} \geqslant 0$.

III. QUANTUM KERNELS WITH LEARNABLE ROTATIONS

In this section, we develop six quantum kernels for multiclass classification, each featuring learnable rotations and built from distinct parameterized quantum circuits.

A. Full, linear, and circular quantum kernels

To give rise to quantum kernels, we utilize an instantaneous quantum polynomial (IQP) circuit [26]. The IQP circuit includes a Hadamard layer, a single-qubit rotation layer, and a two-qubit entanglement layer, with its performance influenced by the quantum circuit structure. To explore this, we examine three distinct quantum circuit structures in the IQP circuit: full, linear, and circular entanglement layers, illustrated in Figs. 2(b)–2(d), respectively. Furthermore, three unique quantum circuit structures yield three distinct quantum kernels: full, linear, and circular. Quantum kernels map the classical input state $\vec{x}_i = (x_0, x_1, \ldots, x_{N-1})^T \in \mathbb{R}^N$ to a quantum state space.

For the full quantum kernel, $\vec{x}_i$ is mapped to $|\phi_f(\vec{x}_i)\rangle = U_f(\vec{x}_i)|0\rangle^{\otimes N}$, where $|0\rangle^{\otimes N}$ is the initial state. As illustrated in Fig. 2(b), implementing a Hadamard gate on each qubit yields the following state: $|\phi_1\rangle = (1/\sqrt{2})^N \sum_{q_0=0}^{1} \sum_{q_1=0}^{1} \cdots \sum_{q_{N-1}=0}^{1} |q_0, q_1, \ldots, q_{N-1}\rangle$, where q_i describes the state of the ith qubit, which can be either 0 or 1. Due to $q = q_{N-1}2^0 + q_{N-2}2^1 + \cdots + q_0 2^{N-1}$, $|\phi_1\rangle$ simplifies to $(1/\sqrt{2})^N \sum_{q=0}^{2^N-1} |q\rangle$. Based on $R_z(\alpha_i)|0\rangle = |0\rangle$ and $R_z(\alpha_i)|1\rangle = e^{i\alpha_i}|1\rangle$, we apply the Pauli-$Z$ rotation gate to each qubit to obtain $|\phi_2\rangle = (1/\sqrt{2})^N \sum_{q=0}^{2^N-1} \exp(i \sum_{i=0}^{N-1} \alpha_i q_i)|q\rangle$. There are two two-qubit gates, $R_{zz}(\alpha)$ and $\tilde{R}_{zz}(\alpha)$, in Fig. 2(a), where $R_{zz}(\alpha) = \tilde{R}_{zz}(\alpha) = \text{diag}(e^{-i\alpha/2}, e^{i\alpha/2}, e^{i\alpha/2}, e^{-i\alpha/2})$ and $\alpha = \alpha_{i,j} = \alpha_i \alpha_j$. Based on $R_{zz}(\alpha)|00\rangle = e^{-i\alpha/2}|00\rangle$, $R_{zz}(\alpha)|01\rangle = e^{i\alpha/2}|01\rangle$, $R_{zz}(\alpha)|11\rangle = e^{-i\alpha/2}|11\rangle$, and $R_{zz}(\alpha)|10\rangle = e^{i\alpha/2}|10\rangle$, we apply the two-qubit gates to the quantum circuit given in Fig. 2(b) to obtain

$$|\phi_3\rangle = \left(\frac{1}{\sqrt{2}}\right)^N \sum_{q=0}^{2^N-1} \exp\left(i \sum_{i=0}^{N-1} \alpha_i q_i\right) \prod_{(i,j)\in I} F_{(i,j)}|q\rangle \quad (9)$$

with $F_{(i,j)} = \exp(i\frac{-\alpha_i\alpha_j}{2}(-1)^{q_i \oplus q_j})$ and $I = \{(i, j) \mid 0 \leqslant i < j \leqslant N-1\}$. Here the feature x_i corresponds to α_i, and $x_i x_j$ corresponds to $\alpha_i \alpha_j$. The final quantum state $|\phi_f(\vec{x}_i)\rangle$ is expressed as $(1/\sqrt{2})^N \sum_{q=0}^{2^N-1} \exp(i \sum_{i=0}^{N-1} x_i q_i) \prod_{(i,j)\in I} F'_{(i,j)}|q\rangle$ with $F'_{(i,j)} = \exp(i\frac{-x_i x_j}{2}(-1)^{q_i \oplus q_j}]$. For the linear quantum kernel, the feature $\vec{x}_i$ is mapped to $|\phi_l(\vec{x}_i)\rangle = U_l(\vec{x}_i)|0\rangle^{\otimes N}$. Similar to the full quantum kernel, the resulting quantum state $|\phi_l(\vec{x}_i)\rangle$ is defined by $(1/\sqrt{2})^N \sum_{q=0}^{2^N-1} \exp(i \sum_{i=0}^{N-1} x_i q_i) \prod_{(i,j)\in \bar{I}} F'_{(i,j)}|q\rangle$ with $\bar{I} = \{(i, j) \mid 0 \leqslant i < j \leqslant N-1, j = i+1\}$. For the circular quantum kernel, the feature $\vec{x}_i$ is mapped to $|\phi_c(\vec{x}_i)\rangle = U_c(\vec{x}_i)|0\rangle^{\otimes N}$. Similar to the full and linear quantum kernels, the obtained quantum state $|\phi_c(\vec{x}_i)\rangle$ is denoted as $(1/\sqrt{2})^N \sum_{q=0}^{2^N-1} \exp(i \sum_{i=0}^{N-1} x_i q_i) \prod_{(i,j)\in \hat{I}} F'_{(i,j)}|q\rangle$ with $\hat{I} = \{(i, j) \mid 0 \leqslant i \leqslant N-1, j = (i+1) \mod N\}$. The above are the three types of quantum kernels generated by the IQP circuit.

B. Pauli-X, Pauli-Y, and Pauli-Z quantum kernels

In addition to IQP circuits, we utilize a parameterized quantum circuit featuring a trainable single-qubit rotation layer to implement quantum kernels. In terms of the single-qubit rotation layer, we employ three distinct quantum circuit structures: Pauli-X, -Y, and -Z rotations, illustrated in Figs. 3(a)–3(c), respectively. Similarly, these quantum circuit structures give rise to three unique quantum kernels: Pauli X, Pauli Y, and Pauli Z. For the Pauli-X quantum kernel, the feature $\vec{x}_i$ is mapped to $|\phi_x(\vec{x}_i)\rangle = U_x(\vec{x}_i)|0\rangle^{\otimes N}$, where $U_x(\vec{x}_i) = R_x(x_0) \otimes \cdots \otimes R_x(x_{N-1})$. Therefore, the quantum state $|\phi_x(\vec{x}_i)\rangle$ is reformulated as

$$|\phi_x(\vec{x}_i)\rangle = \sum_{q_0=0}^{1} \cdots \sum_{q_{N-1}=0}^{1} \prod_{i=0}^{N-1} \left(\cos\frac{x_i}{2}\right)^{1-q_i} \left(-i\sin\frac{x_i}{2}\right)^{q_i} |q\rangle. \quad (10)$$

For the Pauli-Y quantum kernel, the feature $\vec{x}_i$ is mapped to $|\phi_y(\vec{x}_i)\rangle = U_y(\vec{x}_i)|0\rangle^{\otimes N}$, where $U_y(\vec{x}_i) = R_y(x_0) \otimes \cdots \otimes R_y(x_{N-1})$. Similar to Eq. (10), this results in the following

TABLE IV. Performance analysis of the proposed quantum algorithm using optimal kernels across various real-world datasets.

		Macroaverage			Microaverage			Weighted average			
Dataset	Kernel function	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Accuracy
Iris	XQK, YQK, ZQK	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Tae	FQK	0.5926	0.5875	0.5827	0.5870	0.5870	0.5870	0.5919	0.5870	0.5822	0.5870
Penguin	CQK	0.9932	0.9815	0.9870	0.9900	0.9900	0.9900	0.9902	0.9900	0.9899	0.9900
Glass	XQK, YQK, ZQK	0.7357	0.6562	0.6796	0.7692	0.7692	0.7692	0.7451	0.7692	0.7406	0.7692
Ecoli	LQK	0.7584	0.6925	0.7141	0.8515	0.8515	0.8515	0.8423	0.8515	0.8437	0.8515
Vowel	FQK	0.9299	0.9331	0.9267	0.9308	0.9308	0.9308	0.9355	0.9308	0.9278	0.9308

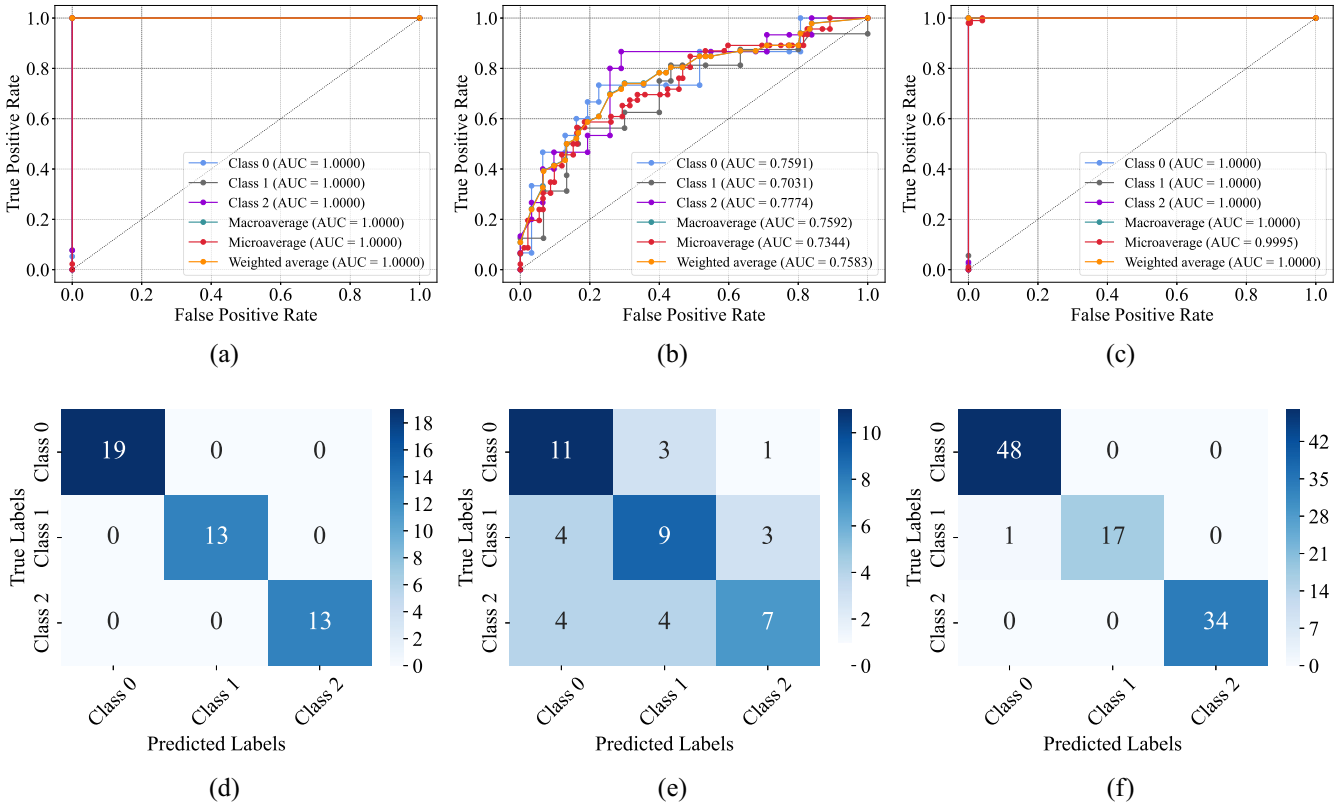


FIG. 8. ROC curve and confusion matrix for each dataset: (a), (d) for the Iris dataset, (b), (e) for the Tae dataset, and (c), (f) for the Penguin dataset. The ROC curve illustrates the trade-off between true positive rate (TPR) and false positive rate (FPR) for each class, and the confusion matrix shows classification accuracy, highlighting correct and incorrect predictions among classes.

quantum state:

$$|\phi_y(\vec{x}_i)\rangle = \sum_{q_0=0}^{1} \cdots \sum_{q_{N-1}=0}^{1} \prod_{i=0}^{N-1} \left(\cos\frac{x_i}{2}\right)^{1-q_i} \left(\sin\frac{x_i}{2}\right)^{q_i} |q\rangle. \tag{11}$$

For the Pauli-Z quantum kernel, we observe that Pauli-Z rotations affect only the phase of the quantum state. Therefore, we introduce a Hadamard layer before the Pauli-Z rotation layer to optimize this quantum feature mapping. As illustrated in Fig. 3(c), the feature $\vec{x}_i$ is mapped to $|\phi_z(\vec{x}_i)\rangle = U_z(\vec{x}_i)|0\rangle^{\otimes N}$, where $U_z(\vec{x}_i) = R_z(x_0)H \otimes \cdots \otimes R_z(x_{N-1})H$. The Pauli-$Z$ rotation gate is represented as $R_z(x_i) = e^{-ix_i/2}|0\rangle\langle 0| + e^{ix_i/2}|1\rangle\langle 1|$. Thus, the quantum state $|\phi_z(\vec{x}_i)\rangle$ has the form

$$|\phi_z(\vec{x}_i)\rangle = \left(\frac{1}{\sqrt{2}}\right)^N \sum_{q=0}^{2^N-1} \exp\left(i\sum_{i=0}^{N-1} x_i q_i\right)|q\rangle. \tag{12}$$

Note that the quantum kernels we used for multiclass classification are detailed in Table I.

