

Article

Hardware-Aware Quantum Kernel Design Based on Graph Neural Networks

Fanxu Meng ^{1,*} , Yuxiang Liu ^{2,3,4} , Lu Wang ^{3,4,5} , Sixuan Li ^{4,5} , Zhixiu Han ^{4,5} and Lidong Liu ^{4,5}

- ¹ College of Artificial Intelligence, Nanjing Tech University, Puzhu South Road, Nanjing 211800, China
- ² National Mobile Communications Research Laboratory, Southeast University, Southeast University Road, Nanjing 210096, China
- ³ Purple Mountain Laboratory, Mozhou East Road, Nanjing 211111, China
- ⁴ Frontiers Science Center for Mobile Information Communication and Security, Southeast University, Southeast University Road, Nanjing 210096, China
- ⁵ State Key Laboratory of Millimeter Waves, Southeast University, Southeast University Road, Nanjing 210096, China
- * Correspondence: mengfanxuchina@163.com

Abstract

Quantum kernels hold significant promise for achieving computational advantages in quantum machine learning (QML), yet their effectiveness critically depends on the design of expressive and hardware-compatible feature maps, a challenge that is particularly pronounced in Noisy Intermediate-Scale Quantum (NISQ) devices with limited qubits, gate errors, and restricted connectivity. In this work, we propose a hardware-aware methodology for automated quantum kernel design that integrates quantum device characteristics with learning-based evaluation. Specifically, candidate quantum circuits explored within the hardware-aware circuit space are represented as directed acyclic graphs (DAGs) encoding hardware-specific information such as gate operations, qubit interactions, and noise properties, while a dual graph neural network (GNN) predictor is employed to estimate key surrogate metrics, including probability of successful trials (PST) and kernel-target alignment (KTA), enabling efficient and accurate assessment of circuit fidelity and kernel performance to facilitate the identification of task-specific quantum kernels. Furthermore, feature selection is incorporated to reduce input dimensionality and ensure compatibility with near-term devices. Extensive experiments on multiple benchmark datasets, including Credit Card (CC), MNIST-5, and FMNIST-4, demonstrate that our method consistently outperforms existing baselines in classification accuracy, effectively balancing hardware constraints and model expressivity under realistic noise conditions. These results highlight the potential of combining hardware-aware design with deep learning techniques to advance practical quantum kernel methods under hardware-calibrated noisy simulation settings.

Keywords: quantum machine learning; noisy intermediate-scale quantum; hardware-aware quantum kernel; graph neural network; probability of successful trials; kernel-target alignment



Received: 29 May 2026

Revised: 13 July 2026

Accepted: 15 July 2026

Published: 1 August 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Quantum computing presents a compelling prospect for revolutionizing machine learning tasks by exploiting the fundamental properties of quantum mechanics, such as superposition, entanglement, and high-dimensional Hilbert spaces [1,2]. Quantum Machine Learning (QML), as a prominent class of quantum algorithms, has demonstrated

practical applicability in diverse real-world tasks, including classification [3–7], generative modeling [8–11], and learning physical systems [12,13], particularly in the Noisy Intermediate-Scale Quantum (NISQ) era [14]. Recent studies have further highlighted the rapid expansion of QML from both methodological and application-oriented perspectives, including emerging research paradigms, knowledge structure analyses, future prospects, and hybrid quantum–classical applications in areas such as mental health-aware sentiment analysis [15,16]. Among various QML approaches, quantum kernel methods have attracted increasing attention due to their solid theoretical foundation and interpretability [17–19]. By embedding classical data into quantum states through parameterized quantum circuits, quantum kernels induce implicit feature maps capable of capturing complex nonlinear relationships beyond classical means.

Despite recent demonstrations of quantum kernel learning on real quantum devices, achieving practical quantum advantage remains challenging, particularly under the constraints of NISQ devices [20–22]. Inappropriate quantum circuit designs can lead to vanishing kernel similarities, reduced expressivity, and degraded generalization [23,24], thereby limiting the overall performance of quantum kernels. These issues are further exacerbated in high-dimensional settings or when circuit depth and qubit counts exceed the noise tolerance of near-term hardware. To address these challenges, prior works have explored both circuit-aware and data-aware optimization strategies. Some approaches focus on tuning variational parameters within fixed-circuit architectures [25–27], while others formulate quantum kernel design as a joint optimization over circuit and parameters, typically addressed using evolutionary or machine learning-based approaches [28–31]. However, these methods not only fail to account for hardware-specific constraints, such as qubit connectivity and gate noise, but also incur prohibitive computational costs when exploring large circuit design spaces.

Motivated by these limitations, there remains a critical need for approaches that can efficiently identify high-quality and task-specific quantum kernel while explicitly incorporating hardware-specific constraints. To this end, we propose a hardware-aware quantum kernel design framework based on graph neural networks. Rather than formulating quantum kernel design as a joint optimization over circuit structures and parameters, our approach decouples the process by first generating hardware-aware candidate circuits and subsequently evaluating their performance using graph neural networks (GNNs). By exploiting the graph structure of quantum circuits, the GNNs predict two surrogate metrics: the probability of successful trials (PST), which captures circuit fidelity under hardware noise, and kernel-target alignment (KTA), which correlates with downstream classification accuracy.

The proposed framework comprises four stages: generation of hardware-aware candidate circuits based on device topology and native gate sets, filtering via PST prediction to eliminate low-fidelity circuits, ranking via KTA prediction to identify high-performing kernels, and subsequent parameter optimization of the top candidates. Furthermore, to further enhance scalability, we incorporate feature selection to reduce input dimensionality and ensure compatibility with NISQ devices. Crucially, by integrating hardware information and task-dependent performance estimation within a unified learning framework, the proposed framework enables efficient identification of quantum kernels that are both hardware-compatible and task-relevant. Extensive hardware-calibrated noisy simulations based on IBM device characteristics demonstrate that the proposed approach consistently outperforms state-of-the-art baselines across diverse classification tasks and simulated hardware configurations.

Our method offers several distinctive advantages that collectively enable efficient quantum kernel design for NISQ devices:

- **Hardware awareness:** Our methodology explicitly incorporates hardware topology and noise characteristics during both circuit generation and performance prediction, enabling the identification of robust and high-quality quantum circuits under hardware-calibrated noisy simulation settings.
- **Dual-GNN-based performance prediction:** By employing two graph neural networks (GNN-1 and GNN-2), the proposed method efficiently predicts PST and KTA as a proxy for classification performance, achieving speedups of over $380\times$ and $58,000\times$ compared to direct simulation.
- **Hardware-aware candidate filtering:** By filtering out low-fidelity circuits early in the pipeline, our framework mitigates the impact of gate errors and decoherence, thereby improving the robustness and reliability of quantum kernel performance on noisy devices.
- **Efficient support for high-dimensional input:** The integration of feature selection techniques (e.g., mRMR) reduces input dimensionality by retaining features with high relevance and low redundancy, thereby mitigating excessive circuit depth, alleviating kernel concentration, and improving compatibility with NISQ-scale hardware constraints.
- **Superior classification accuracy:** Extensive evaluations on multiple benchmark tasks, including Credit Card (CC), MNIST-5, and FMNIST-4, as well as under hardware-calibrated noise models derived from IBM devices, demonstrate that the proposed methodology consistently outperforms existing state-of-the-art methods in the considered noisy simulation settings.

The remainder of this paper is organized as follows: Section 2 reviews the necessary background on quantum computing, kernel methods, graph neural networks, and related studies on quantum architecture search and automatic quantum kernel design. Section 3 presents the proposed framework. Section 4 reports the numerical results and comparative evaluations. Finally, Section 5 concludes this paper and outlines future research directions.

2. Background

2.1. Quantum Computing

A quantum bit (qubit) can exist in a linear superposition of the two computational basis states, $|0\rangle$ and $|1\rangle$, in contrast to a classical bit that can only be in a definite state of either 0 or 1. The state of a single qubit is generally represented as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where α and β are complex amplitudes satisfying the normalization condition $|\alpha|^2 + |\beta|^2 = 1$. In multi-qubit systems, the overall quantum state resides in a tensor-product Hilbert space, allowing for entangled states that cannot be factorized into products of single-qubit states. Quantum gate, as a unitary matrix, transforms the quantum state to another and preserves the norm of the quantum state. Any unitary can be decomposed into a set of basis gates. The basis gate sets explored in this study are $S = \{R_x, R_z, CZ, I\}$ and $S^* = \{R_z, X, CNOT, I\}$, which consist primarily of single-qubit and two-qubit gates. More detailed matrix representations can be found in Appendix A.

2.2. Mechanism of Quantum Kernel

Kernel methods provide a powerful theoretical framework for performing nonlinear and nonparametric learning tasks, owing to their generality and strong interpretability [32–34]. Assume that both the training and test samples are drawn from the same input–output space $\mathcal{X} \times \mathcal{Y}$. The training dataset is denoted as $\mathcal{D} = \{x_i, y_i\}_{i=1}^n \subset \mathcal{X} \times \mathcal{Y}$, where x_i represents the input features of the i -th sample and y_i denotes the corresponding label. The fundamental idea of traditional kernel methods is to map input data x_i from $\mathbb{R}^d$ to a higher-dimensional feature space $\mathbb{R}^q$, i.e., $\phi(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^q$ with $q \gg d$, such that data

that are not linearly separable in the original space may become linearly separable in the transformed space. However, when q is large, explicitly computing the mapped feature vectors $\phi(x_i)$ becomes computationally expensive. To address this, kernel methods employ the kernel trick, which enables implicit computation of inner products in the feature space through a kernel function $\kappa(x_i, x_j)$. This leads to the construction of a kernel matrix $K \in \mathbb{R}^{n \times n}$ with entries defined as follows:

$$[K]_{i,j} = \kappa(x_i, x_j), \quad \forall i, j \in [n], \quad (1)$$

where $\kappa(x_i, x_j)$ corresponds to the inner product $\langle \phi(x_i), \phi(x_j) \rangle$ if an explicit feature map exists. Using the kernel matrix, the learning task reduces to finding a hypothesis function of the form $h(x_i) = \langle w^*, \phi(x_i) \rangle$. The optimal parameter w^* is obtained by minimizing the following loss function:

$$\mathcal{L} = \lambda \langle w, w \rangle + \sum_{i=1}^n (\langle w, \phi(x_i) \rangle)^2 \quad (2)$$

The performance of kernel methods critically depends on the choice of the feature map $\phi(\cdot)$, or equivalently, the kernel function $\kappa(\cdot, \cdot)$. A variety of kernels have been proposed for different applications, including radial basis function (RBF) kernels, Gaussian kernels, and quantum kernels.

Compared to classical kernel methods, the key distinction of quantum kernel approaches lies in the use of quantum feature maps that embed classical data into quantum states. For an N -qubit system, a data sample x_i is encoded via a parameterized quantum circuit $U_c(x_i, \theta)$ acting on the initial state $|0\rangle^{\otimes N}$, resulting in the quantum state $|\varphi(x_i, \theta)\rangle = U_c(x_i, \theta)|0\rangle^{\otimes N}$. This defines an implicit feature map into a high-dimensional Hilbert space. The corresponding density operator is given by $\rho(x_i, \theta) = |\varphi(x_i, \theta)\rangle\langle\varphi(x_i, \theta)|$. A commonly used quantum kernel is defined via the Hilbert–Schmidt inner product, leading to the kernel matrix:

$$[W(x, \theta)]_{i,j} = \text{Tr}(\rho(x_i, \theta)\rho(x_j, \theta)), \quad \forall i, j \in [n] \quad (3)$$

which corresponds to the squared overlap between quantum states for pure-state encodings. Using this kernel matrix, learning proceeds analogously to classical kernel methods. The performance of a quantum kernel critically depends on the design of the encoding circuit $U_c(\cdot, \theta)$, which determines the induced feature space. In practice, U_c can take various forms, including predefined ansatz, hardware-efficient circuits, or randomly generated circuits from a given gate set.

