Research on Several Problems of Quantum Machine Learning

CONG LEI

Ph.D

THE UNIVERSITY OF
SYDNEY

Supervisor: Prof. Tongliang Liu

A thesis submitted in fulfilment of
the requirements for the degree of
Doctor of Philosophy

School of Computer Science
Faculty of Engineering
The University of Sydney
Australia

31August 2025

Statement of Originality

This is to certify that to the best of my knowledge, the content of this thesis is my own work. This thesis has not been submitted for any degree or other purposes.

I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

No content produced by generative AI tools has been used in the preparation of this thesis.

Student Name: **Date:** 31August 2025

Authorship Attribution Statement

This thesis was conducted at the University of Sydney, under the supervision of Prof. Tongliang Liu, between 2021 and 2025. This thesis contains several chapters that are related to the following papers:

(1) **Cong Lei**, Yuxuan Du, Peng Mi, Jun Yu, Tongliang Liu. 'Neural Auto-designer for Enhanced Quantum Kernels'. In: *The Twelfth International Conference on Learning Representations* (2024). [99]

(2) **Cong Lei**, Yuxuan Du, Shichao Zhang, Tongliang Liu. 'Universal parameter initialization for Quantum Neural Networks'. Under review (2025).

In addition to the statements above, in cases where I am not the corresponding author of a published item, permission to include the published material has been granted by the corresponding author.

Student Name: **Date:** 31August 2025

As supervisor for the candidature upon which this thesis is based, I can confirm that the authorship attribution statements above are correct.

Supervisor Name: **Date:** 31August 2025

To my parents.

Acknowledgements

This research reported in this thesis was supported by the award of a China Scholarship Council (CSC) scholarship to the PhD Candidate.

I am very grateful to Prof. Tongliang Liu, my supervisor, who trained me from a junior student to a researcher with independent conducting scientific research capabilities. I feel very lucky to be able to work with Prof. Liu and receive his careful guidance. This experience is a precious treasure for me. Also, his hardworking and care for students have profoundly influenced me. He often told us that as a researcher, we should read papers for at least 2 hours a day and keep an eye on the latest scientific research progress. The cultivation of this habit plays a vital role in our way to excellent researchers. In addition, he often helps students to vent negative emotions, which helps students maintain their enthusiasm for scientific research and allows us to go further and further on the road of research. Prof. Liu not only pays attention to our research, but also pays more attention to the atmosphere within the group and is committed to creating a good living and learning environment. For example, we have a fixed group meeting every week so that everyone can understand the research projects that other students are doing. In addition, Prof. Liu often organizes some activities, such as going to the beach for barbecue, which makes us, the students who have left our hometowns, feel a little warm and relieve our homesickness. Therefore, Prof. Liu is both a teacher and a friend to me (a supervisor in research and a friend in life). I would like to once again express my sincere respect for Prof. Liu.

I would also like to express my sincere gratitude to my collaborator, Dr. Yuxuan Du. His professional knowledge, enthusiastic discussions, and sincere opinions have been invaluable help to me throughout my academic career. His pursuit of academics has also had a profound impact on me. I remember one time, he was infected with COVID-19, but he still discussed with me with his feverish body, which shocked me greatly. In addition, his help in the writing

the paper and cultivating novel ideas are indispensable. I sincerely thank him for the selfless help he provided me.

I also want to express my heartfelt gratitude to my colleagues and friends, Dr. Yang Qian, Dr. Xinbiao Wang, Dr. Chaojian Yu, Dr. Jun Yu, Dr. Shichao Zhang, Wenzhen Zhang, Zhirong Huang, Jinghao Liu, Peng Mi, Zhuo Huang, Shouxing Ma, Ruiyu Liu, Shenshen Du, Xiaoyi Hu, Hui Kang, Xiangyu Sun and all those not mentioned but helped me a lot. Throughout my PhD journey, the selfless assistance and encouragement I have received from them has been an invaluable asset, benefiting both my research endeavors and daily life in immeasurable ways.