IV. QEM-SVM

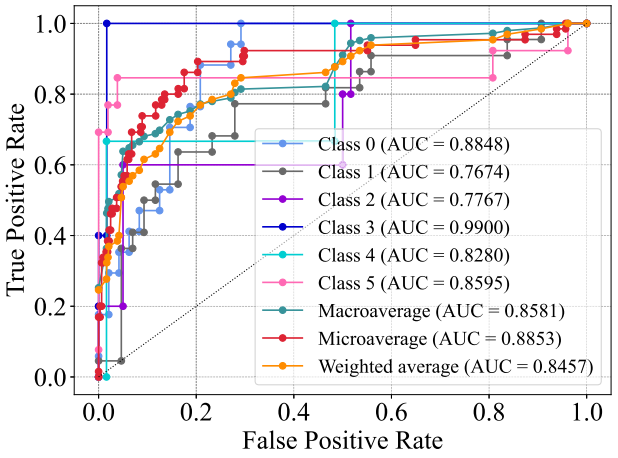
In this section, we provide a detailed description of QEM-SVM. The QEM-SVM includes two main stages, as illustrated in Fig. 4. In the first stage, it calculates the quantum kernel matrix on an NISQ device to evaluate the similarity between quantum states. This includes performing multiple measurements on a parameterized quantum circuit to derive the elements of the quantum kernel matrix. In the following stage, the quantum kernel matrix is utilized to formulate and solve a quadratic programming problem [32] for multiclass classification. This includes generalizing the quadratic programming problem presented in Eq. (6) to the following form:

$$\min_{\vec{\theta},b,\vec{\xi}} \tilde{L}(\vec{\theta}, b, \vec{\xi}) = \frac{1}{2}\sum_{s=1}^{l}\langle\vec{\theta}_s, \vec{\theta}_s\rangle + C\sum_{i=1}^{m}\sum_{s\neq y_i}^{l}\xi_i^s, \tag{13}$$

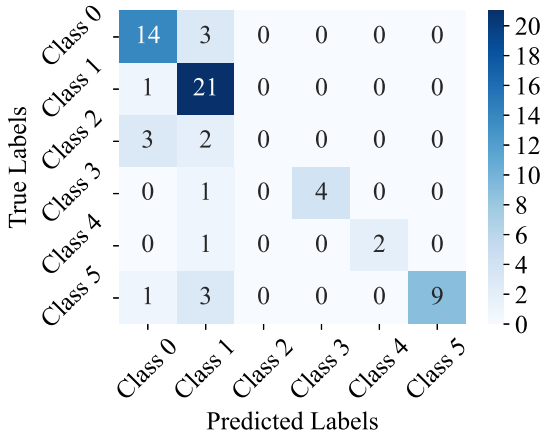
with the following constraints: $\xi_i^s \geqslant 0$, $\langle\vec{\theta}_{y_i}, \vec{x}_i\rangle + b_{y_i} \geqslant \langle\vec{\theta}_s, \vec{x}_i\rangle + b_s + 2 - \xi_i^s$, $\forall s \neq y_i$, and $\forall i = 1, \ldots, m$. In addition, ξ_i^s and C are slack variables and penalty parameters, respectively. To address this quadratic programming problem, we formulate the following generalized Lagrangian function:

$$\begin{aligned}\tilde{L} = &-\sum_{i,s}\alpha_i^s\left[\langle\vec{\theta}_{y_i} - \vec{\theta}_s, \vec{x}_i\rangle + b_{y_i} - b_s - 2 + \xi_i^s\right] \\ &-\sum_{i,s}\mu_i^s\xi_i^s + \frac{1}{2}\sum_{s=1}^{l}\langle\vec{\theta}_s, \vec{\theta}_s\rangle + C\sum_{i,s}\xi_i^s,\end{aligned} \tag{14}$$

with the following constraints: α_i^s, μ_i^s, $\xi_i^s \geqslant 0$ and $\forall s \neq y_i$, where α_i^s and μ_i^s are Lagrange multipliers. Additionally, we have $\alpha_i^{y_i} = 0$, $\mu_i^{y_i} = 0$, and $\xi_i^{y_i} = 2$. Then, the optimal

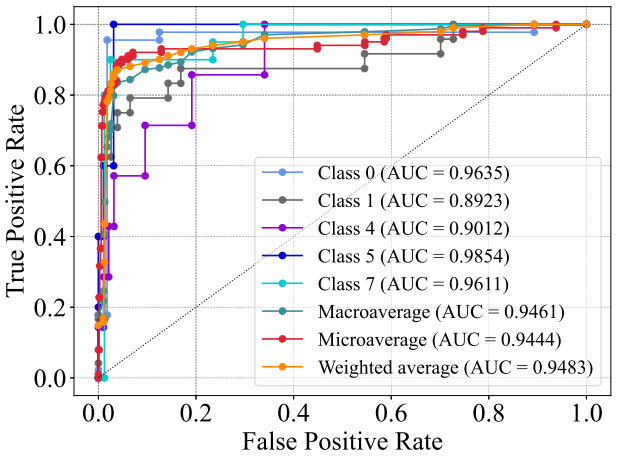


(a)

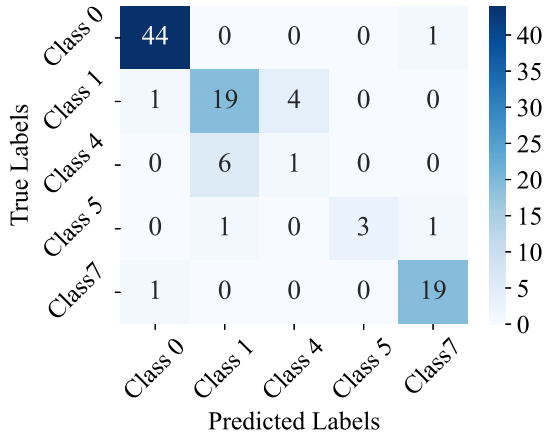


(b)

FIG. 9. ROC curve (a) and confusion matrix (b) for the Glass dataset.



(a)



(b)

FIG. 10. ROC curve (a) and confusion matrix (b) for the Ecoli dataset.

parameter set can be obtained through min-max optimization:

$$\arg\max_{\vec{\alpha},\vec{\mu}} \quad \arg\min_{\vec{\theta},b,\vec{\xi}} \quad \tilde{L}. \tag{15}$$

The partial derivatives of Eq. (14) with respect to the primal variables $(\vec{\theta}_{s'}, b_{s'}, \xi_i^{s'})$ are denoted by $\partial\tilde{L}/\partial\vec{\theta}_{s'} = \sum_{i=1}^{m} \alpha_i^{s'}\vec{x}_i - \sum_{i,s} \alpha_i^s \delta_{i,s'}\vec{x}_i + \vec{\theta}_{s'}$, $\partial\tilde{L}/\partial b_{s'} = \sum_{i=1}^{m} \alpha_i^{s'} - \sum_{i,s} \alpha_i^s \delta_{i,s'}$, and $\partial\tilde{L}/\partial\xi_i^{s'} = C - \alpha_i^{s'} - \mu_i^{s'}$. Finally, the resulting decision function is given by

$$\tilde{f}(\vec{x},\vec{\alpha}) = \arg\max_{s'} \sum_{i=1}^{m} \left(\delta_{i,s'} \sum_{s=1}^{l} \alpha_i^s - \alpha_i^{s'} \right) \kappa(\vec{x}_i, \vec{x}) + b_{s'}. \tag{16}$$

Based on Eq. (16), we determine the output by calculating the values of each decision function and selecting the class with the highest value. The core derivation of Eq. (16) can be found in Appendix B.

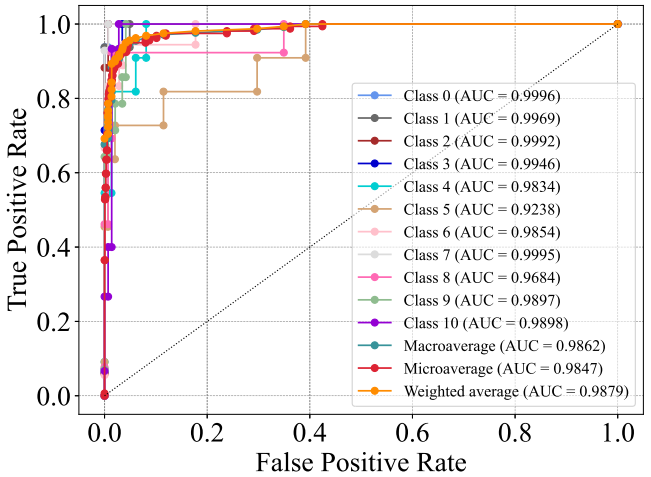


FIG. 11. ROC curve for Vowel dataset. The AUCs all exceed 0.9000.

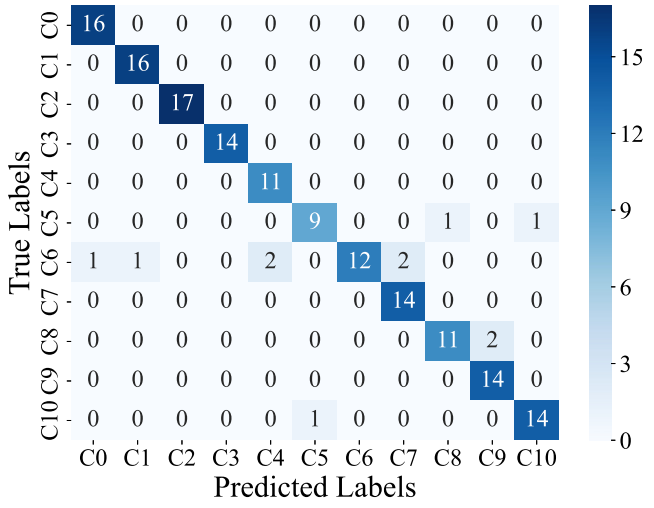


FIG. 12. Confusion matrix for Vowel dataset. C0–C10, Class 0–10.

V. EXPERIMENTS

A. Experimental datasets and simulation platforms

In this part, we introduce the real-world datasets and simulation platforms; details of the datasets are provided in Table II. Notably, the Penguin dataset includes 11 instances with missing values, which have been removed prior to the experiment. Then, we apply z-score normalization [58] to standardize the six real-world datasets during a preprocessing stage. For the latter three datasets, we utilize the cumulative explained variance in principal component analysis (PCA) [59–61] to select principal components. Analysis of Fig. 5 indicates that achieving 85% cumulative explained variance requires five principal components for the Glass and Ecoli datasets, and six principal components for the Vowel dataset. Therefore, features are adjusted from 9 to 5, 7 to 5, and 10 to 6 for the Glass, Ecoli, and Vowel datasets, respectively.

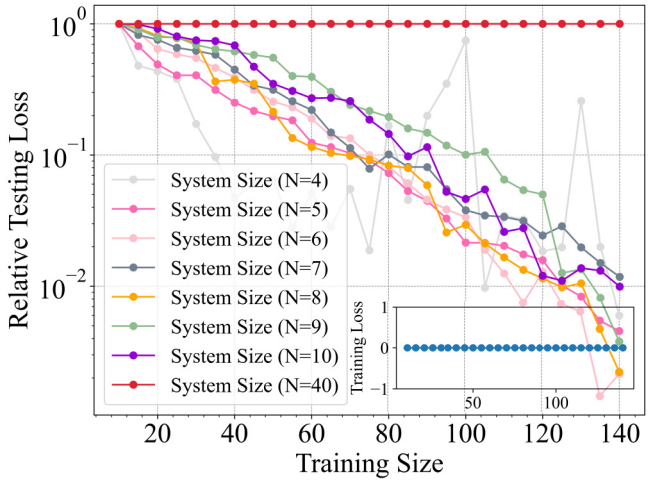


FIG. 13. Effect of exponential concentration on generalization performance. In alignment with the parameters in Ref. [71], we adopt the following settings: 150 data points, 1000 measurement shots, and 20 testing data points. The random seed is set to 15. Each data point is randomly generated in the interval $[0, 2\pi)$. The main plot illustrates how the relative testing loss changes with increasing training size, while the inset shows the variation in absolute training loss as the training size increases.

To validate feasibility and efficiency, the proposed quantum algorithm is implemented with a quantum simulator and applied to the six real-world multiclass datasets. Here are some well-known quantum simulators from various providers: QISKIT.AER [62], CIRQ.SIMULATOR [63], LIGHTNING.QUBIT [64], BRAKET.LOCAL.QUBIT [65], PROJECTQ.SIMULATOR [66], and RIGETTI.NUMPY_WAVEFUNCTION [67]. As illustrated in Fig. 6, the LIGHTNING.QUBIT quantum simulator outperforms other quantum simulators in speed. Therefore, all noise-free quantum simulations for training and testing in this paper are performed using the LIGHTNING.QUBIT quantum simulator. Moreover, noisy quantum simulations are conducted with the DEFAULT.MIXED quantum simulator.