2.3. Graph Neural Networks

Graph neural networks have become a powerful framework for learning over graph-structured data, where relationships among entities play a critical role in the underlying data distribution [35]. By iteratively aggregating and transforming information from neighboring nodes through message-passing mechanisms, GNNs are capable of capturing both local topology and global structural dependencies [36]. This capacity to model complex relational patterns has led to their widespread adoption across a variety of domains, including social network analysis [37], molecular property prediction [38–41], and recommender systems [42,43]. Notably, variants such as Graph Convolutional Networks (GCNs), Graph Attention Networks (GATs), and GraphSAGE have significantly advanced the state of the art in graph representation learning tasks [44–48]. In the context of quantum computing, quantum circuits can be naturally represented as graphs, where qubits correspond to nodes and quantum gates or couplings define the edges. This structural analogy provides a

compelling motivation for applying GNNs to quantum systems, enabling efficient learning and prediction tasks grounded in the intrinsic connectivity of quantum architectures.

2.4. Automated Design of Quantum Circuits and Kernels

The performance of quantum machine learning models is strongly affected by the design of quantum circuits. In the NISQ era, this issue becomes more pronounced because quantum circuits must be constructed under limited qubit resources, restricted hardware connectivity, and device noise. Manually designing effective circuits for different tasks is therefore difficult and usually requires considerable domain knowledge. To reduce the dependence on hand-crafted circuit structures, quantum architecture search (QAS) has been explored as an automated strategy for identifying suitable quantum circuits from a predefined search space [49–56].

Existing QAS methods have shown that automated circuit design can improve the performance of quantum learning models under near-term hardware constraints. For example, QuantumSupernet searches candidate quantum architectures within a large circuit space and aims to reduce the cost of evaluating different circuit structures [56]. QuantumNAS further considers noise-adaptive circuit design, where the searched circuit is optimized together with hardware-related factors such as qubit mapping and device noise [7]. Differentiable QAS formulates discrete architecture choices in a differentiable manner, enabling gradient-based circuit architecture optimization [50], while QuantumDARTS further introduces a differentiable search strategy based on Gumbel-Softmax to reduce repeated circuit sampling and evaluation [51]. Predictor-based QAS accelerates the search process by using learned surrogate models to estimate the performance of candidate circuits [52], and graph self-supervised QAS further improves representation learning for circuit structures when labeled circuits are limited [53]. In addition, hierarchical circuit representations have been investigated to connect neural architecture search concepts with quantum circuit design [54], and reinforcement learning-based QAS has been explored for selecting circuit architectures under hardware errors [55]. EliVagar focuses on efficient circuit search for QML classification tasks [6]. It uses low cost proxies, including Clifford noise resilience, to estimate circuit noise robustness and reject low-fidelity circuits at an early stage, thereby reducing the cost of candidate evaluation. These studies indicate that automatic circuit design is useful for improving QML models, especially when the circuit structure and hardware constraints jointly affect the final performance.

Automatic quantum kernel design is closely related to QAS, but its objective is different. In quantum kernel methods, the quantum circuit defines the feature map that embeds classical data into quantum states. The quality of the resulting kernel is determined not only by the expressivity of the circuit, but also by the induced kernel matrix and its relationship with the target labels. Therefore, directly applying circuit search strategies designed for trainable quantum models may not be sufficient for quantum kernel design. KTA provides a useful criterion for measuring the agreement between a kernel matrix and the label information [34,57]. Based on this idea, QuKerNet uses a neural predictor to estimate KTA and select promising quantum kernels more efficiently than exhaustive kernel evaluation [31].

Although these studies have advanced automated circuit and kernel design, several limitations remain. Most QAS methods mainly focus on trainable quantum circuits for QML classification or variational models, while quantum kernel design requires evaluating the task-dependent quality of the induced kernel matrix. Existing automatic quantum kernel design methods have considered KTA-based kernel evaluation, but hardware-related factors, including device topology, native gate constraints, gate errors, readout errors, and circuit fidelity, are not fully incorporated into a unified design framework. In addition,

direct evaluation of a large number of candidate quantum kernels is computationally expensive because kernel construction requires repeated circuit evaluations over pairs of input samples.

3. Method

We now present the proposed framework, illustrated in Figure 1, which consists of four main steps:

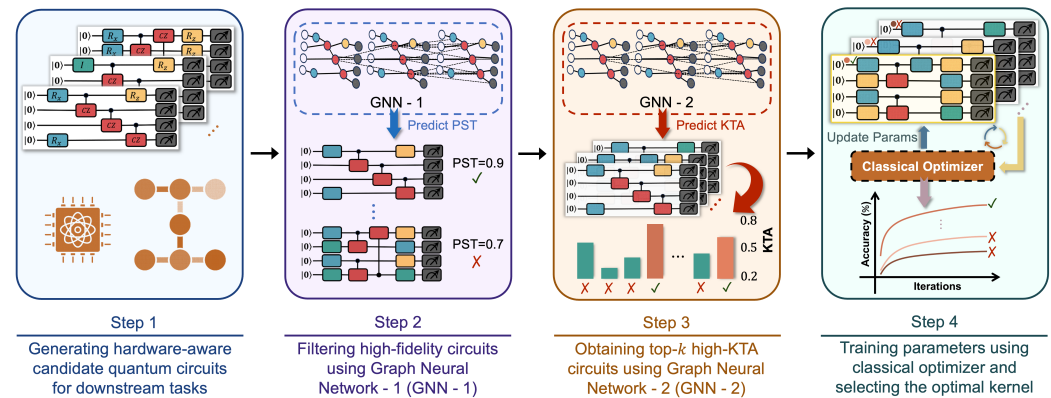


Figure 1. Overview of the proposed framework for hardware-aware quantum kernel design.

Step 1 Candidate Circuits Generation: A subset of qubits is first selected based on device topology and noise characteristics to ensure favorable connectivity and reduced noise impact. Based on this hardware-aware selection, a large set of candidate quantum circuits is generated under the native gate set and connectivity constraints of the target device.

Step 2 PST Prediction: A graph neural network (GNN-1) is employed to predict PST, which serves as a proxy for circuit fidelity. This enables early rejection of low-fidelity candidates, effectively retaining circuits with higher fidelity potential.

Step 3 KTA Prediction: A second graph neural network (GNN-2) is designed to estimate the KTA, a surrogate metric for classification accuracy, over the high-fidelity circuits identified in Step 2. The top- k circuits with the highest predicted KTA values are selected for further refinement.

Step 4 Final Training and Kernel Selection: A classical optimizer is then employed to train the parameters of the top- k circuits, selected based on high KTA and PST values. The circuit achieving the best task-specific classification performance is chosen as the final quantum kernel.

For clarity, the four-step framework is grouped into three components: candidate circuit generation (Step 1, Section 3.1); GNN-based metric prediction (Steps 2–3, Section 3.2); and final training and kernel selection (Step 4, Section 3.3). Although the proposed pipeline contains several interacting components, each component is introduced to address a different bottleneck in hardware-aware quantum kernel design. The candidate circuit generation module defines a hardware-compatible search space by incorporating device topology, native gate constraints, and noise characteristics. The PST predictor is used as a fidelity-oriented filter to remove candidate circuits that are likely to be unreliable under the target noise model. The KTA predictor is then applied to the remaining circuits to estimate the task-dependent quality of the induced quantum kernel. Finally, the downstream training and evaluation stage selects the final kernel according to actual classification performance. Therefore, PST and KTA are not used as final optimization objectives, but as complementary screening criteria for reducing the cost of searching over a large candidate circuit space.

3.1. Candidate Circuit Generation

The generation of candidate quantum circuits consists of three stages. First, a subset of qubits is selected from the quantum device by jointly considering topological connectivity and noise characteristics of both qubits and their couplings, with the goal of identifying a low-noise subgraph that improves hardware efficiency and circuit fidelity. Second, candidate circuits are constructed on the selected qubits under the native gate set and connectivity constraints of the device, where each quantum gate is applied only to physically connected qubits, thereby implicitly satisfying qubit mapping and avoiding the insertion of SWAP gates during compilation, which reduces circuit depth and mitigates noise accumulation. Third, parameterized quantum gates are introduced at different positions within the circuits to enable flexible data embedding, enhancing the expressivity and performance of the resulting quantum kernels.

3.1.1. Subgraph Selection in Device Topology

Due to heterogeneous noise levels across qubits and their couplings in quantum devices, we select a subset of N qubits that are less susceptible to noise, thereby improving hardware efficiency and enhancing circuit fidelity. Figure 2 shows example subgraphs with $N = 4, 5, 7$, and 8 extracted from the 133-qubit IBM Torino topology.

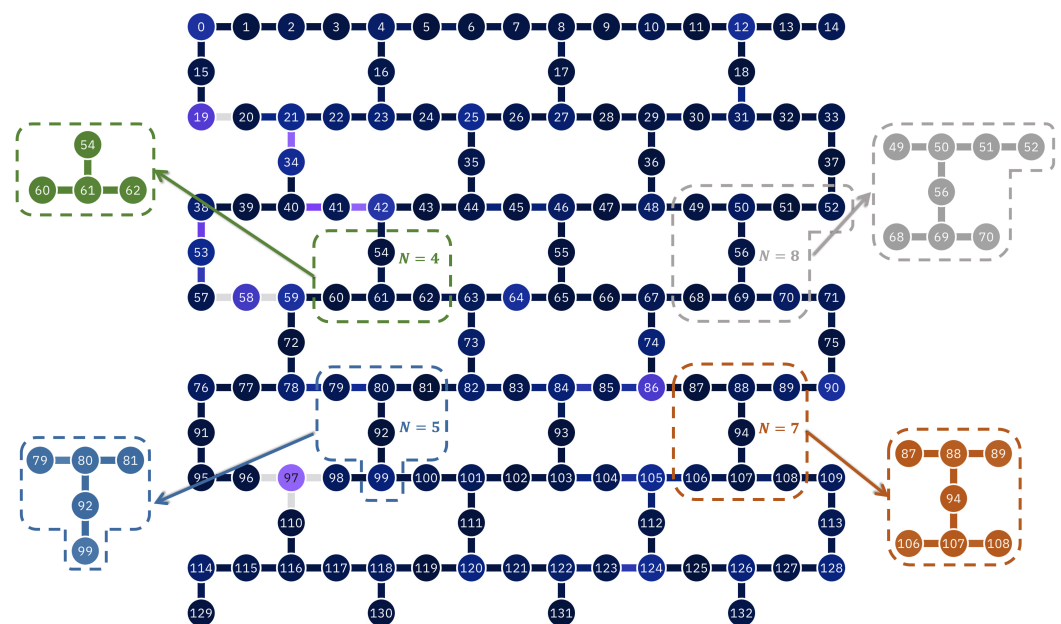


Figure 2. Subgraphs with $N = 4, 5, 7$ and 8 selected from the 133-qubit IBM Torino topology.

Selecting a subgraph with N qubits requires jointly considering multiple hardware-specific factors, including single-qubit gate errors, readout errors, two-qubit gate errors between coupled qubits, and qubit connectivity. The objective is to identify a subgraph with high gate fidelity and strong connectivity while maintaining low readout error. By incorporating device-specific noise information, this selection strategy ensures that the resulting quantum circuits are hardware-aware. Detailed procedures are provided in Appendix B.

3.1.2. Generating Candidate Circuits Based on the Subgraph

After selecting a high-fidelity subgraph G , we generate M candidate quantum circuits under the structural constraints of G and the native gate set S supported by the device, ensuring hardware-aware circuit construction. As illustrated in Step 1 of Figure 1, candidate circuits are generated from a subgraph of the IBM Torino using the gate set $S = \{R_x, R_z, CZ, I\}$. Detailed procedures are provided in Algorithm 1. During circuit

generation, the placement of two-qubit gates between coupled qubits is guided by their gate errors. Specifically, the probability of inserting a two-qubit gate between qubits q_i and q_j , denoted as $P(q_i, q_j)$, is defined as follows:

$$P(q_i, q_j | e_j) = \frac{1/e_j}{\sum_j 1/e_j} \quad (4)$$

where the two-qubit gate error e_{ij} corresponds to the edge $q_i q_j$ in the subgraph G . Single-qubit gate errors are neglected, as two-qubit gate errors are typically an order of magnitude larger and thus dominate circuit performance. Since circuit construction is explicitly guided by noise information, the resulting circuits are inherently hardware-aware.