Abstract

Quantum machine learning methods, such as quantum kernels and variational quantum algorithms (VQAs), are poised to demonstrate quantum advantages in fields like machine learning, chemistry, condensed matter physics, and materials science. However, these methods face challenges that hinder the practicality of leveraging quantum advantages. For instance, quantum kernels require meticulous design to manifest quantum superiority, while VQA training poses a significant obstacle due to the presence of barren plateaus (BPs) in the optimization landscape. To address these issues, this thesis proposes two novel frameworks to guide the design of quantum kernels and to tackle the training challenges of quantum neural networks (QNNs) in VQAs, respectively.

Specifically, to tackle the challenge of designing quantum kernels, we propose a data-driven method for automatic design of quantum kernels, which leveraged the machine learning paradigm to learn from data to help us identify the appropriate quantum kernel. We show this method could automatically find the suitable quantum kernels without the vanishing similarity. Regarding the trainability of VQAs, we propose a parameter prediction framework based on encoder and decoder, where we employ the graph neural network (GNN) as the encoder to transmits messages based on directed acyclic graph (DAG) the of QNN. We demonstrate that the parameters generated by this method has enhanced the robustness, convergence, and trainability of the VQAs.

In conclusion, we propose two new schemes, QuKerNet and QuGHN, for the automatic design of quantum kernels and enhancing the trainability of VQAs. QuKerNet simultaneously considering both circuit layouts and variational parameters. Moreover, it is more resource-efficient for contemporary quantum hardware by considering qubit topology and the available qubit count. While QuGHN can be applied to quantum circuits with varying numbers of qubits, diverse ansatzes, and different tasks. In addition, the parameters QuGHN generates exhibit faster convergence, and also enable the VAQs to find better solutions. All these advancements

will not only enhance the practicality of quantum machine learning on near-term quantum machines, but also bolster its utility in large-scale simulations of quantum circuits.

Contents

List of Figures

List of Tables

List of Abbreviations

CHAPTER 1

Introduction

1.1 Background and Motivation

With the development of quantum computing [135], Quantum Machine Learning (QML) has attracted widespread attention as an emerging field. QML combines the advantages of quantum computing with machine learning methods, exploiting quantum properties to process data and perform calculations, offering new possibilities for solving complex problems [30, 150, 16]. Leveraging properties such as superposition [35] and entanglement [79], quantum machine learning is expected to demonstrate unique advantages in processing large-scale integer factorization [153] and combinatorial optimization problems [134], potentially bringing unprecedented breakthroughs to future intelligent systems and learning algorithms.

Among QML algorithms, quantum kernels [168, 162, 84] and Quantum Neural Networks (QNNs) [161, 1] stand out as leading proposals that can effectively implement on Noisy Intermediate-Scale Quantum (NISQ) machines [136]. And they all show great potential in solving complex tasks. Such as projected quantum kernel [80] demonstrates its quantum advantage in classification, while Variational Quantum Algorithms (VQAs) [28] designed with QNN can provide potential quantum advantage than classical computers in chemistry [179, 105], material sciences [67, 41], and particle physics [54, 121, 42] fields.

Despite the merits of quantum kernel and QNN, they still have the following issues that need to be solved:

(1) **Lack of prior knowledge.** On the one hand, quantum kernel design is quite different from the field of deep learning, where there are many well-perform in structures

that can serve as basic modules to construct neural networks, such as Convolutional Neural Network (CNN) [107], Recurrent Neural Network (RNN) [119], and Transformer [165]. In contrast, quantum machine learning is still in its early stages, and we are uncertain about which well-structured modules can be used to build quantum kernels. This situation is even more challenging in the near-term quantum devices, as it requires considering noise, topology, and various limitations. Designing a good quantum kernel becomes more challenging.

(2) **Exponentially-large search space.** The search space for potential quantum kernel architectures grows exponentially with system size. This makes it difficult for us to find a suitable kernel architecture within an acceptable time.

(3) **Poor trainability.** The exponentially-large Hilbert space of quantum mechanics is a double-edged sword. As this high-dimensional space can bring both quantum advantages and obstacles. This is because high-dimensional space can lead to the notorious Barren Plateaus (BPs) [118] phenomenon in QNN training. In this situation, the loss landscape will become almost flatten, making it impossible to optimize the loss function. Because this means we need exponentially more resources to optimize the loss function, which greatly weakens the practicality of VQA.