B. Classification performance analysis

The classification performance analysis of the proposed quantum algorithm are detailed in this part, which includes two stages. First, we analyze and compare the performance of various kernels across six different multiclass real-world datasets to choose the optimal kernel for each dataset. Second, we assess the performance of the quantum algorithm—using the optimal kernel for each dataset—based on a set of performance metrics, which are detailed in Appendix A.

In the first stage, we employ a stratified k-fold cross-validation method [68–70] to evaluate the proposed quantum algorithm. For each dataset, we divide it into five folds, ensuring that the class label proportions in each fold are consistent with those in the entire dataset.

Based on the comparative analysis in Fig. 7, we obtain the optimal kernel for each dataset. For the Iris and Glass datasets, the Pauli-X, Pauli-Y, and Pauli-Z quantum kernels perform equally well, with the proof of their equivalence provided in the Appendix D. For the Tae and Vowel datasets, the full quantum kernel is optimal. The Penguin dataset benefits most from the circular quantum kernel, and the Ecoli dataset is best served by the linear quantum kernel. Therefore, the optimal quantum kernel is significantly contingent upon the distribution and structure of real-world datasets. In addition, the proposed quantum algorithm with the optimal quantum kernel exhibits superior classification performance compared to its classical counterparts across all six real-world datasets.

In the following stage, we split each dataset into a 70–30 training and testing set, as shown in Table III. We then train and test the proposed quantum algorithm with optimal kernels. Analysis of Table IV reveals that the accuracy values are consistent with the precision, recall, and F1 score values computed using microaverage methods. Moreover, they also match the recall values derived from weighted average methods. Therefore, these five performance metrics serve the same function in evaluating the quantum algorithm.

Further analysis indicates that the quantum algorithm achieves satisfactory classification accuracy, along with robust precision, recall, and F1 score across macroaverage,

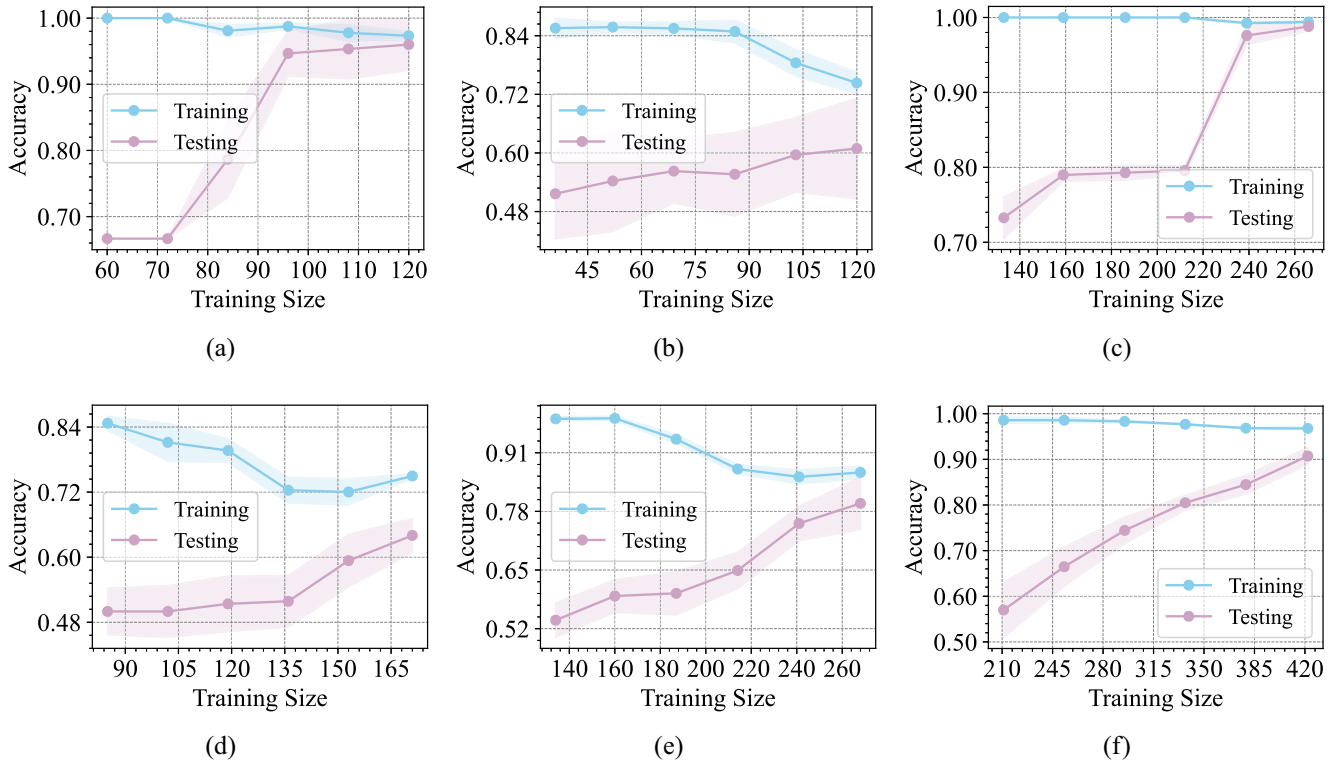


FIG. 14. Learning curves for the proposed quantum algorithm utilizing optimal quantum kernels on the following datasets: (a) Iris, (b) Tae, (c) Penguin, (d) Glass, (e) Ecoli, and (f) Vowel. The learning curve of training illustrates how the training accuracy evolves as the training size increases, and the learning curve of testing depicts the variation in testing accuracy with the growing training size. The blue and purple shaded areas represent the variability in training and testing accuracy, respectively. The generalization performance of the quantum algorithm is assessed on six real-world datasets using a stratified fivefold cross-validation method.

microaverage, and weighted average evaluations. Specifically, the quantum algorithm achieves a perfect classification accuracy of 100% for the Iris dataset and an impressive 99.00% accuracy for the Penguin dataset. It also demonstrates strong performance on the Vowel dataset with an accuracy of 93.08% and the Ecoli dataset with 85.15%. For the Glass dataset, it achieves an accuracy of 76.92%, while for the Tae dataset, it attains a satisfactory accuracy of 58.70%.

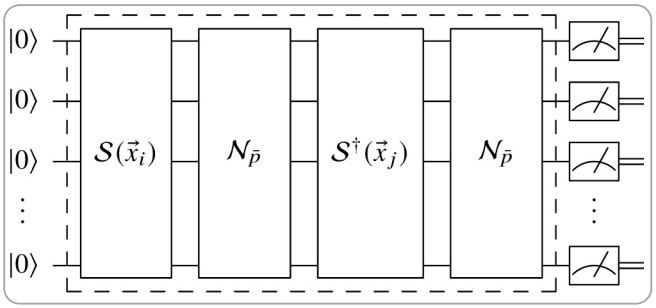


FIG. 15. Parameterized quantum circuit to generate noisy quantum kernels for feature vectors $\vec{x}_i$ and $\vec{x}_j$. The implementation of the circuit consists of the following steps: first, applying the unitary operation $\mathcal{S}(\vec{x}_i)$, followed by a depolarizing noise channel $\mathcal{N}_{\bar{p}}$. Then the inverse operation $\mathcal{S}^\dagger(\vec{x}_j)$ is performed, followed again by a depolarizing noise channel. Finally, a projector Π is employed to measure the resulting quantum state.

To evaluate the proposed quantum algorithm with optimal quantum kernels, we also employ receiver operating characteristic (ROC) curves and confusion matrices. Drawing from a comprehensive analysis of Figs. 8–12, we evaluate the classification performance of quantum algorithms employing optimal quantum kernels across various classes in each dataset, as well as the overall performance using macroaverage, microaverage, and weighted average methods.

For the Iris dataset, the area under the ROC curve (AUC) for each class, as well as the macroaverage, microaverage, and weighted average AUCs, are all 1.0000, demonstrating that the quantum algorithm achieves perfect classification. For the Tae dataset, Class 2 exhibits the strongest classification performance, while Class 1 is relatively weaker; the macroaverage, microaverage, and weighted average AUCs confirm robust overall performance. For the Penguin dataset, the AUC for each class, as well as the macroaverage, microaverage, and weighted average AUCs, are either 1.0000 or close to 1.0000, indicating near-perfect classification performance. For the Glass dataset, Class 3 shows the highest performance, while Class 1 is the lowest; the macroaverage, microaverage, and weighted average AUCs reflect solid overall performance. For the Ecoli dataset, Class 5 achieves the best classification performance, while Class 1 is the worst; the macroaverage, microaverage, and weighted average

TABLE V. Examples of representing noisy quantum kernel functions for input feature vectors $\vec{x}_i$ and $\vec{x}_j$.

Name	Kernel function	Hyperparameters
Noisy full quantum kernel (NFQK)	$\kappa'(\vec{x}_i,\vec{x}_j)=\mathrm{Tr}[(\lvert 0\rangle\langle 0\rvert)^{\otimes N}(\mathcal{N}_{\bar{p}}\circ U_f^{\dagger}(\vec{x}_j)\circ\mathcal{N}_{\bar{p}}\circ U_f(\vec{x}_i))(\lvert 0\rangle\langle 0\rvert)^{\otimes N}]$	$\bar{p}\in[0,1]$
Noisy linear quantum kernel (NLQK)	$\kappa'(\vec{x}_i,\vec{x}_j)=\mathrm{Tr}[(\lvert 0\rangle\langle 0\rvert)^{\otimes N}(\mathcal{N}_{\bar{p}}\circ U_l^{\dagger}(\vec{x}_j)\circ\mathcal{N}_{\bar{p}}\circ U_l(\vec{x}_i))(\lvert 0\rangle\langle 0\rvert)^{\otimes N}]$	$\bar{p}\in[0,1]$
Noisy circular quantum kernel (NCQK)	$\kappa'(\vec{x}_i,\vec{x}_j)=\mathrm{Tr}[(\lvert 0\rangle\langle 0\rvert)^{\otimes N}(\mathcal{N}_{\bar{p}}\circ U_c^{\dagger}(\vec{x}_j)\circ\mathcal{N}_{\bar{p}}\circ U_c(\vec{x}_i))(\lvert 0\rangle\langle 0\rvert)^{\otimes N}]$	$\bar{p}\in[0,1]$
Noisy Pauli-X quantum kernel (NXQK)	$\kappa'(\vec{x}_i,\vec{x}_j)=\mathrm{Tr}[(\lvert 0\rangle\langle 0\rvert)^{\otimes N}(\mathcal{N}_{\bar{p}}\circ U_x^{\dagger}(\vec{x}_j)\circ\mathcal{N}_{\bar{p}}\circ U_x(\vec{x}_i))(\lvert 0\rangle\langle 0\rvert)^{\otimes N}]$	$\bar{p}\in[0,1]$
Noisy Pauli-Y quantum kernel (NYQK)	$\kappa'(\vec{x}_i,\vec{x}_j)=\mathrm{Tr}[(\lvert 0\rangle\langle 0\rvert)^{\otimes N}(\mathcal{N}_{\bar{p}}\circ U_y^{\dagger}(\vec{x}_j)\circ\mathcal{N}_{\bar{p}}\circ U_y(\vec{x}_i))(\lvert 0\rangle\langle 0\rvert)^{\otimes N}]$	$\bar{p}\in[0,1]$
Noisy Pauli-Z quantum kernel (NZQK)	$\kappa'(\vec{x}_i,\vec{x}_j)=\mathrm{Tr}[(\lvert 0\rangle\langle 0\rvert)^{\otimes N}(\mathcal{N}_{\bar{p}}\circ U_z^{\dagger}(\vec{x}_j)\circ\mathcal{N}_{\bar{p}}\circ U_z(\vec{x}_i))(\lvert 0\rangle\langle 0\rvert)^{\otimes N}]$	$\bar{p}\in[0,1]$

AUCs demonstrate strong overall performance. Finally, for the Vowel dataset, Class 0 exhibits the strongest classification performance, while Class 5 is the weakest; the macroaverage, microaverage, and weighted average AUCs indicate near-optimal overall performance. Interestingly, as illustrated in Fig. 10, the quantum algorithm classifies only five classes in the Ecoli dataset, despite it containing eight classes. This is due to a class imbalance in the dataset, with Classes 2, 3, and 6 having insufficient instances, resulting in their exclusion from prediction.

C. Generalization performance analysis

To rigorously evaluate the generalization performance of QEM-SVM, we focus on deriving generalization error bounds for quantum kernels. As noted in Ref. [2], we derive an upper bound on the empirical Rademacher complexity to obtain a generalization error bound for quantum kernels. In other words, investigating the empirical Rademacher complexity $\hat{\mathfrak{R}}_{\mathrm{E}}(\mathcal{F})$ of quantum kernels provides insights into the generalization capabilities of QEM-SVM. As shown in Appendix C, the quantum kernels used in this work exhibit a lower Frobenius norm and empirical Rademacher complexity compared to classical kernels across all datasets. As a result, the findings indicate that QEM-SVM outperforms its classical counterparts in terms of generalization performance. A detailed derivation of $\hat{\mathfrak{R}}_{\mathrm{E}}(\mathcal{F})$ is presented in Appendix C.

VI. IN-DEPTH PERFORMANCE ANALYSIS

Up to now, we only consider the ideal, noise-free scenarios, focusing on the evaluation of parameterized quantum circuits for quantum kernels under perfect scenarios. In fact, when implementing quantum kernels on quantum devices, we also need to consider the effects of exponential concentration and hardware noise on the quantum kernel. In this section, we first evaluate the effect of exponential concentration on the performance of the proposed quantum algorithm, followed by an assessment of the effect of hardware noise.