After generating the candidate circuits, a lightweight optimization step is applied to fuse consecutive parameterized gates whenever possible, reducing circuit redundancy. By iteratively applying the procedure in Algorithm 1 together with this optimization step, a large pool of hardware-aware candidate circuits can be efficiently generated, providing a strong foundation for subsequent GNN-based evaluation and optimal circuit selection.

Algorithm 1 Generate candidate circuit

Require: Device subgraph $G(\{V_i\}, \{E_j\})$; 2-qubit gate error $\{e_j\}$; The gate set S ; The number of gates P .
Ensure: Candidate circuit C .
for k in range(P) **do**
 Sample a gate g_k uniformly from the set S ▷ Sample a gate to be added, e.g., R_x , R_z , or CZ .
 if g_k is a 1-qubit gate **then**
 Select a qubit $q_i \in \{V_i\}$ uniformly at random from the subgraph G ▷ Select a qubit from the subgraph as the target qubit of the quantum gate.
 Append $g_k(q_i)$ to C
 else if g_k is a 2-qubit gate **then**
 Select a pair of qubits q_i, q_j from subgraph G according to the probability distribution $P(q_i, q_j | e_j)$ ▷ 2-qubit gate placement is guided by the associated error e_j .
 Append $g_k(q_i q_j)$ to C
 end if
end for
return C

3.1.3. Data Embedding in Circuits

Inspired by the advantages of randomized data embedding over fixed strategies [6], we adopt a simple yet effective embedding scheme, as illustrated in Figure 3. Specifically, in each quantum circuit, a subset of parameterized single-qubit gates is randomly selected for data encoding, with each selected gate corresponding to one input feature. To ensure full data coverage, the number of embedding gates is set to be no smaller than the input dimensionality. In addition, we prevent consecutive mergeable parameterized gates from encoding different data, thereby avoiding redundant operations after circuit simplification.

For high-dimensional data, such as image classification tasks (e.g., MNIST with 784 features), direct embedding would require a large number of quantum gates, resulting in deep circuits with many qubits. This not only exceeds the capability of NISQ devices but also exacerbates the vanishing kernel similarity issue, leading to degraded performance. To address this, we incorporate a feature selection step in the preprocessing stage to reduce the input dimensionality. Specifically, the max-relevance and min-redundancy (mRMR) method is employed to project the data from d to p dimensions ($p \ll d$), thereby significantly reducing the number of required embedding gates. This strategy mitigates kernel

concentration effects and enables more effective utilization of high-dimensional data on NISQ devices. It should be noted that mRMR is adopted here as a practical preprocessing strategy rather than as an optimal quantum-aware feature selection method. The retained feature dimension is chosen to balance information preservation and NISQ compatibility: retaining more features may preserve more classical information but requires more embedding gates and deeper circuits, whereas retaining fewer features reduces circuit cost but may discard useful information. Since mRMR selects features according to classical relevance and redundancy criteria, it does not guarantee that all features that could be useful for a specific quantum feature map are preserved. Therefore, the feature selection step may introduce information loss, and its influence should be interpreted as part of the trade-off between data representation and hardware feasibility.

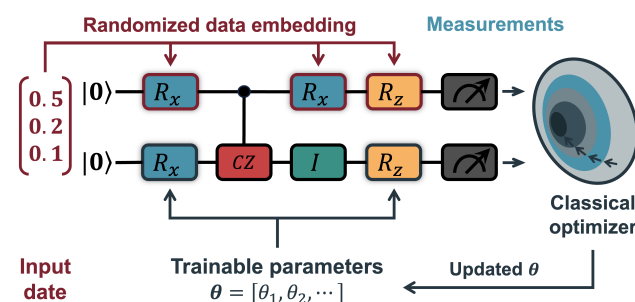


Figure 3. Illustration of randomized data embedding in parameterized quantum circuits.

3.2. Metrics Prediction Based on Graph Neural Networks

To predict the probability of successful trials (PST) under device noise and the corresponding performance metric, kernel-target alignment (KTA), we design two graph neural networks (GNNs), referred to as GNN-1 and GNN-2. Although GNN-1 and GNN-2 share the same graph representation of quantum circuits, they predict two fundamentally different metrics. GNN-1 estimates the PST, which reflects the hardware-dependent fidelity of candidate circuits under the target noise model. In contrast, GNN-2 predicts the KTA, which evaluates the task-dependent quality of the induced quantum kernel through its alignment with the target labels. Since these two prediction targets are governed by different objectives and information sources, we employ two independent GNN models rather than a shared multi-task architecture. This design allows each model to specialize in its corresponding prediction task while providing a clearer interpretation of the two complementary evaluation criteria. While the two models share a similar overall architecture, they differ in specific design choices due to their distinct prediction objectives. Below, we describe the GNN framework in detail, including dataset preparation, graph representation, node features, target definitions, and model architecture.

3.2.1. Construction of the Dataset

The primary motivation for constructing GNN-based predictors is to enable rapid evaluation of a large number of candidate circuits after training on a relatively small dataset. To this end, we adopt a random sampling strategy to construct the training set, where a subset of circuits is selected from the previously generated pool of candidates. To ensure consistency and completeness, all qubits are initialized in the $|0\rangle$ state, and measurement operations are included in each circuit, as illustrated in the left panel of Figure 4. This formulation also facilitates the incorporation of device-specific noise characteristics, such as readout errors, into the node features.

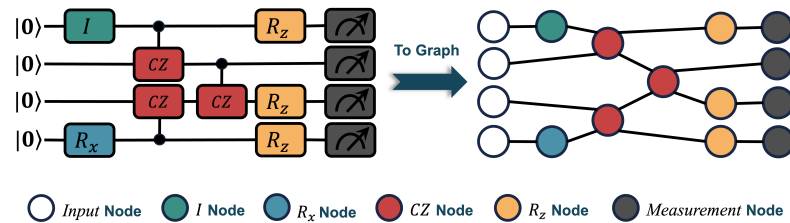


Figure 4. Schematic of quantum circuit-to-graph transformation for GNN-based prediction.

3.2.2. Graph Construction

Motivated by the structural similarity between quantum circuits and graphs, we represent quantum circuits as directed acyclic graphs (DAGs). In this formulation, nodes correspond to qubits, quantum gates, or measurement operations, while edges encode the temporal ordering of operations from left to right. As illustrated in Figure 4, circuits generated on IBM Torino are transformed into graph representations, with each two-qubit gate modeled as a single node to preserve its joint operation. This transformation can also be implemented using existing toolkits, such as Qiskit and PyZX, with appropriate configuration.

3.2.3. Node Features

For each node in the graph, we construct a feature vector to encode its associated properties. The features include the node index, node type, target qubit, tag, relaxation time (T_1), and dephasing time (T_2) of the target qubit, as well as gate error and readout error. The specific feature configurations used for GNN-1 and GNN-2 are illustrated in Figure 5. The feature vectors for both GNN-1 and GNN-2 have a length of 31. It is worth noting that the length of the feature vectors can be manually configured based on either the maximum number of qubits in the circuits or specific requirements, ensuring that each circuit can be properly encoded. In the example shown in Figure 5, we set the maximum supported qubit number to 16, which results in a final feature vector length of 31. The first part of the feature vector encodes the index, which is used for node sorting. The next three parts are one-hot encodings representing the node type, target qubit, and tag, respectively. The node type distinguishes different types of nodes, including initial input qubit, measurement, R_x gate, R_z gate, CZ gate, and identity (I) gate. For example, the one-hot code 000010 indicates that the node corresponds to a CZ gate. The target qubit labels the index of the qubit affected by the operation, and its length can be expanded according to practical needs. The tag distinguishes whether a gate is used for data embedding, switching from 0 to 1 for embedding gates. In addition to capturing the topological connections between circuits, another advantage of GNNs is their ability to incorporate hardware noise information, which is primarily achieved through node feature vectors. In our method, the last seven elements of each node's feature vector are dedicated to describing the noise characteristics of the quantum hardware. During model training, GNN-1 incorporates these noise features to predict the fidelity-related metric, allowing the model to implicitly learn how device noise affects circuit performance, and enabling early rejection of poor-performing circuits. In contrast, GNN-2 focuses on the predicting performance-related metric, with the last 7 elements of its feature vectors zero-filled to exclude noise information. The noise parameters are based on real quantum devices such as IBM Torino, IBM Perth, and IBM Nairobi. Specifically, the first 4 elements represent the relaxation time (T_1) and dephasing time (T_2) for the two involved qubits, the 5th element encodes the gate operation error, and the final two elements capture the readout errors. For nodes without associated noise characteristics, the corresponding values are set to zero. For instance, measurement

nodes have their T_1 , T_2 , and gate error fields set to zero. Figure 5 shows the complete feature vector representation of a CZ gate node for both GNN-1 and GNN-2.

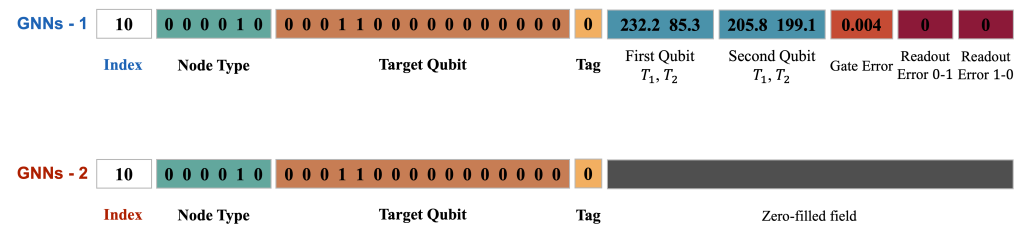


Figure 5. Node feature vectors of GNN-1 and GNN-2 for CZ gate.

3.2.4. Target Values

To evaluate the impact of noise on quantum circuits, we consider fidelity, a standard metric that quantifies the closeness between quantum states. In the presence of noise, fidelity measures the deviation between the state produced by a noisy circuit and its noiseless counterpart. A large deviation indicates that the circuit is significantly affected by noise and may yield unreliable outputs. However, estimating fidelity typically requires quantum state tomography, which is computationally expensive. To circumvent this limitation, we adopt the probability of successful trials (PST) [58] as a proxy metric, which has been shown to provide accurate estimates of circuit fidelity. The PST is defined as follows:

$$PST = \frac{T_{initial}}{T_{total}} \quad (5)$$

where T_{initial} denotes the number of trials that return the initial state $|0\rangle^{\otimes N}$, and T_{total} is the total number of trials. Rather than directly estimating fidelity, we append the inverse of each circuit to itself and apply the resulting circuit to the initial state $|0\rangle^{\otimes N}$. The PST is then obtained as the fraction of measurement outcomes corresponding to the initial state, as illustrated in Figure 6. The resulting PST values are used as target labels for training GNN-1.

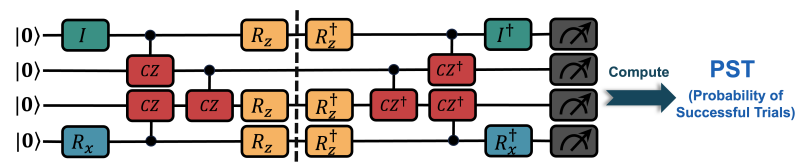


Figure 6. Schematic illustration of PST estimation via circuit inversion.