Motivated by the challenges outlined above, this thesis aims to contribute to the field of quantum kernel design and parameter initialization for QNN. In quantum kernel design, our research will answer the following questions: what is a good kernel and how to design a good kernel. For QNN, our research will first introduce methods that can alleviate BP, and then propose our solutions to improve the trainability of QNN and help QNN find better solutions.

1.2 Contributions

The contributions of this thesis can be summarized as follows:

(1) **Development of a quantum kernel design framework to automatically generate quantum kernels.** This research introduces a novel data-driven quantum kernel design method, named QuKerNet, designed to automatically design problem-specific

quantum feature maps and improve the power of quantum kernels. In addition, QuKerNet does not require prior knowledge of the task and can be used in any scenario, which greatly expands the practicality of QuKerNet.

(2) **Development of a parameter generation paradigm with improved performance across various tasks for QNNs.** This contribution presents a universal parameter initialization scheme for QNNs, called QuGHN. QuGHN uses Graph Neural Network (GNN) [147] to pass messages through the Directed Acyclic Graph (DAG) [46] of quantum circuit to generate efficient parameters. And it employs a pair of dynamic encoder and decoder to produce robust parameters for different system size, distinct ansatz, and diverse tasks.

1.3 Thesis Outline

The remainder of this thesis is organized into four chapters, with the core content of each chapter summarized as follows:

Chapter 2: Preliminary and Literature Review.

This chapter introduces basic knowledge related to this thesis, containing quantum computing basics and quantum data encoding. We then explain the principles of quantum kernel learning, quantum neural networks, variational quantum algorithms, and variational quantum eigensolver.

Chapter 3: The Design of Quantum Kernels.

Quantum kernels hold great promise for offering computational advantages over classical learners, with the effectiveness of these kernels closely tied to the design of the quantum feature map. This chapter presents a data-driven approach, *Quantum Kernel design by neural Networks (QuKerNet)* that automates the design of problem-specific quantum feature maps. QuKerNet leverages feature-selection [103] techniques to handle high-dimensional data on near-term quantum machines with limited qubits, and incorporates a deep neural predictor to efficiently evaluate the performance of various candidate quantum kernels. Through

extensive numerical simulations on different datasets, QuKerNet demonstrates its superiority, the capability of eliminating the kernel concentration issue and identifying the feature map with prediction advantages.

Chapter 4: Learning the Parameters of Quantum Neural Networks with Graph Hyper-Networks.

Parameter initialization is a very practical technique, which is seldom studied in the quantum neural networks. This chapter proposes a universal parameter initialization approach, *Quantum Graph HyperNetwork (QuGHN)* to generate near-optimal parameters to initialize the quantum circuit. QuGHN employs a parameter encoder-decoder architecture to train a model to predict the parameters of the circuit. And the graph neural networks is used as the encoder to acquire both global and local information from the circuit layout, while the hypernetworks is employed as the decoder to output the parameters. The simulation experiments on multiple tasks and quantum systems of different scales demonstrate the superiority and robustness of QuGHN, especially for large-scale circuits.

Chapter 5: Conclusion and Outlook.

This chapter offers a comprehensive summary of the thesis, summarizing the primary contributions and implications of the research conducted, while also delineating potential avenues for future research.

CHAPTER 2

Preliminary and Literature Review

Within this chapter, we briefly present essential foundational concepts vital for this thesis. like quantum computing, quantum data encoding, quantum kernel learning, quantum neural networks, variational quantum algorithms and variational quantum eigensolver.