A. Effect of exponential concentration

Exponential concentration in quantum kernel methods captures the effect where, as the problem size grows, the kernel values between data points mapped by quantum feature mapping progressively approach a fixed constant [71]. Due to the inherent uncertainty of measurements on quantum devices, we must conduct repeated measurements to ensure an accurate estimation of the quantum kernel. Therefore, more measurement shots are essential to distinguish between different kernel values. However, under the constraints of a polynomial number of measurement shots, exponential concentration reduces the distinguishability of kernel values, preventing the proposed quantum algorithm from effectively capturing the patterns of the input data. That is, the trained quantum algorithm may make incorrect predictions on unseen inputs, resulting in poor generalization performance.

To assess the effect of exponential concentration on generalization performance, we employ a relative testing loss ϑ relative to its initial values ($N_{\mathrm{init}}=10$). As stated in Ref. [71], if $N_{\mathrm{init}}>10$ and $\vartheta<1$, it signifies superior generalization with larger training size. Figure 13 depicts the effects of exponential concentration on four-qubit to ten-qubit simulations, with the solid red line representing the 40-qubit simulation that achieves exponential concentration as detailed in Ref. [71]. It is found from Fig. 13 that for simulations ranging from four to ten qubits, the condition $N_{\mathrm{init}}>10$ and $\vartheta<1$ holds, with generalization performance progressively improving as the training size increases. In addition, zero training loss is attained in all simulations. As shown in Table II, this paper utilizes six real-world datasets, with feature dimensions ranging from four to ten. As each feature is mapped to a qubit, the number of qubits in our simulations also ranges

TABLE VI. Performance analysis of the proposed quantum algorithm across various real-world datasets under depolarizing noise.

		Macroaverage			Microaverage			Weighted average			
Dataset	Kernel function	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Accuracy
Iris	NXQK, NYQK, NZQK	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Tae	NFQK	0.6380	0.6319	0.6295	0.6304	0.6304	0.6304	0.6392	0.6304	0.6293	0.6304
Penguin	NCQK	0.9932	0.9815	0.9870	0.9900	0.9900	0.9900	0.9902	0.9900	0.9899	0.9900
Glass	NXQK, NYQK, NZQK	0.7289	0.5895	0.6218	0.7385	0.7385	0.7385	0.7312	0.7385	0.7064	0.7385
Ecoli	NLQK	0.7515	0.6841	0.7074	0.8416	0.8416	0.8416	0.8355	0.8416	0.8357	0.8416
Vowel	NFQK	0.8676	0.8676	0.8536	0.8616	0.8616	0.8616	0.8779	0.8616	0.8535	0.8616

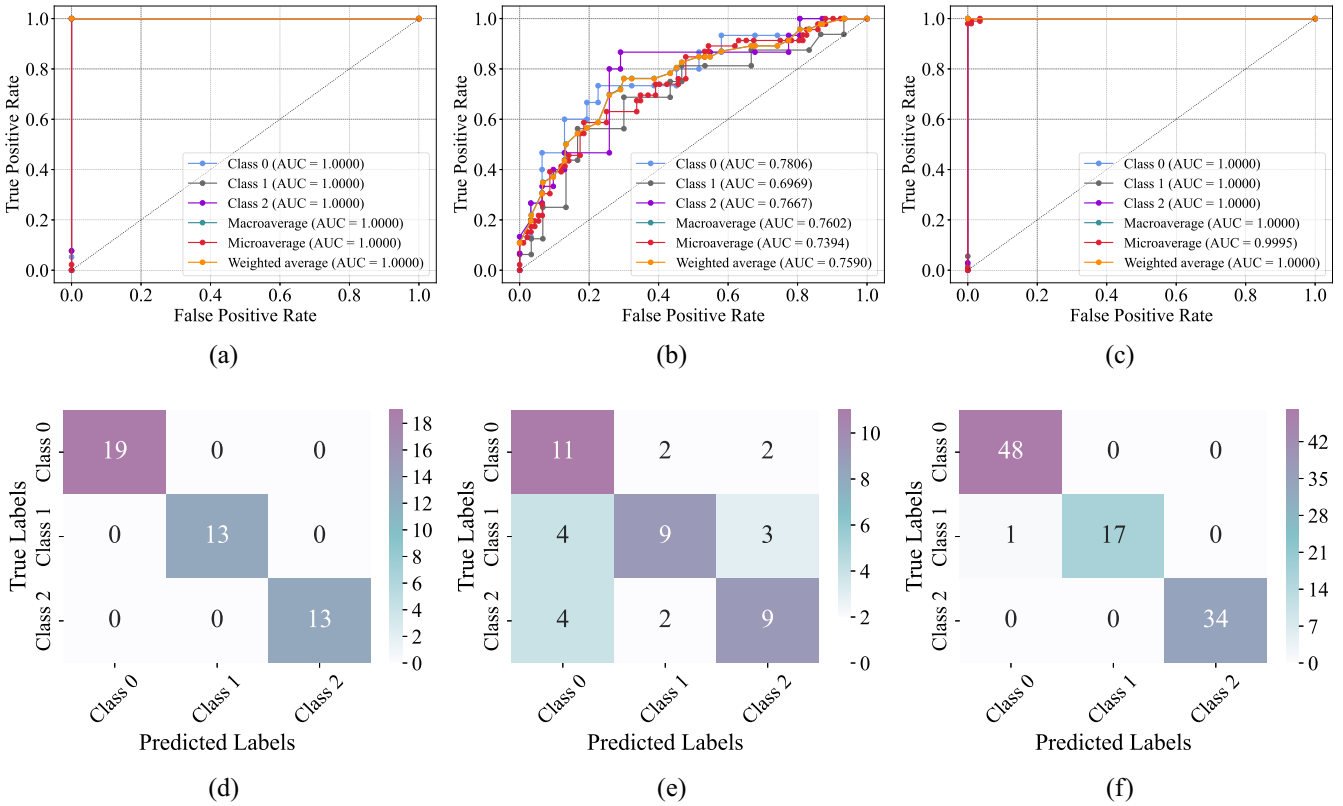


FIG. 16. ROC curve and confusion matrix for each dataset, evaluated under depolarizing noise: (a), (d) for the Iris dataset, (b), (e) for the Tae dataset, and (c), (f) for the Penguin dataset.

from four to ten. Therefore, such exponential concentration does not occur in the six real-world datasets used in this paper. To rigorously preclude the potential for exponential concentration, we further evaluate generalization performance using learning curves derived from six real-world datasets. As illustrated in Fig. 14, the testing accuracy on all six real-world datasets improves with the increasing training size, indicating a progressive improvement in generalization performance. Undoubtedly, the effects of exponential concentration are not observed.

B. Effect of hardware noise

Given the constraints of NISQ devices, hardware noise may disrupt quantum kernels constructed with parameterized quantum circuits, reducing the accuracy of similarity metrics. That is, hardware noise may bias the kernel values away from their true values, potentially affecting the classification accuracy of the proposed quantum algorithm. Here we focus on depolarizing noise and conduct an analytical study of its effect on the quantum algorithm. To investigate the effect of depolarizing noise, we employ the noise model described in Refs. [71,72]. Suppose each layer of an ideal parameterized quantum circuit is followed by a depolarizing noise channel with a probability $\bar{p}$. The model for the depolarizing noise channel is denoted by $\mathcal{N}_{\bar{p}} = \mathcal{N}_{\bar{p}}^1 \otimes \cdots \otimes \mathcal{N}_{\bar{p}}^N$. For any quantum state η, we have

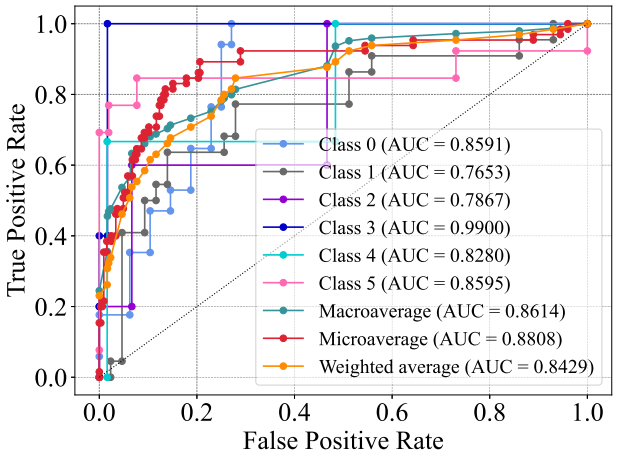
$$\mathcal{N}_{\bar{p}}^I(\eta) = (1-\bar{p})\eta + \frac{\bar{p}}{3}(\sigma_x\eta\sigma_x + \sigma_y\eta\sigma_y + \sigma_z\eta\sigma_z) \qquad (17)$$

with the corresponding Kraus matrices $M_0 = \sqrt{1-\bar{p}}\sigma_0$, $M_1 = \sqrt{\bar{p}/3}\sigma_x$, $M_2 = \sqrt{\bar{p}/3}\sigma_y$, and $M_3 = \sqrt{\bar{p}/3}\sigma_z$. As shown in Fig. 15, the quantum kernel with depolarizing noise is expressed as

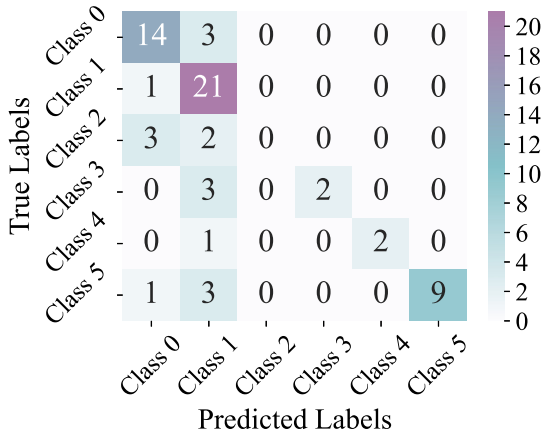
$$\kappa'(\vec{x}_i, \vec{x}_j) = \mathrm{Tr}[\Pi(\mathcal{N}_{\bar{p}} \circ \mathcal{S}^\dagger(\vec{x}_j) \circ \mathcal{N}_{\bar{p}} \circ \mathcal{S}(\vec{x}_i))\Pi], \qquad (18)$$

where $\Pi = (|0\rangle\langle 0|)^{\otimes N}$. From Eqs. (17) and (18), it is evident that the channel is noise-free only when the condition $\bar{p} = 0$ is met. Therefore, hardware noise may inevitably introduce bias, causing kernel values to shift from their true values.

To evaluate the proposed quantum algorithm under depolarizing noise, we construct six distinct noisy quantum kernels, detailed in Table V. Then, we evaluate the classification performance of the quantum algorithm with noisy quantum kernels using confusion matrices and ROC curves (see Figs. 16–20). In this paper, we set the depolarizing noise probability to $\bar{p} = 0.05$ to emulate realistic NISQ-level hardware noise. As shown in Table VI, the noisy Pauli-X, Pauli-Y, and Pauli-Z quantum kernels exhibit identical performance, with their equivalence proven in the Appendix E. From the comparative analysis in Tables IV and VI, we further obtain the following results: for the Iris and Penguin datasets, the presence of depolarizing noise does not affect the classification performance of the quantum algorithm. Interestingly, for the Tae datasets, depolarizing noise appears to improve the quantum algorithm's classification accuracy. In contrast, for the Glass, Ecoli, and Vowel datasets, depolarizing noise negatively affects the quantum algorithm's classification performance.

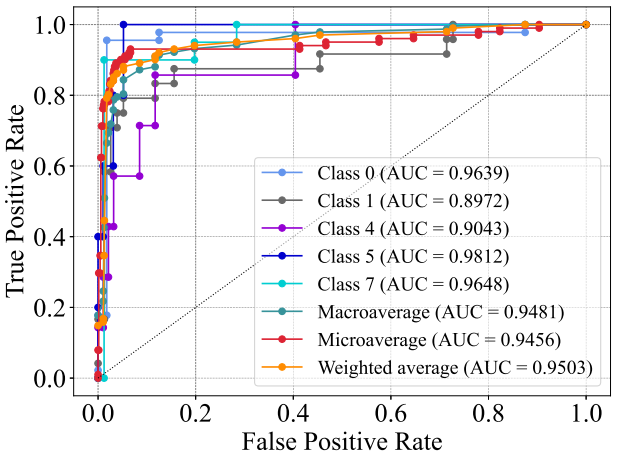


(a)

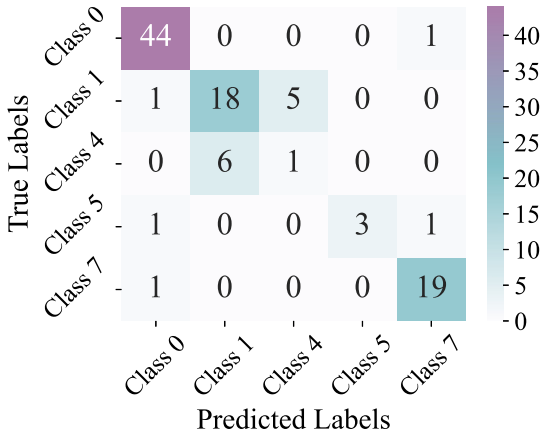


(b)

FIG. 17. ROC curve (a) and confusion matrix (b) for the Glass dataset under depolarizing noise.