GNN-2 is designed to predict the kernel-target alignment (KTA), a metric that has been shown to correlate positively with training accuracy [31], and thus serves as a reliable surrogate for model performance. Before introducing KTA, we briefly review the concept of kernel alignment, originally proposed in [34], which measures the similarity between two kernel matrices, K_1 and K_2 :

$$KA(K_1, K_2) = \frac{\langle K_1, K_2 \rangle_F}{\sqrt{\langle K_1, K_1 \rangle_F \langle K_2, K_2 \rangle_F}} \quad (6)$$

where $\langle K_1, K_2 \rangle_F$ denotes the Frobenius inner product between two matrices, which is given by:

$$\langle K_1, K_2 \rangle_F = \sum_{i=1}^l \sum_{j=1}^l \kappa_1(x_i, x_j) \kappa_2(x_i, x_j) \quad (7)$$

where l denotes the size of the kernel matrix, and $\kappa(x_i, x_j) = [K]_{ij}$ represents the (i, j) -th entry of the kernel matrix K . For classification tasks, an ideal target matrix is defined as $K^* = yy^T$, where $y = (y_1, y_2, \dots, y_l)^T$ denotes the label vector of the training set $\mathcal{D}$. The kernel-target alignment (KTA) is then defined as the alignment between the kernel matrix K (e.g., induced by the quantum kernel) and the target matrix K^* :

$$KTA(K) = KA(K, K^*) = \frac{\langle K, yy^T \rangle_F}{\sqrt{\langle K, K \rangle_F \langle yy^T, yy^T \rangle_F}} \quad (8)$$

Since $\langle A, B \rangle_F = \text{Tr}(A^T B)$ and $K^T = K$, substituting them into Equation (8) yields the following formula:

$$KTA(K) = \frac{\text{Tr}(Kyy^T)}{\sqrt{\text{Tr}(K^2)\text{Tr}(yy^T)^2}} = \frac{y^T Ky}{l\sqrt{\text{Tr}(K^2)}} \quad (9)$$

Here, l denotes the length of the label vector y . For multi-class classification tasks (including the binary case), the ideal target matrix K^* can be constructed as follows [57]:

$$[K^*]_{ij} = \begin{cases} 1 & y_i = y_j \\ \frac{-1}{c-1} & y_i \neq y_j \end{cases} \quad (10)$$

where c ($c \geq 2$) denotes the number of classes. KTA provides a task-dependent criterion for evaluating whether a kernel matrix is consistent with the label information. A larger KTA value usually indicates that the induced kernel matrix has a stronger alignment with the target labels on the given dataset. However, KTA should be regarded as a useful proxy rather than a rigorous guarantee of optimal downstream classification performance. The final classification accuracy may also be affected by hardware noise, parameter optimization, and the generalization behavior of the learned decision function. Therefore, in this work, KTA is used to rank and screen candidate quantum kernels instead of serving as a sufficient condition for optimal classification accuracy. Since constructing kernel matrices from quantum circuits remains computationally expensive, we employ GNN-2 to directly predict KTA and use the predicted values for efficient candidate evaluation.

3.2.5. Construction of GNNs

We adopt similar graph neural network (GNN) architectures for GNN-1 and GNN-2, as illustrated in Figure 7. Both models employ a four-layer message-passing structure, with each layer augmented by an attention mechanism to adaptively weight edges during propagation, thereby enhancing the representation of circuit characteristics. LeakyReLU is used as the activation function between layers.

After graph-based feature extraction, average pooling is applied over the representations of key nodes, excluding input and measurement nodes. The pooled features are then concatenated with the output of a three-layer fully connected network that processes global circuit features. The resulting representation is fed into a multilayer perceptron (MLP) to produce the final prediction.

While the overall architectures are shared, GNN-1 and GNN-2 differ in their global feature inputs. Specifically, GNN-1 uses (g_1, k_1, z_1) , where g_1 denotes the total number of gates, k_1 the number of two-qubit gates, and z_1 the circuit depth. In contrast, GNN-2 uses (g_2, k_2, z_2) , where g_2 denotes the total number of gates, k_2 the number of two-qubit gates, and z_2 the number of embedding gates. These task-specific global features align with the distinct prediction objectives of the two models, thereby improving predictive performance.

It should be noted that the proposed framework does not rely on a specific graph neural network. In this work, we adopt a graph attention network-based message passing architecture because the attention mechanism enables adaptive aggregation of neighboring node information with different importance, which is particularly suitable for heterogeneous quantum circuit graphs containing different gate types and hardware attributes. Other graph learning architectures, such as graph convolutional networks (GCNs), GraphSAGE, or graph transformers, may also be applicable. A systematic comparison among different graph, however, is beyond the scope of the present work and will be investigated in future studies.

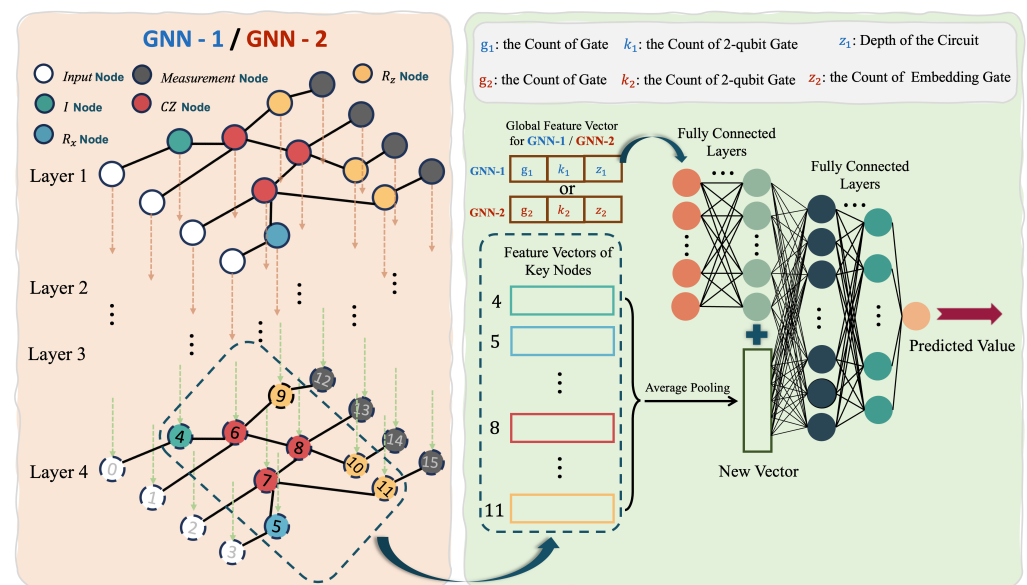


Figure 7. Architecture of GNN-based predictors for quantum circuit evaluation.

3.3. Final Training and Kernel Selection

This step aims to identify the most effective quantum kernel for the target classification task. Based on the results of Steps 2–3, we select the top- k candidate circuits with high predicted PST and KTA values. These circuits are then trained on the target task, where the circuit structure is fixed and only the parameters are optimized from scratch using the Adam optimizer. After training, the classification performance of each circuit is evaluated on the test set, and the circuit achieving the highest accuracy is selected as the final quantum kernel.

It should be emphasized that PST and KTA are used as complementary surrogate criteria for candidate filtering and ranking, rather than as final guarantees of downstream classification performance. PST reflects the fidelity-related robustness of a circuit under hardware noise, while KTA measures the task-dependent alignment between the induced kernel matrix and the label information. These two metrics capture different aspects of a candidate quantum kernel. A circuit with high PST may still have limited discriminative capability if the induced kernel matrix is not well aligned with the target labels. Conversely, a circuit with high KTA may still suffer from hardware noise when implemented on a noisy quantum device. Therefore, the proposed framework does not assume that maximizing PST and KTA always leads to the optimal classifier across all datasets and quantum learning tasks. Instead, PST and KTA are used to reduce the candidate space and retain promising circuits for subsequent training and evaluation. The final quantum kernel is still selected according to the downstream classification performance after parameter optimization.

This design also clarifies the attribution of the final performance. The hardware-aware generation stage, PST-based filtering, KTA-based ranking, and final downstream selection

are not interchangeable modules. Instead, they act at different stages of the pipeline and address different sources of difficulty, including hardware feasibility, noise robustness, task relevance, and final predictive performance. Consequently, the performance gains reported in the experiments should be understood as the result of the complete hardware-aware kernel design pipeline. We do not claim that any single component alone is sufficient to guarantee the observed improvement across all datasets and hardware backends.

4. Experiment

In this section, we present a comprehensive experimental evaluation of the proposed framework. This study is organized into two parts. The first evaluates the predictive performance and computational efficiency of the proposed graph neural networks, GNN-1 and GNN-2 (Section 4.1). The second assesses the overall performance of our methodology against recent state-of-the-art methods across multiple datasets (Section 4.2).

4.1. Predictive Performance of the Graph Neural Networks

Model and Training Setups: In the default setup, both GNN-1 and GNN-2 adopt a four-layer graph neural network architecture with attention mechanisms integrated into each layer, following the design of graph attention networks. This enables adaptive weighting of neighboring nodes during message passing. At the final layer, average pooling is applied over key nodes (excluding input and measurement nodes) to obtain a 31-dimensional representation. This representation is concatenated with a global feature vector produced by a three-layer fully connected network, where each hidden layer has 12 units.

The concatenated vector is then passed through a multilayer perceptron (MLP) with hidden layer sizes of 256, 128, and 64 to generate the final prediction. LeakyReLU with $\alpha = 0.02$ is used as the activation function throughout the network. Training is performed using the Adam optimizer with a learning rate of 0.01, a batch size of 512, and the mean squared error (MSE) as the loss function for 200 epochs. During preprocessing, node features are normalized using Min–Max scaling across the dataset. The predictive performance is evaluated using the coefficient of determination (R^2), defined as follows:

$$R^2 = 1 - \frac{\sum_{i=1}^N (y_i - \hat{y}_i)^2}{\sum_{k=1}^N (y_i - \bar{y})^2} \quad (11)$$

where $\hat{y}_i$ denotes the predicted value for the i -th sample, y_i is the corresponding ground-truth value, $\bar{y}$ is the mean of the ground-truth values, and N is the number of samples in the test set.

The hyperparameters of both GNN models, including the number of graph neural network layers, hidden dimensions, learning rate, batch size, and training epochs, are selected empirically according to the prediction performance on a validation subset of the training data. After this tuning process, the same hyperparameter configuration was fixed for all datasets and hardware settings throughout the experiments to ensure consistency and fair comparison.

Dataset Setup: Following the methodology described in Section 3.1, we generate 50,000 candidate circuits, from which 5000 are randomly sampled to construct the dataset for training GNN-1 and GNN-2. Among these, 4000 circuits are used for training and the remaining 1000 for testing. Although surrogate metrics (PST and KTA) are employed to reduce the cost of label generation, their large-scale computation remains time-consuming, a trade-off that will be further examined in the subsequent runtime analysis.

The main computational bottleneck in constructing the GNN training dataset lies in label generation rather than in GNN inference. For PST labels, each candidate circuit must be concatenated with its inverse circuit and evaluated under the corresponding hardware-

calibrated noise model. The cost therefore increases with the number of candidate circuits, the circuit depth, the number of shots, and the complexity of noisy simulation. For KTA labels, the cost is even higher because constructing a kernel matrix requires repeated evaluations over pairs of input samples. If the training subset used for kernel evaluation contains n samples, each candidate quantum kernel requires $O(n^2)$ circuit evaluations before the KTA value can be computed. This explains why direct KTA evaluation is substantially more expensive than PST estimation. In the proposed framework, these expensive labels are generated only for a sampled subset of candidate circuits to train the surrogate models. Once trained, the GNN predictors can be reused to rapidly evaluate a much larger pool of candidate circuits.

For GNN-1, the noise models used during training are derived from real quantum hardware, including several 7-qubit devices (IBM Perth, IBM Lagos, IBM Nairobi, and IBM Jakarta), as well as the 133-qubit IBM Torino. The considered noise sources include gate errors, relaxation time (T_1), dephasing time (T_2), and readout errors. In addition, all 7-qubit devices share the same native gate set $S^* = \{R_z, X, \text{CNOT}, I\}$.

To evaluate the predictive performance of GNN-1 and GNN-2 for PST and KTA prediction, respectively, we conduct simulations across different system sizes with $N = 4, 5, 7$, and 8 qubits. Although these scales are relatively small, they are sufficient to demonstrate the effectiveness of the proposed approach under NISQ-relevant conditions. Extending the framework to larger quantum systems will be the subject of future work, potentially leveraging tensor-network-based simulations [59] and direct evaluations on real quantum hardware.

As shown in Figure 8, simulations are carried out on corresponding quantum devices: IBM Perth for $N = 4$, and IBM Torino for $N = 5, 7$, and 8. For visualization purposes, 100 samples are randomly selected from the test set in each case, while the coefficient of determination (R^2) is computed over the full test set.