2.1 Quantum Computing

In the following, we elaborate basic concepts of quantum computing. For a more comprehensive exploration of quantum computing and quantum information, we suggest the readers to read [125]. Quantum mechanics works in the Hilbert space $\mathcal{H}$ with $\mathcal{H} \cong \mathbb{C}$, where $\mathbb{C}$ represents the complex Euclidean space. We use Dirac notation to denote quantum states. A *pure quantum state* is defined by a vector $|\cdot\rangle$ (named 'ket') with the unit length. The mathematical expression of a state is $|\boldsymbol{a}\rangle = \sum_{i=1}^{d} \boldsymbol{a}_i \boldsymbol{e}_i = \sum_{i=1}^{d} \boldsymbol{a}_i |i\rangle \in \mathbb{C}^d$ with $\sum_i |\boldsymbol{a}_i|^2 = 1$, where the computational basis $|i\rangle$ stands for the unit basis vector $\boldsymbol{e}_i \in \mathbb{C}^d$. The inner product of two quantum states $|\boldsymbol{a}\rangle$ and $|\boldsymbol{b}\rangle$ is denoted by $\langle \boldsymbol{a}|\boldsymbol{b}\rangle$ or $\langle \boldsymbol{a}, \boldsymbol{b}\rangle$, where $\langle \boldsymbol{a}|$ refers to the conjugate transpose of $|\boldsymbol{a}\rangle$. A state $|\boldsymbol{a}\rangle$ is in *superposition* if the number of nonzero entries in $\boldsymbol{a}$ is larger than one. Analogous to the 'ket' notation, the *density matrix* can be used to describe more general qubit states evolved in open quantum systems. Formally, the density matrix of an n-qubit pure state $|\psi\rangle$ is $\rho = |\psi\rangle \langle\psi| \in \mathbb{C}^{2^n \times 2^n}$, where $\langle\psi| = |\psi\rangle^\dagger$ refers to the complex conjugate transpose of $|\psi\rangle$. For an ensemble of pure states of a quantum system, $\{p_j, |\psi_j\rangle\}_{j=1}^m$ with $p_j > 0$, where $|\psi_j\rangle$ is one state of this quantum system and j is an index, $\sum_{j=1}^m p_j = 1$, and $|\psi_j\rangle \in \mathbb{C}^{2^n}$ for $j \in [m]$, its density matrix is $\rho = \sum_{j=1}^m p_j \rho_j$ with $\rho_j = |\psi_j\rangle \langle\psi_j|$ and $\mathrm{Tr}(\rho) = 1$.

The basic element in quantum computation is the quantum bit (*qubit*). A qubit is a two-dimensional quantum state that can be written as $|\boldsymbol{a}\rangle = \boldsymbol{a}_1 |0\rangle + \boldsymbol{a}_2 |1\rangle$. Let $|\boldsymbol{b}\rangle$ be another qubit. The quantum state represented by these two qubits is formulated by the tensor product, i.e., $|\boldsymbol{a}\rangle \otimes |\boldsymbol{b}\rangle$ as a 4-dimensional vector. Following conventions, $|\boldsymbol{a}\rangle \otimes |\boldsymbol{b}\rangle$ can also be written as $|\boldsymbol{a}, \boldsymbol{b}\rangle$ or $|\boldsymbol{a}\rangle |\boldsymbol{b}\rangle$. For clearness, we sometimes denote $|\boldsymbol{a}\rangle |\boldsymbol{b}\rangle$ as $|\boldsymbol{a}\rangle_A |\boldsymbol{b}\rangle_B$, which means that the qubits $|\boldsymbol{a}\rangle_A$ ($|\boldsymbol{b}\rangle_B$) is assigned in the quantum register A (B). There are two typical quantum operations. The first one is *quantum (logic) gates* operating on a small number qubits. Any quantum gate corresponds to a unitary transformation and can be stated in the circuit model, e.g., an N-qubit quantum gate $U \in SU(2^N)$ satisfies $UU^\dagger = U^\dagger U = \mathbb{I}_{2^N}$. The second one is the *quantum measurement*, aiming to extract quantum information such as the computation result into the classical form. Given a density operator ρ, the outcome m will be measured with the probability $p_m = \mathrm{Tr}(\mathbf{K}_m \rho \mathbf{K}_m^\dagger)$ and the post-measurement state will be $\mathbf{K}_m \rho \mathbf{K}_m^\dagger / p_m$ with $\sum_b \mathbf{K}_b^\dagger \mathbf{K}_b = \mathbb{I}$.