(a)



(b)

FIG. 18. ROC curve (a) and confusion matrix (b) for the Ecoli dataset under depolarizing noise.

The negative effect can be further analyzed in detail through Figs. 9(b), 10(b), 12, 17(b), 18(b), and 20. It is evident that depolarizing noise mainly affects the classification of Class 3 in the Glass dataset, Class 1 in the Ecoli dataset, and Classes 1, 2, 4, 5, 6, and 10 in the Vowel dataset. To analyze the effects of depolarizing noise on overall performance, we also adopt ROC curves. It is found from Figs. 9(a), 10(a), 11, 17(a), 18(a), and 19 that the overall performance of the quantum algorithm is affected to varying extents under depolarizing noise.

Building on the analysis above, hardware noise introduces bias to kernel values, leading to varied outcomes: it may enhance category separation and boost classification performance, blur boundaries and diminish accuracy, or have no effect at all. Therefore, future work should focus on exploring nontrivial quantum kernels, particularly those with entangled structures, to further improve generalization and classification performance. We should also investigate the trade-off between kernel complexity and hardware noise, with a focus on real-world applications constrained by limited resources.

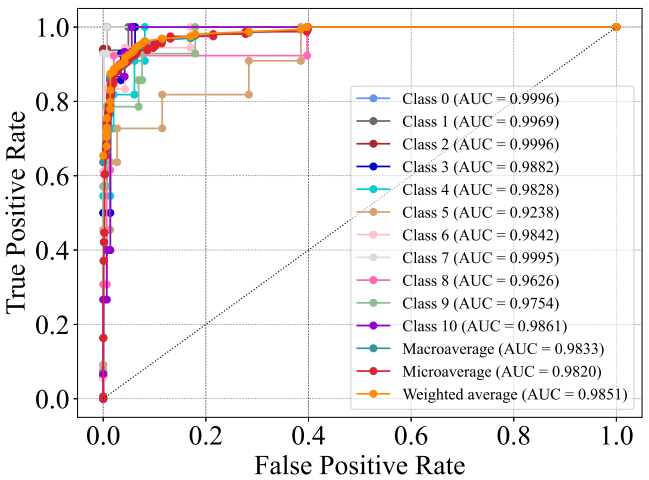


FIG. 19. ROC curve for the Vowel dataset under depolarizing noise. The AUCs also all exceed 0.9000.

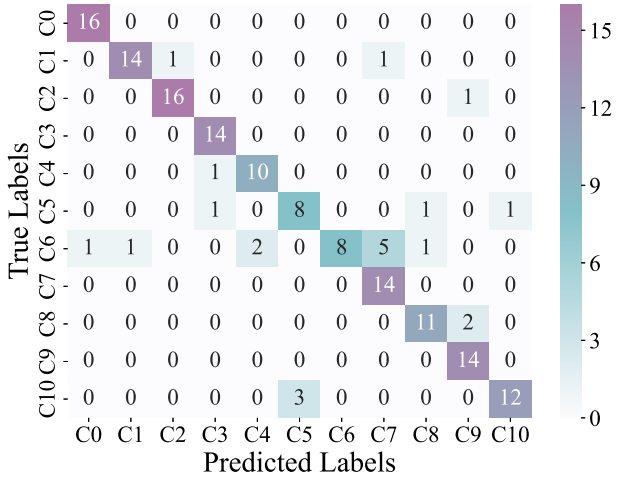


FIG. 20. Confusion matrix for the Vowel dataset under depolarizing noise. C0–C10, Class 0 to 10.

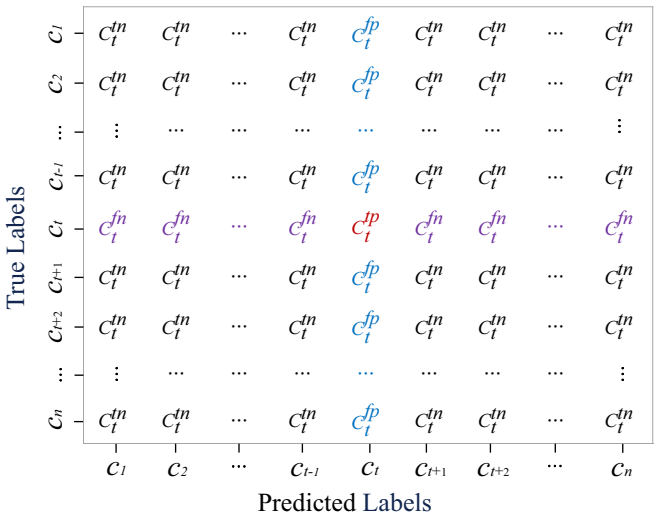


FIG. 21. Schematic diagram of the confusion matrix for Class t (c_t).

VII. CONCLUSION

In the current NISQ era, it is crucial to develop quantum machine learning algorithms tailored for effective operation on NISQ devices. This paper proposed a quantum machine learning algorithm for multiclass classification problems on NISQ devices, termed QEM-SVM. The results from quantum simulations demonstrated that the proposed quantum algorithm surpasses its classical counterparts in both classification accuracy and generalization performance. The quantum simulations further illuminated the effects of quantum kernel methods on classification and generalization performance, demonstrating their potential in advancing quantum machine learning. Furthermore, this paper successfully addressed the future work proposed by Havlíček *et al.* [26] and further expanded upon their research.

ACKNOWLEDGMENTS

This work is supported by ASTAR (M21K2c0116, M24M8b0004), Singapore National Research foundation (NRF-CRP22-2019-0004, NRF-CRP30-2023-0003, NRF-CRP31-0001, NRF2023-ITC004-001 and NRF-MSG-2023-0002) and Singapore Ministry of Education Tier 2 Grant (MOE-T2EP50221-0005, MOE-T2EP50222-0018). Furthermore, we acknowledge the support of Dieter Schwarz Stiftung gGmbH for the QUASAR project. This work is also supported by the National Natural Science Foundation of China (Grants No. 61703151 and No. 62076091) and the Natural Science Foundation of Hunan Province (Grant No. 2023JJ30167).

APPENDIX A: PERFORMANCE METRICS

In this Appendix, we provide a comprehensive set of performance metrics [73,74] for evaluating the effectiveness of the proposed quantum-enhanced multiclass SVM. These metrics are meticulously designed to precisely evaluate the predictive capability, accuracy, and other key aspects of the quantum algorithm in multiclass classification tasks.

As illustrated in Fig. 21, the precision M_t^ϕ, recall M_t^θ, and F1 score M_t^τ for Class t can be denoted as follows: $M_t^\phi = C_t^{tp}/(C_t^{tp} + C_t^{fp})$, $M_t^\theta = C_t^{tp}/(C_t^{tp} + C_t^{fn})$, and $M_t^\tau = 2C_t^{tp}/(2C_t^{tp} + C_t^{fn} + C_t^{fp})$. As shown in Fig. 21, C_t^{tp} represents testing instances correctly predicted as Class t; C_t^{fn} denotes testing instances incorrectly predicted as not Class t despite belonging to it; C_t^{fp} is testing instances incorrectly predicted as Class t despite not belonging to it; C_t^{tn} describes testing instances correctly predicted as not Class t. Given a classification task with l classes, calculating the performance metrics for each class is essential.

To achieve a thorough evaluation of algorithm performance within such multiclass classification tasks, we integrate macroaverage, microaverage, and weighted average approaches. In the macroaverage approach, the computational expressions for macroaverage precision M_{mac}^ϕ, macroaverage recall M_{mac}^θ, and macroaverage F1 score M_{mac}^τ are given by $M_{\text{mac}}^\phi = 1/l \sum_{i=1}^l M_i^\phi$, $M_{\text{mac}}^\theta = 1/l \sum_{i=1}^l M_i^\theta$, and $M_{\text{mac}}^\tau = 1/l \sum_{i=1}^l M_i^\tau$. In the weighted average approach, the computational expressions for weighted average precision M_{wgt}^ϕ, weighted average recall M_{wgt}^θ, and weighted average F1 score M_{wgt}^τ can be expressed as follows: $M_{\text{wgt}}^\phi = \sum_{i=1}^l \gamma_i M_i^\phi$, $M_{\text{wgt}}^\theta = \sum_{i=1}^l \gamma_i M_i^\theta$, and $M_{\text{wgt}}^\tau = \sum_{i=1}^l \gamma_i M_i^\tau$, where γ_i is the weight of Class i. In the microaverage approach, the computational expressions for microaverage precision M_{mic}^ϕ, microaverage recall M_{mic}^θ, and microaverage F1 score M_{mic}^τ are described as follows: $M_{\text{mic}}^\phi = \sum_{i=1}^l C_i^{tp} / \sum_{i=1}^l (C_i^{tp} + C_i^{fp})$, $M_{\text{mic}}^\theta = \sum_{i=1}^l C_i^{tp} / \sum_{i=1}^l (C_i^{tp} + C_i^{fn})$, and $M_{\text{mic}}^\tau = \sum_{i=1}^l 2C_i^{tp} / \sum_{i=1}^l (2C_i^{tp} + C_i^{fn} + C_i^{fp})$, where M_{mic}^τ is the harmonic mean of M_{mic}^ϕ and M_{mic}^θ. Furthermore, overall accuracy is regarded as a performance metric, expressed as $M' = 1/C' \sum_{i=1}^l C_i^{tp}$, where C' denotes all testing instances. According to the performance metrics discussed above, a higher metric value signifies a stronger classification performance of the proposed quantum algorithm, whereas a lower value indicates inferior classification performance.

APPENDIX B: DERIVATION OF THE FINAL DECISION FUNCTION

In this Appendix, we present the step-by-step derivation from Eqs. (15) to (16). Derived from the Karush-Kuhn-Tucker (KKT) conditions [75] and Eq. (15), we have $\partial\tilde{L}/\partial\vec{\theta}_{s'}=0$, $\partial\tilde{L}/\partial b_{s'}=0$, and $\partial\tilde{L}/\partial\xi_i^{s'}=0$. Therefore, the optimal conditions we obtain can be expressed as

$$\vec{\theta}_{s'}=\sum_{i=1}^{m}\left(\delta_{i,s'}\sum_{s=1}^{l}\alpha_i^s-\alpha_i^{s'}\right)\vec{x}_i, \tag{B1}$$

$$\sum_{i=1}^{m}\alpha_i^{s'}=\sum_{i,s}\alpha_i^s\delta_{i,s'}, \tag{B2}$$

$$C\geqslant\alpha_i^{s'}\geqslant 0,\quad C=\alpha_i^{s'}+\mu_i^{s'}. \tag{B3}$$

Then we define the following expressions:

$$\left(\sum_{i,s}\alpha_i^s\right)b_{y_i}=\sum_{s=1}^{l}b_s\left(\sum_{i=1}^{m}\delta_{i,s}\sum_{s=1}^{l}\alpha_i^s\right)=\sum_{s=1}^{l}b_s\sum_{i,s}\alpha_i^s\delta_{i,s} \tag{B4}$$

and

$$\sum_{i,s}\alpha_i^s b_s=\sum_{s=1}^{l}b_s\sum_{i=1}^{m}\alpha_i^s. \tag{B5}$$

Given that $\sum_{i=1}^{m}\alpha_i^{s'}=\sum_{i,s}\alpha_i^s\delta_{i,s'}$, we obtain $(\sum_{i,s}\alpha_i^s)b_{y_i}=\sum_{i,s}\alpha_i^s b_s$. Therefore, the dual formulation is expressed as

$$\begin{aligned}\max_{\vec{\alpha}}\tilde{L}(\vec{\alpha})=2\sum_{s,i}\alpha_i^s+\frac{1}{2}\sum_{s,i,j}\Bigg(&\delta_{i,s}\sum_{s=1}^{l}\alpha_i^s\cdot\delta_{j,s}\sum_{s=1}^{l}\alpha_j^s\\&-\alpha_j^s\delta_{i,s}\sum_{s=1}^{l}\alpha_i^s+\alpha_i^s\delta_{j,s}\sum_{s=1}^{l}\alpha_j^s\\&-2\alpha_i^s\delta_{j,y_i}\sum_{s=1}^{l}\alpha_j^s+2\alpha_i^s\alpha_j^{y_i}-\alpha_i^s\alpha_j^s\Bigg)\langle\vec{x}_i,\vec{x}_j\rangle,\end{aligned} \tag{B6}$$

with the following constraints: $\alpha_i^{y_i}=0$, $\sum_{i=1}^{m}\alpha_i^{s'}=\sum_{i,s}\alpha_i^s\delta_{i,s'}$, $C\geqslant\alpha_i^{s'}\geqslant 0$, $\forall s\neq y_i$, and $\forall s'=1,\ldots,l$. Practically speaking, the solution to Eq. (B6) demonstrates greater simplicity compared with that of Eq. (13). To enhance this inherent simplicity, we simplify the dual formulation in two steps. First, we attempt to simplify the middle two terms. Given that $\sum_{s,i,j}\alpha_j^s(\delta_{i,s}\sum_{s=1}^{l}\alpha_i^s)=\sum_{s,i,j}\alpha_i^s(\delta_{j,s}\sum_{s=1}^{l}\alpha_j^s)$, the dual formulation can be expressed as $2\sum_{s,i}\alpha_i^s+1/2\sum_{s,i,j}(\delta_{i,s}\sum_{s=1}^{l}\alpha_i^s\cdot\delta_{j,s}\sum_{s=1}^{l}\alpha_j^s-2\alpha_i^s\delta_{j,y_i}\sum_{s=1}^{l}\alpha_j^s+2\alpha_i^s\alpha_j^{y_i}-\alpha_i^s\alpha_j^s)\langle\vec{x}_i,\vec{x}_j\rangle$. Second, based on $\sum_{s=1}^{l}\delta_{i,s}\delta_{j,s}=\delta_{i,y_j}=\delta_{j,y_i}=1$ if $y_i=y_j$ and $\sum_{s=1}^{l}\delta_{i,s}\delta_{j,s}=\delta_{i,y_j}=\delta_{j,y_i}=0$ otherwise, the dual formulation can be simplified to $2\sum_{s,i}\alpha_i^s+1/2\sum_{s,i,j}[-\delta_{j,y_i}(\sum_{s=1}^{l}\alpha_i^s)(\sum_{s=1}^{l}\alpha_j^s)+2\alpha_i^s\alpha_j^{y_i}-\alpha_i^s\alpha_j^s]\langle\vec{x}_i,\vec{x}_j\rangle$.