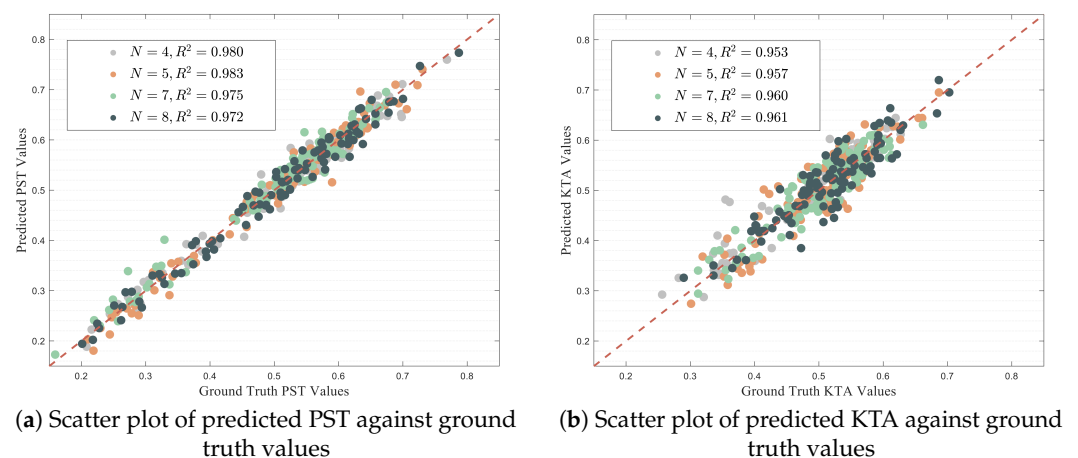


Figure 8. Scatter plots of predicted PST and KTA values with $N = 4, 5, 7$ and 8.

The results in Figure 8 demonstrate that both GNN-1 and GNN-2 achieve high prediction accuracy, with R^2 exceeding 0.97 for PST and 0.95 for KTA. The high accuracy of GNN-1 in predicting PST under hardware noise is attributed to the inclusion of noise information in the node feature vectors, which enhances the model's ability to learn noise-informed representations. Moreover, we observe that KTA values are tightly clustered, while PST values exhibit a broader distribution. This implies that quantum kernels with similar KTA values, i.e., theoretically similar classification performance, can exhibit significantly different results on real hardware due to noise. This observation further justifies our strategy of early rejection of low-fidelity candidate circuits based on PST prediction.

To further demonstrate the efficiency of our approach, we evaluate the computational speedup achieved by GNN-1 and GNN-2 in circuit performance estimation. Specifically, we compare the inference time for 100 GNN-1 predictions of PST with the time required for direct PST computation, and similarly compare the inference time for 100 GNN-2 predictions of KTA with that of direct KTA evaluation. As shown in Figure 9 and summarized in Table 1, the results clearly highlight the substantial efficiency gains of the proposed approach. Note that the y-axis in Figure 9 is presented on a logarithmic scale.

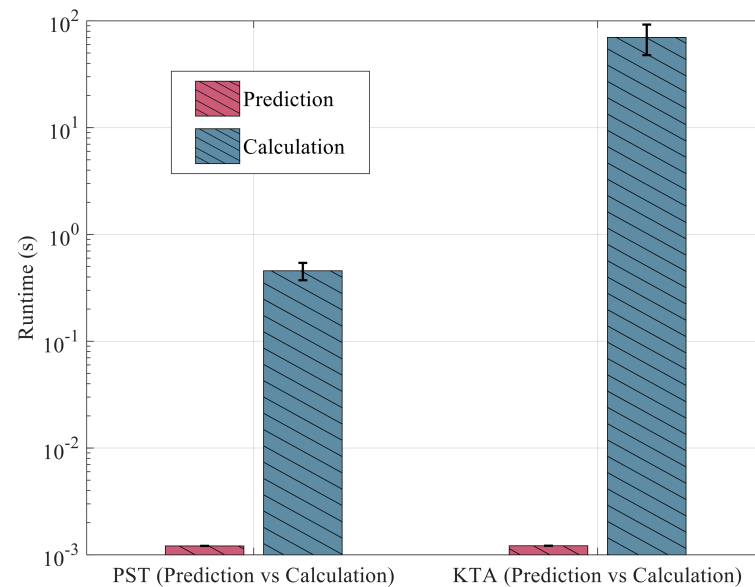


Figure 9. Comparison of runtime of prediction and direct calculation for PST and KTA.

Table 1. Runtime of Prediction and Direct Calculation for PST and KTA.

Metric	Prediction (·s)	Calculation (·s)	Speedup
PST	0.0012 ± 0.0000	0.4572 ± 0.0845	381×
KTA	0.0012 ± 0.0000	70.0540 ± 22.2887	58,378×

Compared to direct computation, GNN-1 and GNN-2 achieve speedups of 381× and 58,378× for PST and KTA prediction, respectively. This comparison reflects the inference cost of the trained GNNs relative to the expense of explicit computation. Rather than a direct algorithmic acceleration, the improvement arises from replacing costly evaluations with learned surrogate models, enabling rapid performance estimation once training is completed.

Such a paradigm is particularly advantageous when direct computation becomes prohibitively expensive, allowing for efficient evaluation of a large number of candidate circuits and making large-scale quantum kernel screening feasible. The computational cost of the GNN models is mainly associated with offline training. For a fixed training dataset, the training cost scales with the number of circuit graphs, the number of graph nodes and edges determined by the circuit size and depth, the hidden dimension of the GNN layers, and the number of training epochs. In contrast, after training is completed, inference only requires a forward pass on each candidate circuit graph and is therefore much cheaper than explicit PST or KTA computation. This distinction is important for scalability; the proposed method does not remove the initial cost of generating representative labeled circuits, but amortizes this cost over the subsequent screening of a much larger candidate pool. Therefore, the framework is most beneficial in scenarios where many candidate circuits need to be evaluated for the same or similar hardware configuration and circuit

generation rule. In addition to computational efficiency, we further examine the scalability of GNN-1 and GNN-2 with respect to the number of qubits, with detailed results provided in Appendix C.

While the above results demonstrate substantial computational speedup, the efficiency gain should be interpreted together with the prediction accuracy of the surrogate models. The GNN predictors are useful only when the approximation errors remain sufficiently small for reliable candidate screening. As shown in Figure 8, both GNN-1 and GNN-2 achieve high prediction accuracy on test circuits, with R^2 exceeding 0.97 for PST and 0.95 for KTA. Therefore, the large runtime reduction reported in Table 1 does not come at the cost of a severe loss of predictive reliability within the considered circuit search space.

It should also be noted that the proposed GNN predictors are used for filtering and ranking candidate circuits, rather than as the final evaluator of quantum kernel performance. After GNN-based prediction, the top- k candidate circuits with high predicted PST and KTA values are further trained and evaluated in the downstream classification task. This design mitigates the influence of occasional prediction errors, since the final quantum kernel is selected based on downstream performance rather than solely on a single predicted score.

4.2. Performance of Our Framework on Benchmarks

Model and Training Setups: After evaluating candidate circuits using GNN-1 and GNN-2, we select the top- k circuits with high KTA and PST values. In our experiments, $k = 10$, corresponding to the top 10 high-performing quantum kernels. These circuits are then fine-tuned by optimizing their parameters from scratch. The training objective is to maximize classification accuracy using support vector machines (SVMs) constructed from the corresponding quantum kernels. Parameter optimization is performed using the Adam optimizer with a learning rate of 0.01.

We further evaluate the proposed method across multiple noisy simulators incorporating hardware-calibrated noise models derived from IBM Perth, IBM Lagos, IBM Nairobi, IBM Jakarta, and IBM Torino. Performance is compared against seven representative baselines. Each method is evaluated over 10 independent runs, and the results are reported as mean values with error bars indicating the standard deviation (SD).

Random Quantum Kernel: A total of 25 quantum kernels are randomly sampled from the gate set, with their circuit depths matched to those generated by the proposed method. Each kernel is trained independently, and the one achieving the highest classification accuracy is selected as the representative quantum kernel for each simulation run.

Training Embedding Kernels (TEK) [25]: This baseline adopts a fixed-circuit structure with parameter optimization. In our setup, each TEK kernel is composed of six blocks, including encoding blocks and training blocks. Each block consists of two layers of single-qubit gates followed by a layer of two-qubit gates arranged in a ring topology. Specifically, on 7-qubit devices, the single-qubit layers comprise an X layer (applying X gates to all qubits) followed by an R_z layer, whereas on the IBM Torino device, an R_x layer is used instead of the X layer, followed by an R_z layer.

In the encoding blocks, dimension-reduced and normalized input features are embedded into the circuit by assigning them to the phases of parameterized gates after scaling by π . Once all features have been encoded, any remaining parameterized gates continue the embedding process by reusing the feature vector in a cyclic manner. The parameters in the training blocks are subsequently optimized to further refine the behavior and performance of the resulting quantum kernel.

Radial Basis Function Kernel (RBFK) [60,61]: A widely used classical kernel with the kernel matrix defined as $[K]_{i,j} = \exp(-\gamma \|x_i - x_j\|^2)$, where γ is a hyperparameter. In the

simulations, γ is set to the reciprocal of the input feature vector dimension, which serves as a reasonable heuristic to balance the sensitivity of the kernel.

QuantumSupernet [56]: Most hyperparameters follow the settings in [56], with one modification: we replace full-batch gradient descent with mini-batch gradient descent (batch size = 32) to ensure efficient and thorough parameter updates during SuperCircuit training. During the construction of the SuperCircuit, we prohibit the occurrence of consecutive quantum gates that can be merged or canceled, such as repeated 1-qubit gates of the same type applied consecutively.

QuantumNAS [7]: We adopt the same hyperparameter settings as in [7] for both SuperCircuit training and evolutionary search. The gate set is adapted to match the constraints of each target quantum device.

EliVagar [6]: This method also incorporates early rejection of low-fidelity circuits. We follow the hyperparameter settings provided in [6] and adapt the gate set to match the supported operations of the target quantum devices.

QuKerNet [31]: This baseline is a recent neural predictor-based approach for quantum kernel design. In our experiments, we follow the hyperparameter settings reported in [31] and simulate the circuits using noise models derived from real quantum hardware. The method employs a multilayer perceptron (MLP) to predict the kernel-target alignment (KTA) of quantum circuits represented in matrix form.

Data encoding follows a sequential scheme, where an initial subset of parameterized gates is used to embed the input features, while the remaining parameters are optimized during training. Similar to our approach, a top- k selection strategy is adopted, with the same value of k used for fair comparison.

To ensure a fair comparison, all quantum baselines are evaluated under the same data preprocessing, qubit-number setting, hardware constraints, and noise assumptions as the proposed method. Specifically, for each benchmark task, the same train/test split, feature selection procedure, and feature normalization are used for all methods. The quantum baselines are configured to use the same number of qubits as the proposed method, and their gate sets are adapted to the native or supported operations of the corresponding target backend whenever applicable. All quantum methods are evaluated using the same hardware-calibrated noisy simulator derived from the target IBM device characteristics. When the logical circuits generated by a baseline are incompatible with the target hardware connectivity, Qiskit's default mapping and routing strategies are applied for compilation. Specifically, logical qubits are mapped to physical qubits in a sequential manner, and SWAP gates are heuristically inserted to satisfy hardware connectivity constraints. The classical RBF kernel baseline does not involve quantum circuits, gate sets, or hardware noise models; therefore, these hardware-related constraints are not applicable to RBFK. Nevertheless, RBFK is evaluated using the same train/test split and the same dimension-reduced input features as the quantum methods.

Dataset Setup: We evaluate the proposed method on three quantum machine learning (QML) benchmark tasks: Credit Card (CC) fraud detection (binary classification), MNIST-5 (five-class classification over digits 0–4), and FMNIST-4 (four-class classification over T-shirt, Trouser, Bag, and Ankle Boot).

For MNIST-5 and FMNIST-4, balanced datasets are constructed by selecting 2000 training samples and 400 test samples from each dataset. The original 784-dimensional images are reduced to 40 dimensions using the minimum Redundancy Maximum Relevance (mRMR) algorithm [31].

For the CC task, 1000 samples are drawn while ensuring that at least 5% correspond to fraudulent transactions. The dataset is randomly split into training and test sets with

a 70/30 ratio. The original 28-dimensional features are reduced to 14 dimensions using mRMR.