Any unitary can be decomposed into a set of basis gates. The basis gate set explored in this study is $\mathcal{G} = \{H, R_X(\alpha), R_Y(\beta), R_Z(\gamma), \mathrm{CNOT}\}$, containing Hadamard gate, three rotational single-qubit gates along X, Y and Z-axis, respectively, and one two-qubit Control-not gate. The mathematical expression of these five basis gates is $H \equiv \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$, $R_\sigma(\theta) = \exp(-i\theta\sigma/2)$ with $\sigma \in \{X, Y, Z\}$, $X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, $Y \equiv \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$, $Z \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$, and $\theta \in [0, 2\pi]$, and $\mathrm{CNOT} \equiv |0\rangle \langle 0| \otimes \mathbb{I}_2 + |1\rangle \langle 1| \otimes \mathrm{X}$.

2.2 Quantum Data Encoding

In quantum machine learning, we typically use quantum algorithms to process classical data to gain a potential quantum advantage. The first step of these algorithms is to convert classical data into quantum states, a process called quantum data encoding [149]. In this section, we summarize several typical encoding methods.

2.2.1 Basis Encoding

Basis encoding is used to encode discrete classical data into quantum states. It is primarily used to process binary data. It maps data into Hilbert space according to the following formul:

$$x \to |x_{n-1}\rangle \cdots |x_1\rangle |x_0\rangle \tag{2.1}$$

where $x = \sum_{k=0}^{n-1} 2^k x_k$. For instance, 7 is encoded into $7 = 111 \to |1\rangle|1\rangle|1\rangle$. So it encoding a scalar requires $O(n)$ qubits, while encoding a vector with d dimensions necessitates $O(nd)$ qubits. This is often not an efficient use of qubits on current quantum devices.

2.2.2 Angle Encoding

For real continuous data, we usually consider using angle encoding to load classical data, as the angle values are continuous. This type of embedding encodes a scalar $x \in \mathbb{R}$ into a quantum state with the mapping:

$$x \to R_k(x)|0\rangle = e^{-i\pi\sigma_k/2}|0\rangle \tag{2.2}$$

where $k \in \{x, y, z\}$ is the axis of rotation in the Bloch sphere [125]. We usually set $k = x$. Note that the scalar x will be normalized to $[0, 2\pi]$.

2.2.3 Amplitude Encoding

Amplitude encoding is a very general encoding method as it can be used to handle complex continuous data. And it is generally used to encode a vector into a quantum state. More precisely, it will encode a d-dimensional vector $\boldsymbol{x}$ in the following way:

$$\boldsymbol{x} \to |x\rangle \equiv \sum_{i=0}^{d-1} x_i |i\rangle \tag{2.3}$$

where we assume $d = 2^N$, N is the number of qubits. And if that is not the case, we will pad all zero vectors to the end of $\boldsymbol{x}$ to make sure $d = 2^N$ holds true. It is worth noting that before $\boldsymbol{x}$ is encoded, it is normalized to $||\boldsymbol{x}|| = 1$. This encoding method can use only $O(\log d)$ qubits to encode d-dimensional vectors, which is very efficient. Unfortunately, this encoding

cannot be accurately implemented on current quantum devices, as the quantum circuits for such encoding may need to be exponentially deep to achieve suitable accuracy [39]. This motivates us to look for more practical encoding methods.

2.2.4 Parameterized Quantum Circuits (PQC)-based Encoding

PQC-based encoding strategy uses parameterized quantum gates to encode classical data, which is very intuitive. In general, it need d parameterized quantum gates of quantum circuit to encode a d-dimensional vector $\boldsymbol{x}$. In most cases, we use the single-qubit rotation gate to load classical data. Therefore, the above angle encoding can be seen as a special case of PQC-based encoding. In theory, a specific PQC corresponds to a concrete encoding strategy. Therefore, it is essential to identify a suitable encoding method from numerous PQC encoding strategies to enhance the performance of quantum machine learning algorithms.