The solution to the dual formulation can be obtained through the sequential minimal optimization (SMO) algorithm [76–78]. This yields the decision function

$$\tilde{f}(\vec{x},\vec{\alpha})=\arg\max_{s'}\left(\sum_{i:y_i=s'}\sum_{s=1}^{l}\alpha_i^s-\sum_{i:y_i\neq s'}\alpha_i^{s'}\right)\langle\vec{x}_i,\vec{x}\rangle+b_{s'}. \tag{B7}$$

Here we replace $\langle\vec{x}_i,\vec{x}_j\rangle$ with a quantum kernel $\kappa(\vec{x}_i,\vec{x}_j)$ to leverage the expressive power of quantum state spaces [4]. Thus, the resulting decision function is given by

$$\tilde{f}(\vec{x},\vec{\alpha})=\arg\max_{s'}\sum_{i=1}^{m}\left(\delta_{i,s'}\sum_{s=1}^{l}\alpha_i^s-\alpha_i^{s'}\right)\kappa(\vec{x}_i,\vec{x})+b_{s'}. \tag{B8}$$

ALGORITHM 1. QEM-SVM.

Require: Training dataset $\mathcal{X}=\{(\vec{x}_1,y_1),\ldots,(\vec{x}_m,y_m)\}$, where $\vec{x}_i=(x_1,x_2,\ldots,x_N)^T$ is the feature vector of the ith data point and y_i is the corresponding label; Label matrix L; Testing instance $\vec{x}_t$
Ensure: Predicted class label $\tilde{y}$

Initialization:
Initialize parameters: $\vec{\alpha}$, $\vec{\theta}$, b
Set C as the penalty parameter
Set M as the resulting bit strings
Set Z as the number of measurements
Prepare quantum registers for initial states $|0\rangle^{\otimes N}$
Calibrate the $NISQ$ device to construct the quantum circuit
Quantum kernel estimation:
for each pair of data points $(\vec{x}_i,\vec{x}_j)$ **do**
  Set the counter $R=\mathrm{r}(0,\ldots,0)=0$
  for $z\leftarrow 1$ **to** Z **do**
    Apply $\mathcal{S}(\vec{x}_i)$ to prepare quantum state $|\phi(\vec{x}_i)\rangle$
    Apply $\mathcal{S}^\dagger(\vec{x}_j)$ to the prepared quantum state $|\phi(\vec{x}_i)\rangle$
    Measure the resulting state $\mathcal{S}^\dagger(\vec{x}_j)\mathcal{S}(\vec{x}_i)|0\rangle^{\otimes N}$
    if $M=0^N$ **then**
      $R=R+1$
    else
      $R=R$
  Use the frequency RZ^{-1} to estimate K_{ij}
  Store K_{ij} in the quantum kernel matrix K
Return: Quantum kernel matrix K
Training:
for $s\leftarrow 1$ **to** l **do**
  Construct the binary label vector L_s
  for $i\leftarrow 1$ **to** m **do**
    if $L[i]=s$ **then**
      $L_s[i]=1$
    else
      $L_s[i]=-1$
  Update to optimal parameters:
  $\vec{\alpha}^s,\vec{\theta}_s,b_s\leftarrow\mathrm{SMO}(K,L_s,C)$
Return: Trained parameters $(\vec{\alpha},\vec{\theta},b)$
Prediction:
for $s\leftarrow 1$ **to** l **do**
  Calculate the sth decision function using the trained parameters $(\vec{\alpha},\vec{\theta},b)$ and the testing instance $\vec{x}_t$: $\tilde{f}_s(\vec{x}_t)=\sum_{i\in\Omega}\alpha_i^s y_i K_{it}+b_s$
Return: $\tilde{y}=\arg\max_{s=1,\ldots,l}\tilde{f}_s(\vec{x}_t)$

The pseudocode for the QEM-SVM can be found in Algorithm 1.

APPENDIX C: GENERALIZATION ERROR BOUNDS

Given a sample $\mathrm{E} = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_\mathrm{D}\}$, where $\vec{x}_i \in \mathbb{R}^N$. Let $\kappa : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ denote a quantum kernel function, and let $K_{ij} = \kappa(\vec{x}_i, \vec{x}_j)$ be the corresponding quantum kernel matrix. Furthermore, a class of functions can be denoted by

$$\mathcal{F} = \left\{ f(x) = \sum_{i=1}^{\mathrm{D}} O_i \kappa(\vec{x}, \vec{x}_i) \mid \|\vec{O}\|_2 \leqslant \Delta \right\} \tag{C1}$$

with $\vec{O} = (O_1, \ldots, O_\mathrm{D})^T \in \mathbb{R}^\mathrm{D}$. From the description in Ref. [79], the empirical Rademacher complexity of $\mathcal{F}$ is defined as

$$\hat{\mathfrak{R}}_\mathrm{E}(\mathcal{F}) = \mathbb{E}_\varpi \left[\sup_{\|\vec{O}\|_2 \leqslant \Delta} \frac{1}{\mathrm{D}} \sum_{i=1}^{\mathrm{D}} \varpi_i \sum_{j=1}^{\mathrm{D}} O_j \kappa(\vec{x}_i, \vec{x}_j) \right], \tag{C2}$$

where $\varpi_1, \varpi_2, \ldots, \varpi_\mathrm{D}$ are i.i.d. Rademacher random variables uniformly drawn from $\{-1, 1\}$. Reordering the summation, Eq. (C2) becomes

$$\hat{\mathfrak{R}}_\mathrm{E}(\mathcal{F}) = \mathbb{E}_\varpi \left[\sup_{\|\vec{O}\|_2 \leqslant \Delta} \frac{1}{\mathrm{D}} \sum_{j=1}^{\mathrm{D}} O_j \vec{v}_j \right], \tag{C3}$$

where

$$\vec{v}_j := \sum_{i=1}^{\mathrm{D}} \varpi_i \kappa(\vec{x}_i, \vec{x}_j). \tag{C4}$$

Given that $\|\vec{O}\|_2 \leqslant \Delta$, we employ the Cauchy–Schwarz inequality to deduce

$$\sum_{j=1}^{\mathrm{D}} O_j \vec{v}_j \leqslant \|\vec{O}\| \|\vec{v}\| \leqslant \Delta \|\vec{v}\|. \tag{C5}$$

Substituting Eq. (C5) results in a simplification of Eq. (C3), yielding

$$\hat{\mathfrak{R}}_\mathrm{E}(\mathcal{F}) = \frac{\Delta}{\mathrm{D}} \mathbb{E}_\varpi [\|\vec{v}\|]. \tag{C6}$$

By the Cauchy-Schwarz inequality, we have

$$(\mathbb{E}_\varpi [\|\vec{v}\|])^2 \leqslant \mathbb{E}_\varpi [\|\vec{v}\|^2]. \tag{C7}$$

Therefore, $\frac{\Delta}{\mathrm{D}} \sqrt{\mathbb{E}_\varpi [\|\vec{v}\|^2]}$ denotes the upper bound of $\hat{\mathfrak{R}}_\mathrm{E}(\mathcal{F})$:

$$\hat{\mathfrak{R}}_\mathrm{E}(\mathcal{F}) \leqslant \frac{\Delta}{\mathrm{D}} \sqrt{\mathbb{E}_\varpi [\|\vec{v}\|^2]}. \tag{C8}$$

To determine the upper bound of $\hat{\mathfrak{R}}_\mathrm{E}(\mathcal{F})$, we first analyze $\|\vec{v}\|^2$, where

$$\begin{aligned} \|\vec{v}\|^2 &= \sum_{j=1}^{\mathrm{D}} \vec{v}_j^2 \\ &= \sum_{j=1}^{\mathrm{D}} \left(\sum_{i=1}^{\mathrm{D}} \varpi_i \kappa(\vec{x}_i, \vec{x}_j) \right)^2 \\ &= \sum_{j=1}^{\mathrm{D}} \sum_{i=1}^{\mathrm{D}} \sum_{i'=1}^{\mathrm{D}} \varpi_i \varpi_{i'} \kappa(\vec{x}_i, \vec{x}_j) \kappa(\vec{x}_{i'}, \vec{x}_j). \end{aligned} \tag{C9}$$

Given that the Rademacher random variables satisfy $\mathbb{E}_\varpi [\varpi_i \varpi_{i'}] = \begin{cases} 1, & i = i' \\ 0, & i \neq i' \end{cases}$, we have

$$\mathbb{E}_\varpi [\|\vec{v}\|^2] = \sum_{i,j}^{\mathrm{D}} \kappa(\vec{x}_i, \vec{x}_j)^2 = \|K\|_F^2, \tag{C10}$$

where $\|K\|_F$ is the Frobenius norm of the quantum kernel matrix K. By substituting Eq. (C10) into Eq. (C8), we conclude that $\hat{\mathfrak{R}}_\mathrm{E}(\mathcal{F})$ satisfies the condition

$$\hat{\mathfrak{R}}_\mathrm{E}(\mathcal{F}) \leqslant \frac{\Delta}{\mathrm{D}} \|K\|_F. \tag{C11}$$

Therefore, the upper bound of the empirical Rademacher complexity is proportional to the Frobenius norm of the quantum kernel matrix. The generalization analysis results are listed in Table VII. It can be inferred from Table VII that quantum kernel methods exhibit better generalization performance than classical kernel methods.

APPENDIX D: PROOF OF EQUIVALENCE AMONG PAULI-X, PAULI-Y, AND PAULI-Z QUANTUM KERNELS

Remarkably, Pauli-X, Pauli-Y, and Pauli-Z quantum kernels achieve identical mean accuracy in multiclass classification tasks on various real-world datasets. Although these quantum kernels exhibit different physical properties and mechanisms, our experimental results reveal that their classification performance is remarkably consistent on various multiclass classification tasks. This consistency has prompted further investigation into the potential theoretical equivalence of these quantum kernels.

1. Mathematical analysis of Pauli-X quantum kernels

For any input classical state $\vec{x}_i = (2x_0, 2x_1, \ldots, 2x_{N-1})^T \in \mathbb{R}^N$, the Pauli-$X$ quantum kernel has the form

$$\kappa_x(\vec{x}_i, \vec{x}_i') = |\langle \phi_x(\vec{x}_i) | \phi_x(\vec{x}_i') \rangle|^2. \tag{D1}$$

Let us first consider the process of encoding $\vec{x}_i$ as a quantum state, which we denote by

$$|\phi_x(\vec{x}_i)\rangle = \begin{bmatrix} \cos x_0 \\ -i \sin x_0 \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} \cos x_{N-1} \\ -i \sin x_{N-1} \end{bmatrix}. \tag{D2}$$

Substituting Eq. (D2) into Eq. (D1), we have

$$\kappa_x(\vec{x}_i, \vec{x}_i') = \prod_{i=0}^{N-1} |\cos(x_i - x_i')|^2. \tag{D3}$$

Here we utilize $\langle 0|0\rangle = \langle 1|1\rangle = 1$ and $\langle 0|1\rangle = \langle 1|0\rangle = 0$.

2. Mathematical analysis of Pauli-Y quantum kernels

Similar to the Pauli-X quantum kernel, the Pauli-Y quantum kernel is denoted as

$$\kappa_y(\vec{x}_i, \vec{x}_i') = |\langle \phi_y(\vec{x}_i) | \phi_y(\vec{x}_i') \rangle|^2. \tag{D4}$$

Then we encode $\vec{x}_i$ as a quantum state, given by

$$|\phi_y(\vec{x}_i)\rangle = \begin{bmatrix} \cos x_0 \\ \sin x_0 \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} \cos x_{N-1} \\ \sin x_{N-1} \end{bmatrix}. \tag{D5}$$

TABLE VII. Generalization analysis of various kernel functions.