The same mRMR-selected features and the same normalized input data are used for the proposed method and all baseline methods so that the comparison is not affected by different feature selection settings. The choice of 14 retained features for the CC task is made to keep the number of embedding gates and the resulting circuit depth compatible with the considered NISQ-scale circuits. We acknowledge that classification performance may depend on the retained feature dimension. Increasing the number of retained features may preserve more information but would also increase the circuit depth and noise accumulation, whereas using fewer features may reduce circuit cost at the expense of losing task-relevant information. A systematic sensitivity analysis over different retained feature dimensions is therefore an important direction for future work.

The three benchmark datasets are selected to cover different data modalities and classification settings under NISQ-compatible experimental conditions. The CC task represents tabular binary classification with class imbalance, while MNIST-5 and FMNIST-4 represent multi-class image classification tasks with different visual patterns and feature distributions. These datasets therefore provide representative test cases for evaluating whether the proposed hardware-aware quantum kernel design framework can work across both tabular and image-based QML benchmarks. Nevertheless, we acknowledge that these datasets do not cover all possible quantum learning scenarios. Therefore, the experimental results should be interpreted as evidence of effectiveness on representative benchmark tasks, rather than as a complete demonstration of broad generalizability across arbitrary datasets and quantum learning tasks.

To validate the performance advantage of the quantum kernels discovered by our framework, we conduct experiments on multiple benchmark tasks under different hardware configurations.

For the 4-qubit Credit Card (CC) task, experiments are performed on two IBM quantum devices with different scales: a 7-qubit device (IBM Perth) and a 133-qubit device (IBM Torino), where only a 4-qubit subset is used in each case. As shown in Figure 10, the proposed methodology achieves the highest classification accuracy among all compared methods, with more pronounced improvements on the larger IBM Torino device. The more pronounced improvement observed under the IBM Torino-derived noise model reflects the combined operation of hardware-aware subgraph selection and circuit generation, PST- and KTA-based candidate screening, and downstream parameter optimization.

We further evaluate the 7-qubit MNIST-5 task using five hardware-calibrated noisy simulators derived from IBM Perth, IBM Lagos, IBM Nairobi, IBM Jakarta, and IBM Torino, where a 7-qubit subset is used for each simulated backend. As shown in Figure 11, our method consistently outperforms all baselines in terms of classification accuracy, while also exhibiting improved stability as indicated by smaller error bars. On 7-qubit devices, the limited hardware scale constrains the effectiveness of hardware-aware qubit selection due to the prevalence of noisy qubits. Nevertheless, the complete framework maintains strong performance across these simulated backends. Under the IBM Torino-derived noise model, the performance advantage becomes more pronounced, suggesting that the integrated pipeline can more effectively utilize the heterogeneous topology and noise profile represented in the simulated backend.

For the FMNIST-4 task, which requires 8 qubits, experiments are conducted on IBM Torino. As shown in Figure 12, the proposed scheme continues to achieve significantly higher classification accuracy compared to competing methods, further demonstrating its effectiveness in larger and more complex settings.

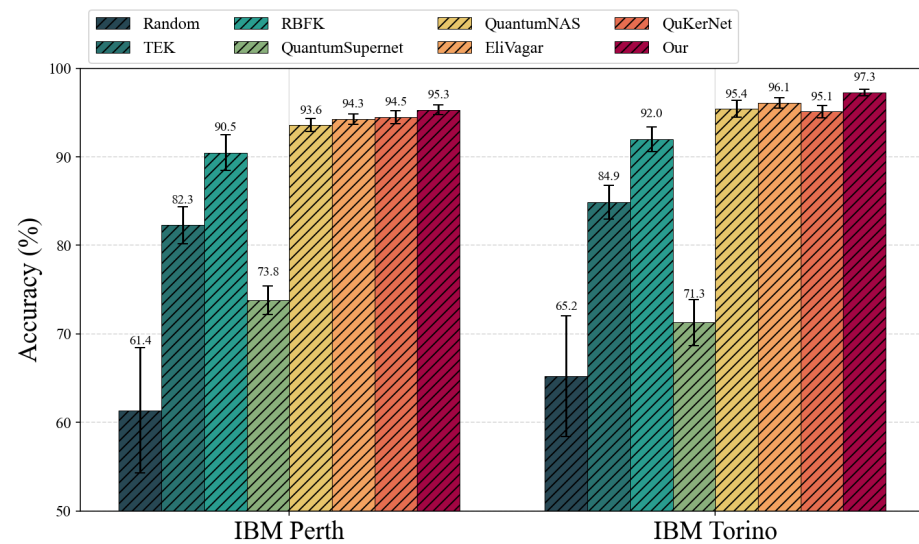


Figure 10. Classification accuracy of different methods on the Credit Card (CC) task under noisy simulation.

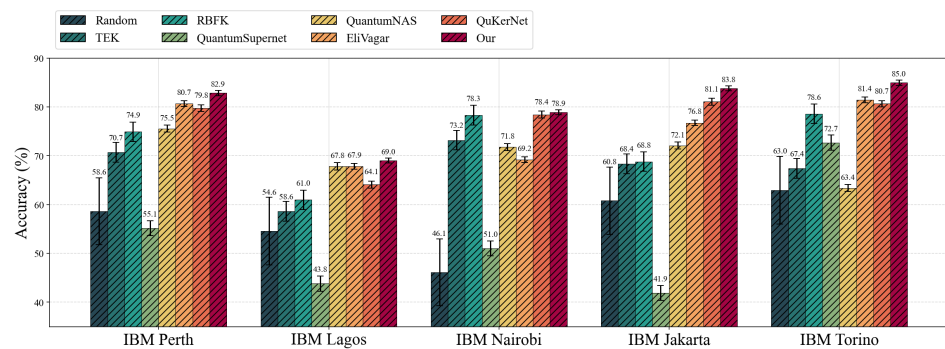


Figure 11. Classification accuracy of different methods on the MNIST-5 classification task under noisy simulation.

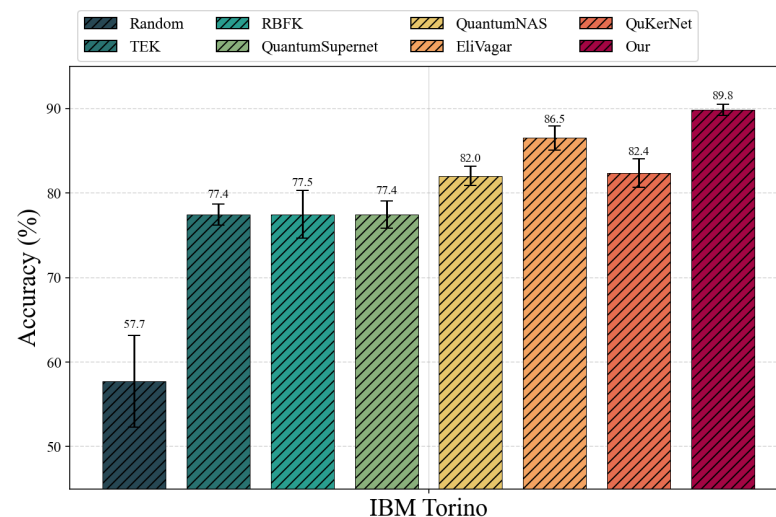


Figure 12. Classification accuracy of different methods on the FMNIST-4 classification task under noisy simulation.

5. Conclusions

In this work, we proposed a hardware-aware framework for quantum kernel design that leverages graph neural networks to evaluate and select task-specific quantum

circuits under the constraints of NISQ hardware. By incorporating real device topology, noise characteristics, feature selection, and GNN-based surrogate evaluation into a unified workflow, the proposed approach provides an efficient strategy for constructing quantum kernels within the considered NISQ-scale settings. The dual graph neural network models enable fast prediction of circuit fidelity and kernel-related performance metrics, thereby reducing the cost of candidate circuit screening. The experimental results on representative classification benchmarks suggest that the integrated hardware-aware workflow can improve quantum kernel performance under hardware-calibrated noisy simulation settings. Accordingly, the observed improvements should be interpreted as the outcome of the integrated framework, rather than as evidence of the independent contribution of any individual component.

These results further indicate that combining learning-based strategies with hardware-aware design can help narrow the gap between theoretical quantum kernel models and hardware-constrained implementations. Nevertheless, hardware-calibrated noise models remain an approximation of real device behavior. Although they provide a more realistic evaluation setting than ideal noiseless simulations, they cannot fully capture all effects present on physical quantum processors, including calibration drift, crosstalk, shot-to-shot fluctuations, and backend queue-time variations. Therefore, practical deployment on real noisy quantum devices still requires further validation through direct hardware experiments when sufficient device access and stability are available.

The proposed framework also has several limitations. First, its scalability is limited by the cost of generating reliable PST and KTA labels for representative candidate circuits. Although GNN inference is efficient after training, label generation and model recalibration may be required when moving to substantially larger circuit sizes, deeper search spaces, or different quantum hardware. Second, the current feature selection stage relies on mRMR, which is a classical relevance-redundancy criterion rather than a quantum-aware feature selection strategy. While this step helps reduce circuit depth and improve compatibility with NISQ-scale devices, it may discard features that are useful for certain quantum feature maps.

Future work will therefore focus on reducing the cost of label generation, improving transferability across circuit sizes and hardware platforms, and evaluating the framework under more diverse hardware-constrained settings. In particular, the scalability results in Appendix C indicate that prediction accuracy may degrade when models trained on smaller circuits are directly applied to larger qubit regimes. This suggests that future large-scale applications will require representative labeled circuits from the target regime, together with retraining, fine-tuning, or transfer-learning strategies to maintain reliable surrogate prediction. Moreover, we will explore the generalization of learned quantum kernels across broader datasets, different feature distributions, and more diverse quantum learning tasks. Extending the framework to quantum systems beyond the reach of classical simulation remains an important long-term direction, but such deployment will require direct validation on real quantum hardware and careful consideration of device access, calibration stability, and time-dependent noise effects.

Author Contributions: Conceptualization, F.M. and L.L.; methodology, F.M.; software, F.M.; validation, F.M., Y.L. and L.W.; formal analysis, F.M.; investigation, F.M., Z.H. and S.L.; resources, L.L.; data curation, F.M.; writing original draft preparation, F.M.; writing review and editing, Y.L., L.W. and L.L.; visualization, F.M.; supervision, L.L.; project administration, L.L.; funding acquisition, L.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Jiangsu Funding Program for Excellent Postdoctoral Talent, grant number 2022ZB139; the Natural Science Foundation of Jiangsu Province Higher Education Basic Science, grant number 24KJB120005; the Frontier Technologies R&D Program of Jiangsu, grant number BF2025066; and the Open Project of the State Key Laboratory of Millimeter Waves, grant number KN202502-09.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

QML	Quantum Machine Learning
NISQ	Noisy Intermediate-Scale Quantum
GNN	Graph Neural Network
DAG	Directed Acyclic Graph
PST	Probability of Successful Trials
KTA	Kernel-Target Alignment
CC	Credit Card
SVM	Support Vector Machine
RBFK	Radial Basis Function Kernel
RBF	Radial Basis Function

Appendix A. Quantum Gate Matrix Representations

We use the gate sets $S = \{R_x, R_z, CZ, I\}$ and $S^* = \{R_z, X, CNOT, I\}$, which primarily consist of the 1-qubit gates I , X , R_x and R_z , whose matrix representations are given as follows:

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad (A1)$$

$$R_x(\theta) = \begin{pmatrix} \cos \frac{\theta}{2} & -i \sin \frac{\theta}{2} \\ -i \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{pmatrix} \quad R_z(\theta) = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix} \quad (A2)$$

In addition, the 2-qubit gates $CNOT$ and CZ are also included, with their corresponding matrix representations defined as follows:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad CZ = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \quad (A3)$$

Appendix B. Strategy for Selecting Subgraphs from the Device Topology

In our investigation of quantum device noise characteristics, we observed a rapid decay in certain types of errors for large-scale quantum devices, such as the 133-qubit IBM Torino. Specifically, the 1-qubit gate error, 2-qubit gate error, and readout assignment error exhibit a sharp decline when ranked across qubits. As illustrated in Figure A1a presents the

R_x gate errors sorted in descending order, while Figure A1b shows the readout assignment errors also sorted from highest to lowest.