Dataset	Kernel function	$\|K\|_F$	$\hat{\mathfrak{R}}_E(\mathcal{F})$	Upper bound
Iris	LK	303.5553	2.5593	2.8910
	PK	527.1340	4.8091	5.0203
	SK	76.3759	0.6851	0.7274
	GK	50.4150	0.4601	0.4801
	FQK	39.9393	0.3589	0.3804
	LQK	41.9019	0.3814	0.3991
	CQK	40.4679	0.3683	0.3854
	XQK, YQK, ZQK	49.4697	0.4519	0.4711
Tae	LK	266.1292	2.4115	2.5346
	PK	445.2876	4.1964	4.2408
	SK	76.4926	0.6603	0.7285
	GK	37.7070	0.3385	0.3591
	FQK	26.2685	0.2370	0.2502
	LQK	28.5439	0.2559	0.2718
	CQK	27.5773	0.2491	0.2626
	XQK, YQK, ZQK	32.2922	0.2928	0.3075
Penguin	LK	778.3569	2.6868	3.3406
	PK	1074.1190	4.0841	4.6100
	SK	168.8427	0.6911	0.7246
	GK	92.4048	0.3862	0.3966
	FQK	54.5493	0.2353	0.2341
	LQK	62.1466	0.2649	0.2667
	CQK	58.5630	0.2501	0.2513
	XQK, YQK, ZQK	81.9560	0.3430	0.3517
Glass	LK	573.0572	3.3799	3.8460
	PK	10762.2241	69.6747	72.2297
	SK	109.0790	0.6680	0.7321
	GK	66.5629	0.4125	0.4467
	FQK	49.7011	0.3135	0.3336
	LQK	54.1024	0.3412	0.3631
	CQK	53.1684	0.3361	0.3568
	XQK, YQK, ZQK	60.8262	0.3812	0.4082
Ecoli	LK	795.9544	3.1618	3.3870
	PK	424827.4131	1807.6520	1807.7762
	SK	169.9102	0.7020	0.7230
	GK	97.1009	0.4020	0.4132
	FQK	62.1455	0.2549	0.2644
	LQK	70.9959	0.2872	0.3021
	CQK	69.5493	0.2822	0.2960
	XQK, YQK, ZQK	86.7654	0.3574	0.3692
Vowel	LK	1443.9961	3.3813	3.9133
	PK	2169.7484	5.3771	5.8801
	SK	268.2608	0.6948	0.7270
	GK	87.5059	0.2337	0.2371
	FQK	40.6547	0.1107	0.1102
	LQK	47.2345	0.1259	0.1280
	CQK	43.8518	0.1176	0.1188
	XQK, YQK, ZQK	63.6651	0.1700	0.1725

Substituting Eq. (D5) into Eq. (D4), we have

$$\kappa_y(\vec{x}_i, \vec{x}_i') = \prod_{i=0}^{N-1} |\cos(x_i - x_i')|^2. \tag{D6}$$

3. Mathematical analysis of Pauli-Z quantum kernels

Following the approach for deriving the Pauli-X and Pauli-Y quantum kernels, the Pauli-Z quantum kernel is expressed as

$$\kappa_z(\vec{x}_i, \vec{x}_i') = |\langle \phi_z(\vec{x}_i) | \phi_z(\vec{x}_i') \rangle|^2. \tag{D7}$$

Then we encode $\vec{x}_i$ as a quantum state, denoted by

$$|\phi_z(\vec{x}_i)\rangle = \left(\frac{\sqrt{2}}{2}\right)^N \begin{bmatrix} \mathcal{M}_{(x_0)} \\ \mathcal{N}_{(x_0)} \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} \mathcal{M}_{(x_{N-1})} \\ \mathcal{N}_{(x_{N-1})} \end{bmatrix} \tag{D8}$$

with $\mathcal{M}_{(x)} = \cos x - i \sin x$, $\mathcal{N}_{(x)} = \cos x + i \sin x$. The complex conjugate transpose of $|\phi_z(\vec{x}_i)\rangle$ is thus given by

$$\langle \phi_z(\vec{x}_i')| = \left(\frac{\sqrt{2}}{2}\right)^N \begin{bmatrix} \mathcal{N}_{(x_0')} \\ \mathcal{M}_{(x_0')} \end{bmatrix}^T \otimes \cdots \otimes \begin{bmatrix} \mathcal{N}_{(x_{N-1}')} \\ \mathcal{M}_{(x_{N-1}')} \end{bmatrix}^T. \tag{D9}$$

Substituting Eqs. (D8) and (D9) into Eq. (D7), we have

$$\begin{aligned} \kappa_z(\vec{x}_i, \vec{x}_i') &= \left(\frac{1}{4}\right)^N \prod_{i=0}^{N-1} \left| \mathcal{N}_{(x_i)} \mathcal{M}_{(x_i')} + \mathcal{M}_{(x_i)} \mathcal{N}_{(x_i')} \right|^2 \\ &= \prod_{i=0}^{N-1} |\cos(x_i - x_i')|^2. \end{aligned} \tag{D10}$$

From Eqs. (D3), (D6), and (D10), we conclude that Pauli-X, Pauli-Y, and Pauli-Z quantum kernels are equivalent.

APPENDIX E: PROOF OF EQUIVALENCE AMONG NOISY PAULI-X, PAULI-Y, AND PAULI-Z QUANTUM KERNELS

The noisy Pauli-X, Pauli-Y, and Pauli-Z quantum kernels achieve identical accuracy in multiclass classification tasks across various real-world datasets. This raises the question: do these noisy quantum kernels remain theoretically equivalent? To address this, we further investigate their theoretical equivalence. As indicated in Eq. (17) of Sec. VI, the depolarizing noise channel for a single qubit can be equivalently expressed as

$$\mathcal{N}_{\bar{p}}^d(\eta) = \left(1 - \frac{4\bar{p}}{3}\right)\eta + \frac{2\bar{p}}{3}\mathbb{I}. \tag{E1}$$

In noisy Pauli-X and Pauli-Y quantum kernels, we first consider the case of a single-qubit system, where

$$\begin{aligned} \Gamma_k = &\left(1 - \frac{4\bar{p}}{3}\right)^2 \mathsf{R}^\dagger\left(\frac{x_k'}{2}\right)\mathsf{R}\left(\frac{x_k}{2}\right)|0\rangle\langle 0|\mathsf{R}^\dagger\left(\frac{x_k}{2}\right)\mathsf{R}\left(\frac{x_k'}{2}\right) \\ &+ \left(1 - \frac{4\bar{p}}{3}\right)\frac{2\bar{p}}{3}\mathbb{I} + \frac{2\bar{p}}{3}\mathbb{I} \end{aligned} \tag{E2}$$

with $\mathsf{R} \in \{R_x, R_y\}$. Similarly, in noisy Pauli-Z quantum kernels, we also first explore the case of a single-qubit system, where

$$\begin{aligned} \Gamma_k = &\left(1 - \frac{4\bar{p}}{3}\right)^2 H R_z^\dagger\left(\frac{x_k'}{2}\right)R_z\left(\frac{x_k}{2}\right)H|0\rangle\langle 0|H R_z^\dagger\left(\frac{x_k}{2}\right) \\ &\times R_z\left(\frac{x_k'}{2}\right)H + \left(1 - \frac{4\bar{p}}{3}\right)\frac{2\bar{p}}{3}\mathbb{I} + \frac{2\bar{p}}{3}\mathbb{I}. \end{aligned} \tag{E3}$$

Let Γ represent the density matrix of an N-qubit system. In this case, $\Gamma = \Gamma_0 \otimes \Gamma_1 \otimes \cdots \otimes \Gamma_{N-1}$. Given that the projector $\Pi = (|0\rangle\langle 0|)^{\otimes N}$, the noisy quantum kernel is given by

$$\kappa'(\vec{x}_i, \vec{x}_i') = \mathrm{Tr}\left[\Pi(\Gamma_0 \otimes \Gamma_1 \otimes \cdots \otimes \Gamma_{N-1})\right] = \prod_{k=0}^{N-1} \langle 0|\Gamma_k|0\rangle. \tag{E4}$$

The following mathematical analysis studies the noisy Pauli-X, Pauli-Y, and Pauli-Z quantum kernels individually.

1. Mathematical analysis of noisy Pauli-X quantum kernels

For any input classical state $\vec{x}_i = (2x_0, 2x_1, \ldots, 2x_{N-1})^T \in \mathbb{R}^N$, the noisy Pauli-$X$ quantum kernel is given by

$$\kappa_x'(\vec{x}_i, \vec{x}_i') = \mathrm{Tr}\left[\Pi(\mathcal{N}_{\bar{p}} \circ U_x{}^{\dagger}(\vec{x}_i') \circ \mathcal{N}_{\bar{p}} \circ U_x(\vec{x}_i))\Pi\right]. \tag{E5}$$

Combining Eqs. (E2) and (E4), Eq. (E5) can be expanded as

$$\kappa_x'(\vec{x}_i, \vec{x}_i') = \prod_{k=0}^{N-1} \left[\left(1 - \frac{4\bar{p}}{3}\right)^2 |\langle 0|R_x{}^{\dagger}(x_k')R_x(x_k)|0\rangle|^2 + \left(2 - \frac{4\bar{p}}{3}\right)\frac{2\bar{p}}{3}\right]. \tag{E6}$$

Substituting Eq. (2) leads to a simplification, yielding

$$\kappa_x'(\vec{x}_i, \vec{x}_i') = \prod_{k=0}^{N-1} \left[\left(1 - \frac{4\bar{p}}{3}\right)^2 |\cos(x_k - x_k')|^2 + \left(2 - \frac{4\bar{p}}{3}\right)\frac{2\bar{p}}{3}\right]. \tag{E7}$$

2. Mathematical analysis of noisy Pauli-Y quantum kernels

Similar to the noisy Pauli-X quantum kernel, the noisy Pauli-Y quantum kernel is expressed as

$$\kappa_y'(\vec{x}_i, \vec{x}_i') = \mathrm{Tr}[\Pi(\mathcal{N}_{\bar{p}} \circ U_y{}^{\dagger}(\vec{x}_i') \circ \mathcal{N}_{\bar{p}} \circ U_y(\vec{x}_i))\Pi]. \tag{E8}$$

Combining Eqs. (E2) and (E4), Eq. (E8) can be expanded as

$$\kappa_y'(\vec{x}_i, \vec{x}_i') = \prod_{k=0}^{N-1} \left[\left(1 - \frac{4\bar{p}}{3}\right)^2 |\langle 0|R_y{}^{\dagger}(x_k')R_y(x_k)|0\rangle|^2 + \left(2 - \frac{4\bar{p}}{3}\right)\frac{2\bar{p}}{3}\right]. \tag{E9}$$

Substituting Eq. (3) results in a simplification, yielding

$$\kappa_y'(\vec{x}_i, \vec{x}_i') = \prod_{k=0}^{N-1} \left[\left(1 - \frac{4\bar{p}}{3}\right)^2 |\cos(x_k - x_k')|^2 + \left(2 - \frac{4\bar{p}}{3}\right)\frac{2\bar{p}}{3}\right]. \tag{E10}$$

3. Mathematical analysis of noisy Pauli-Z quantum kernels

Building on the approach used to derive the noisy Pauli-X and Pauli-Y quantum kernels, the noisy Pauli-Z quantum kernel is described as

$$\kappa_z'(\vec{x}_i, \vec{x}_i') = \mathrm{Tr}[\Pi(\mathcal{N}_{\bar{p}} \circ U_z{}^{\dagger}(\vec{x}_i') \circ \mathcal{N}_{\bar{p}} \circ U_z(\vec{x}_i))\Pi]. \tag{E11}$$

Combining Eqs. (E3) and (E4), Eq. (E11) can be expanded as

$$\kappa_z'(\vec{x}_i, \vec{x}_i') = \prod_{k=0}^{N-1} \left[\left(1 - \frac{4\bar{p}}{3}\right)^2 |\langle 0|HR_z{}^{\dagger}(x_k')R_z(x_k)H|0\rangle|^2 + \left(2 - \frac{4\bar{p}}{3}\right)\frac{2\bar{p}}{3}\right]. \tag{E12}$$

Substituting Eq. (4) simplifies to

$$\kappa_z'(\vec{x}_i, \vec{x}_i') = \prod_{k=0}^{N-1} \left[\left(1 - \frac{4\bar{p}}{3}\right)^2 |\cos(x_k - x_k')|^2 + \left(2 - \frac{4\bar{p}}{3}\right)\frac{2\bar{p}}{3}\right]. \tag{E13}$$

From Eqs. (E7), (E10), and (E13), we conclude that noisy Pauli-X, Pauli-Y, and Pauli-Z quantum kernels are also equivalent.

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

[2] H.-Y. Huang, M. Broughton, M. Mohseni, R. Babbush, S. Boixo, H. Neven, and J. R. McClean, Power of data in quantum machine learning, Nat. Commun. **12**, 2631 (2021).

[3] M. Sajjan, J. Li, R. Selvarajan, S. H. Sureshbabu, S. S. Kale, R. Gupta, V. Singh, and S. Kais, Quantum machine learning for chemistry and physics, Chem. Soc. Rev. **51**, 6475 (2022).

[4] M. Schuld and N. Killoran, Quantum machine learning in feature Hilbert spaces, Phys. Rev. Lett. **122**, 040504 (2019).

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

[6] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, Quantum circuit learning, Phys. Rev. A **98**, 032309 (2018).

[7] D. Zhu, N. M. Linke, M. Benedetti, K. A. Landsman, N. H. Nguyen, C. H. Alderete, A. Perdomo-Ortiz, N. Korda, A. Garfoot, C. Brecque *et al.*, Training of quantum circuits on a hybrid quantum computer, Sci. Adv. **5**, eaaw9918 (2019).

[8] 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).