From Figure A1a, we observe that after excluding the top 14 qubits with the highest R_x gate errors, the remaining 119 qubits exhibit relatively uniform R_x errors. A similar trend is observed in Figure A1b: after excluding the top 18 qubits with the highest readout errors, the error differences among the remaining qubits become much smaller. Based on these observations, we propose a subgraph selection algorithm to identify qubits with more consistent noise characteristics. The detailed procedure is presented in Algorithm A1. The main idea of Algorithm A1 is to remove vertices (qubits) and connecting edges that are heavily affected by noise from the device topology, resulting in a subgraph with low noise impact.

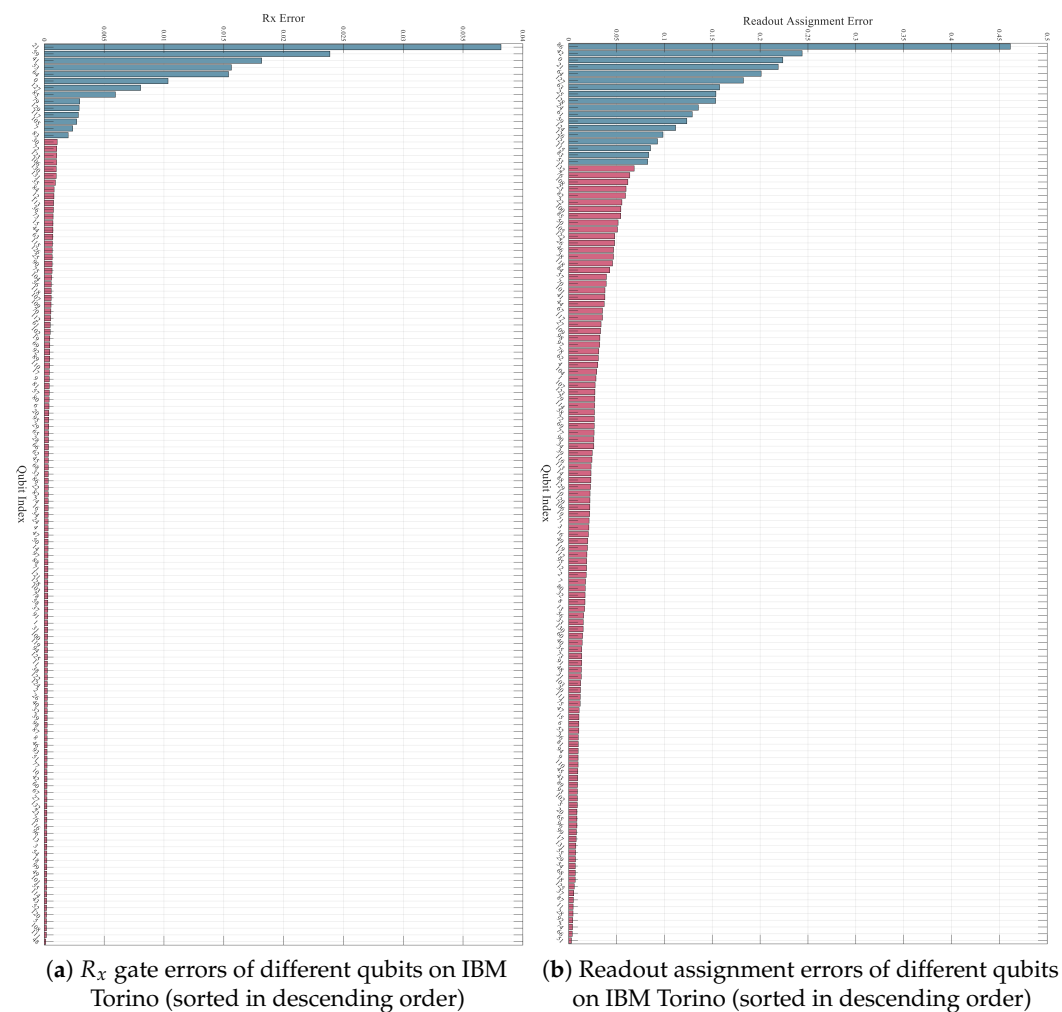


Figure A1. Noise characteristics of different qubits on IBM Torino. In each panel, the horizontal axis denotes the corresponding hardware error rate, while the vertical axis lists the original physical qubit indices arranged according to the sorted error values. The blue bars highlight the high-error qubits removed during the noise-aware preprocessing step, whereas the pink bars indicate the remaining qubits retained as candidates for subsequent subgraph selection. Specifically, the top 14 qubits with the largest R_x gate errors are excluded in panel (a), and the top 18 qubits with the largest readout assignment errors are excluded in panel (b). The sharp decay of the leading bars shows that only a relatively small subset of physical qubits exhibits significantly larger errors, while the remaining qubits have more uniform noise levels.

Algorithm A1 Select subgraph from the device topology

Require: Device topology $G^* (\{V_i^*\}, \{E_j^*\})$; Excluded value E_{xc} ; The number of qubits N ; Noise types T_{noise} .

Ensure: Subgraph G .

```

1: for  $type$  in  $range(T_{noise})$  do
2:   if  $type$  related to a single qubit then
3:     Sort all qubits by error in descending order and select the top  $E_{xc}$  qubits with
     the highest error, denoting their associated vertices and edges as  $\{V^*\}$  and  $\{E^*\}$ .
4:      $G^* = G^* \setminus (\{V^*\}, \{E^*\})$   $\triangleright$  Remove  $\{V^*\}$  and  $\{E^*\}$  from topology  $G^*$ .
5:   else if  $type$  related to two qubits then
6:     Sort all connections by error in descending order and select the top  $E_{xc}$  connec-
     tions with the highest error, denoting their associated edges as  $\{E^*\}$ .
7:      $G^* = G^* \setminus \{E^*\}$   $\triangleright$  Remove  $\{E^*\}$  from topology  $G^*$ .
8:   end if
9: end for
10: Select a subgraph  $G$  from the topology  $G^*$  with  $N$  qubits and high degree of connectivity.
     $\triangleright$  The degree of connectivity is defined as the maximum vertex degree.
11: return Subgraph  $G$ 

```

Appendix C. Qubit Scalability of GNN-1 and GNN-2

To investigate the qubit scalability of GNN-1 and GNN-2, we conducted a simulation experiment based on the IBM Torino device, which features 133 qubits. Specifically, we extended the “Target Qubit” component of the node feature vectors used in Figure 5 from dimension 16 to 21 to accommodate larger subgraphs. Figure A2 presents the scalability performance of GNN-1 for predicting PST and GNN-2 for predicting KTA across varying qubit numbers.

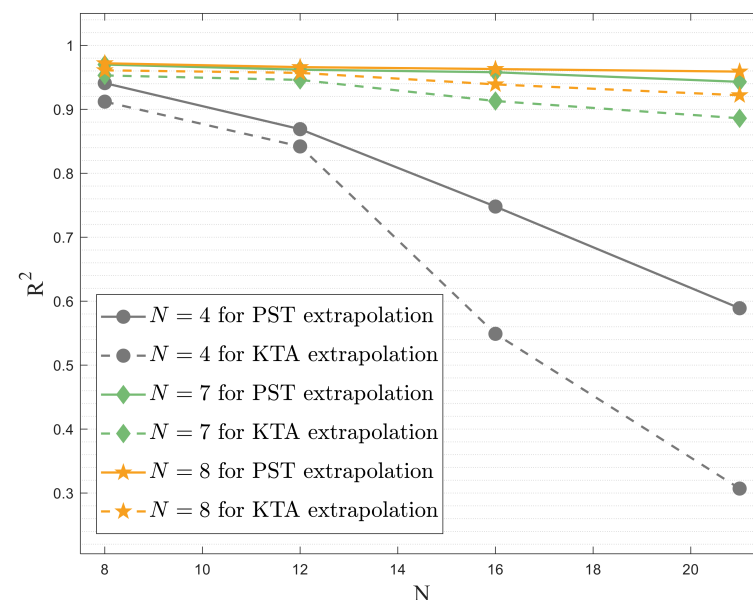


Figure A2. Qubit scalability of GNN-1 and GNN-2 for PST and KTA prediction.

Overall, we observe that as the number of qubits increases, models trained on smaller subgraphs experience notable performance degradation when applied to circuits with more qubits. In particular, models trained on $N = 4$ qubits show a sharp decline in predictive accuracy as qubit count increases. In contrast, models trained on larger subgraphs (e.g., $N = 7$ or 8) exhibit only a gradual decline in performance, suggesting improved scalability. These findings imply the existence of a threshold in model generalizability across qubit

counts, aligning with the results reported in [62], where convolutional neural networks (CNNs) were used to predict expectation values of quantum circuits.

These results also indicate that scalability should be understood together with the distribution covered by the training data. When a model trained on small qubit circuits is directly applied to larger circuits, the graph size, possible gate patterns, and hardware noise configurations may differ from those observed during training. This distribution shift can lead to degraded prediction accuracy, especially when the qubit number and circuit depth increase simultaneously. Therefore, for larger quantum systems or substantially different hardware backends, representative labeled circuits from the target regime should be included for retraining or fine-tuning the GNN predictors. In this sense, the proposed framework is scalable as a surrogate-assisted screening strategy, but its predictive reliability depends on whether the training data cover the relevant circuit size and hardware noise distributions. These observations indicate that improving cross-size generalization and transferability is an important direction for future work.

References

1. Feynman, R.P. Simulating physics with computers. In *Feynman and Computation*; CRC Press: Boca Raton, FL, USA, 2018; pp. 133–153.
2. Biamonte, J.; Wittek, P.; Pancotti, N.; Rebentrost, P.; Wiebe, N.; Lloyd, S. Quantum machine learning. *Nature* **2017**, *549*, 195–202. [[CrossRef](#)] [[PubMed](#)]
3. Farhi, E.; Neven, H. Classification with quantum neural networks on near term processors. *arXiv* **2018**, arXiv:1802.06002.
4. Huang, H.-Y.; Broughton, M.; Mohseni, M.; Babbush, R.; Boixo, S.; Neven, H.; McClean, J.R. Power of data in quantum machine learning. *Nat. Commun.* **2021**, *12*, 2631. [[CrossRef](#)] [[PubMed](#)]
5. Lloyd, S.; Schuld, M.; Ijaz, A.; Izaac, J.; Killoran, N. Quantum embeddings for machine learning. *arXiv* **2020**, arXiv:2001.03622.
6. Anagolum, S.; Alavisamani, N.; Das, P.; Qureshi, M.; Shi, Y. Élivágar: Efficient quantum circuit search for classification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*; Association for Computing Machinery: New York, NY, USA, 2024; pp. 336–353.
7. Wang, H.; Ding, Y.; Gu, J.; Lin, Y.; Pan, D.Z.; Chong, F.T.; Han, S. Quantumnas: Noise-adaptive search for robust quantum circuits. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*; IEEE: Piscataway, NJ, USA, 2022; pp. 692–708.
8. Huang, H.L.; Du, Y.; Gong, M.; Zhao, Y.; Wu, Y.; Wang, C.; Li, S.; Liang, F.; Lin, J.; Xu, Y.; et al. Experimental quantum generative adversarial networks for image generation. *Phys. Rev. Appl.* **2021**, *16*, 24051. [[CrossRef](#)]
9. Niu, M.Y.; Zlokapa, A.; Broughton, M.; Boixo, S.; Mohseni, M.; Smelyanskiy, V.; Neven, H. Entangling quantum generative adversarial networks. *Phys. Rev. Lett.* **2022**, *128*, 220505. [[CrossRef](#)]
10. Rudolph, M.S.; Toussaint, N.B.; Katarbawa, A.; Johri, S.; Peropadre, B.; Perdomo-Ortiz, A. Generation of high-resolution handwritten digits with an ion-trap quantum computer. *Phys. Rev. X* **2022**, *12*, 31010. [[CrossRef](#)]
11. Zoufal, C. Generative quantum machine learning. *arXiv* **2021**, arXiv:2111.12738.
12. Huang, H.-Y.; Broughton, M.; Cotler, J.; Chen, S.; Li, J.; Mohseni, M.; Neven, H.; Babbush, R.; Kueng, R.; Preskill, J.; et al. Quantum advantage in learning from experiments. *Science* **2022**, *376*, 1182–1186. [[CrossRef](#)] [[PubMed](#)]
13. Liu, Y.; Arunachalam, S.; Temme, K. A rigorous and robust quantum speed-up in supervised machine learning. *Nat. Phys.* **2021**, *17*, 1013–1017. [[CrossRef](#)]
14. Preskill, J. Quantum computing in the NISQ era and beyond. *Quantum* **2018**, *2*, 79. [[CrossRef](#)]
15. Rawat, K.S.; Sharma, T. Emerging paradigm of quantum machine learning: Knowledge insights and future prospects. In *IEEE Transactions on Artificial Intelligence*; IEEE: Piscataway, NJ, USA, 2026; pp. 1–15.
16. Rawat, K.S.; Pandey, G.; Sharma, T. Mental health-aware sentiment analysis using a hybrid quantum–classical approach. *J. Supercomput.* **2026**, *82*, 154. [[CrossRef](#)]
17. Schuld, M.; Killoran, N. Quantum machine learning in feature hilbert spaces. *Phys. Rev. Lett.* **2019**, *122*, 40504. [[CrossRef](#)]
18. Havlíček, V.; Córcoles, A.D.; Temme, K.; Harrow, A.W.; Kandala, A.; Chow, J.M.; Gambetta, J.M. Supervised learning with quantum-enhanced feature spaces. *Nature* **2019**, *567*, 209–212. [[CrossRef](#)] [[PubMed](#)]
19. Blank, C.; Park, D.K.; Rhee, J.K.; Petruccione, F. Quantum classifier with tailored quantum kernel. *npj Quantum Inf.* **2020**, *6*, 41. [[CrossRef](#)]
20. Preskill, J. Quantum computing in the NISQ era and beyond. *Bull. Am. Phys. Soc.* **2019**, *64*, 9.

21. Bharti, K.; Cervera-Lierta, A.; Kyaw, T.H.; Haug, T.; Alperin-Lea, S.; Anand, A.; Degroote, M.; Heimonen, H.; Kottmann, J.S.; Menke, T.; et al. Noisy intermediate-scale quantum algorithms. *Rev. Mod. Phys.* **2022**, *94*, 15004. [\[CrossRef\]](#)
22. Cerezo, M.; Arrasmith, A.; Babbush, R.; Benjamin, S.C.; Endo, S.; Fujii, K.; McClean, J.R.; Mitarai, K.; Yuan, X.; Cincio, L.; et al. Variational quantum algorithms. *Nat. Rev. Phys.* **2021**, *3*, 625–644. [\[CrossRef\]](#)
23. Thanasilp, S.; Wang, S.; Cerezo, M.; Holmes, Z. Exponential concentration and untrainability in quantum kernel methods. *Nat. Commun.* **2024**, *15*, 5200. [\[CrossRef\]](#) [\[PubMed\]](#)
24. Wang, X.; Du, Y.; Luo, Y.; Tao, D. Towards understanding the power of quantum kernels in the NISQ era. *Quantum* **2021**, *5*, 531. [\[CrossRef\]](#)
25. Hubregtsen, T.; Wierichs, D.; Gil-Fuster, E.; Derks, P.H.S.; Faehrmann, P.K.; Meyer, J.J. Training quantum embedding kernels on near-term quantum computers. *Phys. Rev. A* **2022**, *106*, 42431. [\[CrossRef\]](#)
26. Glick, J.R.; Gujarati, T.P.; Corcoles, A.D.; Kim, Y.; Kandala, A.; Gambetta, J.M.; Temme, K. Covariant quantum kernels for data with group structure. *Nat. Phys.* **2024**, *20*, 479–483. [\[CrossRef\]](#)
27. Gentinetta, G.; Sutter, D.; Zoufal, C.; Fuller, B.; Woerner, S. Quantum kernel alignment with stochastic gradient descent. In *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*; IEEE: Piscataway, NJ, USA, 2023; Volume 1, pp. 256–262.
28. Altares-López, S.; Ribeiro, A.; García-Ripoll, J.J. Automatic design of quantum feature maps. *Quantum Sci. Technol.* **2021**, *6*, 45015. [\[CrossRef\]](#)
29. Pellow-Jarman, R.; Pillay, A.; Sinayskiy, I.; Petruccione, F. Hybrid genetic optimization for quantum feature map design. *Quantum Mach. Intell.* **2024**, *6*, 45. [\[CrossRef\]](#)
30. Torabian, E.; Krems, R.V. Compositional optimization of quantum circuits for quantum kernels of support vector machines. *Phys. Rev. Res.* **2023**, *5*, 13211. [\[CrossRef\]](#)
31. Lei, C.; Du, Y.; Mi, P.; Yu, J.; Liu, T. Neural auto-designer for enhanced quantum kernels. *arXiv* **2024**, arXiv:2401.11098.
32. Mohri, M.; Rostamizadeh, A.; Talwalkar, A. *Foundations of Machine Learning*; MIT Press: Cambridge, MA, USA, 2018.
33. Aronszajn, N. Theory of reproducing kernels. *Trans. Am. Math. Soc.* **1950**, *68*, 337–404. [\[CrossRef\]](#)
34. Cristianini, N.; Shawe-Taylor, J.; Elisseeff, A.; Kandola, J. On kernel-target alignment. *Adv. Neural Inf. Process. Syst.* **2001**, *14*, 367–373.
35. Corso, G.; Stark, H.; Jegelka, S.; Jaakkola, T.; Barzilay, R. Graph neural networks. *Nat. Rev. Methods Prim.* **2024**, *4*, 17. [\[CrossRef\]](#)
36. Khemani, B.; Patil, S.; Kotecha, K.; Tanwar, S. A review of graph neural networks: Concepts, architectures, techniques, challenges, datasets, applications, and future directions. *J. Big Data* **2024**, *11*, 18. [\[CrossRef\]](#)
37. Zhang, Y. Social network user profiling for anomaly detection based on graph neural networks. *arXiv* **2025**, arXiv:2503.19380.
38. He, Z.; Chen, H.; Zhou, Y.; Situ, H.; Li, Y.; Li, L. Self-supervised representation learning for bayesian quantum architecture search. *Phys. Rev. A* **2025**, *111*, 32403. [\[CrossRef\]](#)
39. Wieder, O.; Kohlbacher, S.; Kuenemann, M.; Garon, A.; Ducrot, P.; Seidel, T.; Langer, T. A compact review of molecular property prediction with graph neural networks. *Drug Discov. Today Technol.* **2020**, *37*, 1–12. [\[CrossRef\]](#) [\[PubMed\]](#)
40. Chen, S.; Wulamu, A.; Zou, Q.; Zheng, H.; Wen, L.; Guo, X.; Chen, H.; Zhang, T.; Zhang, Y. MD-GNN: A mechanism-data-driven graph neural network for molecular properties prediction and new material discovery. *J. Mol. Graph. Model.* **2023**, *123*, 108506. [\[CrossRef\]](#) [\[PubMed\]](#)
41. Shui, Z.; Karypis, G. Heterogeneous molecular graph neural networks for predicting molecule properties. In *2020 IEEE International Conference on Data Mining (ICDM)*; IEEE: Piscataway, NJ, USA, 2020; pp. 492–500.
42. Wu, S.; Sun, F.; Zhang, W.; Xie, X.; Cui, B. Graph neural networks in recommender systems: A survey. *ACM Comput. Surv.* **2022**, *55*, 1–37. [\[CrossRef\]](#)
43. Gao, C.; Wang, X.; He, X.; Li, Y. Graph neural networks for recommender system. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*; Association for Computing Machinery: New York, NY, USA, 2022; pp. 1623–1625.
44. Mahdi, A.S.; Shati, N.M. A survey on fake news detection in social media using graph neural networks. *J. Al-Qadisiyah Comput. Sci. Math.* **2024**, *16*, 23–41. [\[CrossRef\]](#)
45. Yang, H.; Li, Z.; Qi, Y. Predicting traffic propagation flow in urban road network with multi-graph convolutional network. *Complex Intell. Syst.* **2024**, *10*, 23–35. [\[CrossRef\]](#)
46. Guan, R.; Li, Z.; Tu, W.; Wang, J.; Liu, Y.; Li, X.; Tang, C.; Feng, R. Contrastive multi-view subspace clustering of hyperspectral images based on graph convolutional networks. *IEEE Trans. Geosci. Remote Sens.* **2024**, *62*, 5510514. [\[CrossRef\]](#)
47. Xu, Y.; Zuo, R. An interpretable graph attention network for mineral prospectivity mapping. *Math. Geosci.* **2024**, *56*, 169–190. [\[CrossRef\]](#)
48. Wang, C.; Wang, Y.; Ding, P.; Li, S.; Yu, X.; Yu, B. MI-fgat: Identification of multi-label protein subcellular localization by interpretable graph attention networks and feature-generative adversarial networks. *Comput. Biol. Med.* **2024**, *170*, 107944. [\[CrossRef\]](#) [\[PubMed\]](#)
49. Martyniuk, D.; Jung, J.; Paschke, A. Quantum architecture search: A survey. *arXiv* **2024**, arXiv:2406.06210.

50. Zhang, S.-X.; Hsieh, C.-Y.; Zhang, S.; Yao, H. Differentiable quantum architecture search. *Quantum Sci. Technol.* **2022**, *7*, 45023. [\[CrossRef\]](#)
51. Wu, W.; Yan, G.; Lu, X.; Pan, K.; Yan, J. QuantumDARTS: Differentiable quantum architecture search for variational quantum algorithms. In *Proceedings of the 40th International Conference on Machine Learning*; PMLR: Cambridge, MA, USA, 2023; Volume 202, pp. 37745–37764.
52. Zhang, S.-X.; Hsieh, C.-Y.; Zhang, S.; Yao, H. Neural predictor based quantum architecture search. *Mach. Learn. Sci. Technol.* **2021**, *2*, 45027. [\[CrossRef\]](#)
53. He, Z.; Deng, M.; Zheng, S.; Li, L.; Situ, H. GSQas follows: Graph self-supervised quantum architecture search. *Phys. A Stat. Mech. Its Appl.* **2023**, *630*, 129286. [\[CrossRef\]](#)
54. Lourens, M.; Sinayskiy, I.; Park, D.K.; Blank, C.; Petruccione, F. Hierarchical quantum circuit representations for neural architecture search. *npj Quantum Inf.* **2023**, *9*, 79. [\[CrossRef\]](#)
55. Patel, Y.J.; Kundu, A.; Ostaszewski, M.; Bonet-Monroig, X.; Dunjko, V.; Danaci, O. Curriculum reinforcement learning for quantum architecture search under hardware errors. *arXiv* **2024**, arXiv:2402.03500.
56. Du, Y.; Huang, T.; You, S.; Hsieh, M.-H.; Tao, D. Quantum circuit architecture search for variational quantum algorithms. *npj Quantum Inf.* **2022**, *8*, 62. [\[CrossRef\]](#)
57. Camargo, J.E.; González, F.A. A multi-class kernel alignment method for image collection summarization. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications: 14th Iberoamerican Conference on Pattern Recognition, CIARP 2009, Guadalajara, Jalisco, Mexico, 15–18 November 2009. Proceedings 14*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 545–552.
58. Wang, H.; Liang, Z.; Gu, J.; Li, Z.; Ding, Y.; Jiang, W.; Shi, Y.; Pan, D.Z.; Chong, F.T.; Han, S. Torchquantum case study for robust quantum circuits. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design; ICCAD '22*; Association for Computing Machinery: New York, NY, USA, 2022.
59. Metcalf, M.; Andrés-Martínez, P.; Fitzpatrick, N. Realizing quantum kernel models at scale with matrix product state simulation. In *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*; IEEE: Piscataway, NJ, USA, 2024; pp. 1–20.
60. Buhmann, M.D. Radial basis functions. *Acta Numer.* **2000**, *9*, 1–38. [\[CrossRef\]](#)
61. Chapelle, O.; Haffner, P.; Vapnik, V.N. Support vector machines for histogram-based image classification. *IEEE Trans. Neural Netw.* **1999**, *10*, 1055–1064. [\[CrossRef\]](#) [\[PubMed\]](#)
62. Cantori, S.; Vitali, D.; Pilati, S. Supervised learning of random quantum circuits via scalable neural networks. *Quantum Sci. Technol.* **2023**, *8*, 25022. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.