[9] A. Pérez-Salinas, D. López-Núñez, A. García-Sáez, P. Forn-Díaz, and J. I. Latorre, One qubit as a universal approximant, Phys. Rev. A **104**, 012405 (2021).

[10] G. González-García, R. Trivedi, and J. I. Cirac, Error propagation in NISQ devices for solving classical optimization problems, PRX Quantum **3**, 040326 (2022).

[11] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke *et al.*, Noisy intermediate-scale quantum algorithms, Rev. Mod. Phys. **94**, 015004 (2022).

[12] T. Dutta, A. Pérez-Salinas, J. P. S. Cheng, J. I. Latorre, and M. Mukherjee, Single-qubit universal classifier implemented on an ion-trap quantum device, Phys. Rev. A **106**, 012411 (2022).

[13] S. Chen, J. Cotler, H.-Y. Huang, and J. Li, The complexity of NISQ, Nat. Commun. **14**, 6001 (2023).

[14] J. Miao, C.-Y. Hsieh, and S.-X. Zhang, Neural-network-encoded variational quantum algorithms, Phys. Rev. Appl. **21**, 014053 (2024).

[15] T. Dutta, A. Jin, C. L. Huihong, J. I. Latorre, and M. Mukherjee, Practicality of training a quantum-classical machine in the NISQ era, arXiv:2401.12089.

[16] T. Jiang, J. L. Gradus, and A. J. Rosellini, Supervised machine learning: A brief primer, Behav. Ther. **51**, 675 (2020).

[17] S. A. Leachman and G. Merlino, The final frontier in cancer diagnosis, Nature (London) **542**, 36 (2017).

[18] Z. Zhang, P. Chen, M. McGough, F. Xing, C. Wang, M. Bui, Y. Xie, M. Sapkota, L. Cui, J. Dhillon *et al.*, Pathologist-level interpretable whole-slide cancer diagnosis with deep learning, Nat. Mach. Intell. **1**, 236 (2019).

[19] M. A. Myszczynska, P. N. Ojamies, A. M. Lacoste, D. Neil, A. Saffari, R. Mead, G. M. Hautbergue, J. D. Holbrook, and L. Ferraiuolo, Applications of machine learning to diagnosis and treatment of neurodegenerative diseases, Nat. Rev. Neurol. **16**, 440 (2020).

[20] K. Cao, Y. Xia, J. Yao, X. Han, L. Lambert, T. Zhang, W. Tang, G. Jin, H. Jiang, X. Fang *et al.*, Large-scale pancreatic cancer detection via non-contrast CT and deep learning, Nat. Med. **29**, 3033 (2023).

[21] E. N. Feinberg, D. Sur, Z. Wu, B. E. Husic, H. Mai, Y. Li, S. Sun, J. Yang, B. Ramsundar, and V. S. Pande, Potentialnet for molecular property prediction, ACS Cent. Sci. **4**, 1520 (2018).

[22] S. Ekins, A. C. Puhl, K. M. Zorn, T. R. Lane, D. P. Russo, J. J. Klein, A. J. Hickey, and A. M. Clark, Exploiting machine learning for end-to-end drug discovery and development, Nat. Mater. **18**, 435 (2019).

[23] L. Patel, T. Shukla, X. Huang, D. W. Ussery, and S. Wang, Machine learning methods in drug discovery, Molecules **25**, 5277 (2020).

[24] W. P. Walters and R. Barzilay, Applications of deep learning in molecule generation and molecular property prediction, Acc. Chem. Res. **54**, 263 (2021).

[25] W. Chen, X. Liu, S. Zhang, and S. Chen, Artificial intelligence for drug discovery: Resources, methods, and applications, Mol. Ther. Nucleic Acids **31**, 691 (2023).

[26] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, Supervised learning with quantum-enhanced feature spaces, Nature (London) **567**, 209 (2019).

[27] 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).

[28] S. L. Wu, S. Sun, W. Guan, C. Zhou, J. Chan, C. L. Cheng, T. Pham, Y. Qian, A. Z. Wang, R. Zhang *et al.*, Application of quantum machine learning using the quantum kernel algorithm on high energy physics analysis at the LHC, Phys. Rev. Res. **3**, 033221 (2021).

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

[30] R.-X. Zhao, J. Shi, and X. Li, QKSAN: A quantum kernel self-attention network, IEEE Trans. Pattern Anal. Mach. Intell. **46**, 10184 (2024).

[31] P. Rodriguez-Grasa, Y. Ban, and M. Sanz, Neural quantum kernels: Training quantum kernels with quantum neural networks, arXiv:2401.04642.

[32] C.-W. Hsu and C.-J. Lin, A comparison of methods for multiclass support vector machines, IEEE Trans. Neural Netw. **13**, 415 (2002).

[33] W. S. Noble, What is a support vector machine? Nat. Biotechnol. **24**, 1565 (2006).

[34] C. Ding, S. Wang, Y. Wang, Z. Wu, J. Sun, and Y. Mao, Machine-learning-based detection for quantum hacking attacks on continuous-variable quantum-key-distribution systems, Phys. Rev. A **107**, 062422 (2023).

[35] C. Campbell, Kernel methods: A survey of current techniques, Neurocomputing **48**, 63 (2002).

[36] V. D. Sánchez A, Advanced support vector machines and kernel methods, Neurocomputing **55**, 5 (2003).

[37] G. Pillonetto, F. Dinuzzo, T. Chen, G. De Nicolao, and L. Ljung, Kernel methods in system identification, machine learning and function estimation: A survey, Automatica **50**, 657 (2014).

[38] C.-C. Chang and C.-J. Lin, LIBSVM: A library for support vector machines, ACM Trans. Intell. Syst. Technol. **2**, 1 (2011).

[39] N. García-Pedrajas and D. Ortiz-Boyer, An empirical study of binary classifier fusion methods for multiclass classification, Inf. Fusion **12**, 111 (2011).

[40] M. Liu, D. Zhang, S. Chen, and H. Xue, Joint binary classifier learning for ECOC-based multi-class classification, IEEE Trans. Pattern Anal. Mach. Intell. **38**, 2335 (2015).

[41] J.-H. Hong and S.-B. Cho, A probabilistic multi-class strategy of one-vs.-rest support vector machines for cancer classification, Neurocomputing **71**, 3275 (2008).

[42] J. Xu, An extended one-versus-rest support vector machine for multi-label classification, Neurocomputing **74**, 3114 (2011).
[43] M. A. Kumar and M. Gopal, Reduced one-against-all method for multiclass SVM classification, Expert Syst. Appl. **38**, 14238 (2011).
[44] R. V. Sharan and T. J. Moir, Noise robust audio surveillance using reduced spectrogram image feature and one-against-all SVM, Neurocomputing **158**, 90 (2015).
[45] H. Faris, M. Habib, M. Faris, M. Alomari, and A. Alomari, Medical speciality classification system based on binary particle swarms and ensemble of one vs. rest support vector machines, J. Biomed. Inform **109**, 103525 (2020).
[46] M. Schuld, Supervised quantum machine learning models are kernel methods, arXiv:2101.11020.
[47] X. Wang, Y. Du, Y. Luo, and D. Tao, Towards understanding the power of quantum kernels in the NISQ era, Quantum **5**, 531 (2021).
[48] T. Hubregtsen, D. Wierichs, E. Gil-Fuster, P.-J. H. Derks, P. K. Faehrmann, and J. J. Meyer, Training quantum embedding kernels on near-term quantum computers, Phys. Rev. A **106**, 042431 (2022).
[49] S. Jerbi, L. J. Fiderer, H. Poulsen Nautrup, J. M. Kübler, H. J. Briegel, and V. Dunjko, Quantum machine learning beyond kernel methods, Nat. Commun. **14**, 517 (2023).
[50] A. E. Paine, V. E. Elfving, and O. Kyriienko, Quantum kernel methods for solving regression problems and differential equations, Phys. Rev. A **107**, 032428 (2023).
[51] B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, Reproducing kernel Hilbert space, Mercer's theorem, eigenfunctions, Nyström method, and use of kernels in machine learning: Tutorial and survey, arXiv:2106.08443.
[52] M. Kelly, R. Longjohn, and K. Nottingham, Iris data set, UCI machine learning repository (2023), https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data.
[53] M. Kelly, R. Longjohn, and K. Nottingham, Tae, The UCI machine learning repository (2023), https://archive.ics.uci.edu/ml/machine-learning-databases/tae/tae.data.
[54] A. M. Horst, A. P. Hill, and K. B. Gorman, palmerpenguins: Palmer Archipelago (Antarctica) penguin data, R package version 0.1.0 (2020), https://raw.githubusercontent.com/allisonhorst/palmerpenguins/master/inst/extdata/penguins.csv.
[55] M. Kelly, R. Longjohn, and K. Nottingham, Glass, The UCI machine learning repository (2023), https://archive.ics.uci.edu/ml/machine-learning-databases/glass/glass.data.
[56] M. Kelly, R. Longjohn, and K. Nottingham, Ecoli, The UCI machine learning repository (2023), https://archive.ics.uci.edu/ml/machine-learning-databases/ecoli/ecoli.data.
[57] M. Kelly, R. Longjohn, and K. Nottingham, Vowel, The UCI machine learning repository (2023), https://archive.ics.uci.edu/ml/machine-learning-databases/undocumented/connectionist-bench/vowel/vowel.data.
[58] S. Patro and K. K. Sahu, Normalization: A preprocessing stage, arXiv:1503.06462.
[59] M. Ringnér, What is principal component analysis? Nat. Biotechnol. **26**, 303 (2008).
[60] J. Shlens, A tutorial on principal component analysis, arXiv:1404.1100.
[61] M. Greenacre, P. J. Groenen, T. Hastie, A. I. d'Enza, A. Markos, and E. Tuzhilina, Principal component analysis, Nat. Rev. Methods Primers **2**, 100 (2022).
[62] G. Aleksandrowicz, T. Alexander, P. Barkoutsos, L. Bello, Y. Ben-Haim, D. Bucher, F. J. Cabrera-Hernndez, J. Carballo-Franquis, A. Chen, C. Chen *et al.*, Qiskit: An open-source framework for quantum computing, Zenodo (2019), https://doi.org/10.5281/zenodo.2562111.
[63] C. Developers, Cirq, Zenodo (2024), https://doi.org/10.5281/zenodo.11398048.
[64] 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.
[65] A. W. Services, Amazon braket: Accelerate quantum computing research, https://aws.amazon.com/braket/ (2022).
[66] D. S. Steiger, T. Häner, and M. Troyer, Projectq: An open source software framework for quantum computing, Quantum **2**, 49 (2018).
[67] P. J. Karalekas, M. P. Harrigan, S. Heidel, W. J. Zeng, M. S. Alam, M. I. Appleby, L. E. Capelluto, G. E. Crooks, M. J. Curtis, E. J. Davis *et al.*, PyQuil: Quantum programming in Python, Zenodo (2020), https://zenodo.org/records/3631770.
[68] M. Bhagat and B. Bakariya, Implementation of logistic regression on diabetic dataset using Train-Test-Split, K-Fold and Stratified K-Fold approach, Natl. Acad. Sci. Lett. **45**, 401 (2022).
[69] S. Prusty, S. Patnaik, and S. K. Dash, SKCV: Stratified K-fold cross-validation on ML classifiers for predicting cervical cancer, Front. Nanotechnol. **4**, 972421 (2022).
[70] T. Mahesh, O. Geman, M. Margala, M. Guduri *et al.*, The stratified K-folds cross-validation and class-balancing methods with high-performance ensemble classifiers for breast cancer classification, Healthcare Anal. **4**, 100247 (2023).
[71] S. Thanasilp, S. Wang, M. Cerezo, and Z. Holmes, Exponential concentration in quantum kernel methods, Nat. Commun. **15**, 5200 (2024).
[72] S. Wang, E. Fontana, M. Cerezo, K. Sharma, A. Sone, L. Cincio, and P. J. Coles, Noise-induced barren plateaus in variational quantum algorithms, Nat. Commun. **12**, 6961 (2021).
[73] A. Luque, A. Carrasco, A. Martín, and A. de Las Heras, The impact of class imbalance in classification performance metrics based on the binary confusion matrix, Pattern Recogn. **91**, 216 (2019).
[74] M. Grandini, E. Bagli, and G. Visani, Metrics for multi-class classification: An overview, arXiv:2008.05756.
[75] B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, KKT conditions, first-order and second-order optimization, and distributed optimization: Tutorial and survey, arXiv:2110.01858.
[76] L. J. Cao, S. S. Keerthi, C. J. Ong, J. Q. Zhang, U. Periyathamby, X. J. Fu, and H. P. Lee, Parallel sequential minimal optimization for the training of support vector machines, IEEE Trans. Neural Netw. **17**, 1039 (2006).
[77] T.-W. Kuan, J.-F. Wang, J.-C. Wang, P.-C. Lin, and G.-H. Gu, Vlsi design of an SVM learning core on sequential minimal optimization algorithm, IEEE Trans. Very Large Scale Integr. Syst. **20**, 673 (2011).
[78] X. Huang, L. Shi, and J. A. Suykens, Sequential minimal optimization for SVM with pinball loss, Neurocomputing **149**, 1596 (2015).
[79] J. Xiao, Y. Fan, R. Sun, and Z.-Q. Luo, Adversarial Rademacher complexity of deep neural networks, arXiv:2211.14966.