2.3 Quantum Kernel Learning

Kernel methods are widely used in machine learning [77], such as classification [131], dimensionality reduction [148]. These algorithms that apply kernel methods to the field of machine learning are all based on kernel learning. The general kernel learning model can be expressed as:

$$f(x) = \sum\nolimits_{i=1}^{n} w_i \mathrm{K}(x_i, x) \tag{2.4}$$

where $f(\cdot)$ is the predict function, x is the test sample, w_i refers to the weight of the training sample x_i, $\mathrm{K}(x_i, x)$ represents the kernel function. The aim of kernel learning is to find the best w to obtain the minimum prediction error.

The key idea behind kernel methods is to use kernel functions to implement nonlinear mappings in high-dimensional feature spaces, thereby enabling linear classification or regression of raw data even in nonlinear problems. On the other hand, kernel methods allow calculations to be performed directly in the original feature space without explicitly calculating the high-dimensional feature space, thus avoiding the curse of dimensionality [90] and increased

complexity, and improving the efficiency and accuracy of machine learning algorithms. The more common ones are radial basis function kernel and polynomial kernel [171]. Before the rise of deep learning, kernel methods was a very powerful learning machine that be used in various fields [26].

Quantum kernel learning is to replace traditional kernel functions with quantum kernel functions. With the help of quantum resources (superposition and entanglement), quantum kernel methods have more powerful capabilities than classical kernel methods. Therefore, quantum kernel methods have become a potential research direction.

Essentially, quantum data encoding is the process of nonlinear feature map of classical data [149]. In this situation, quantum kernel learning model has the following forms:

$$f(x) = \sum_{i=1}^{n} w_i \mathrm{K}(\phi(x)_i, \phi(x)) \tag{2.5}$$

where $\phi(\cdot)$ indicates the effect of quantum data encoding. Therefore, the quantum data encoding strategy determines the final performance of the quantum kernel learning model. This once again proves that designing a suitable quantum data encoding method is very important in the field of quantum machine learning.

2.4 Quantum Neural Networks

The execution process of any algorithm can be divided into three steps: input, computation, and output. When quantum algorithms are applied to classical data, the corresponding three stages are quantum data encoding, quantum evolution, and measurement. Quantum data encoding was introduced in Sec. 2.2, this section focuses on quantum neural networks that implement quantum evolution.

Generally speaking, a quantum neural network defines a unitary transformation characterized by a set of adjustable parameters. When applying it to quantum states, one can implement computational processes similar to those in classical neural networks. By adjusting the parameters in the QNN to change the output quantum state or quantum embedding, thereby

modifying the prediction. This process of continuously adjusting parameters to make the model converge is the same as that of a classic neural network.

Here, we give a specific example. It is commonly used on a quantum computer, named hardware-efficient quantum neural network [13]:

$$U(\boldsymbol{\theta}) = \prod_{j=1}^{B} \hat{U}_j U_j(\theta_j) \tag{2.6}$$

where B denotes the number of blocks of QNN, $U_j(\theta_j) = \exp(-i\theta_j P_j/2)$ refers to the unitary derived from a Hermitian operator P_j, and $\hat{U}_j$ is the non-parameterized quantum gate, like CNOT gate. Therefore, we can design QNN with different parameterized quantum gates P_j and distinct non-parameterized quantum gates. Since different QNN architectures will significantly affect the performance of quantum machine learning algorithms. Quantum architecture search has become a hot research direction [49].

2.5 Variational Quantum Algorithms

QNN is a purely quantum model that can only run on quantum devices. However, on near-term quantum devices, the power of quantum neural networks is limited, making it difficult to release their full potential. With the development of deep learning technology, a hybrid classical and quantum algorithm framework has emerged, namely the variational quantum algorithms (VQAs) [134]. The VQAs uses classical optimizers to train QNNs, thereby producing a synergistic effect. In this case, VQAs appear to the best hope for obtaining quantum advantage [28].

In general, VQAs are to solve the following problem:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} C(U(\boldsymbol{\theta})) \tag{2.7}$$

where $C(\cdot)$ is the cost function related to the problem, $U(\cdot)$ denotes the unitary derived from a QNN. And $\boldsymbol{\theta}$ corresponds to the trainable parameters in QNN. From Eq. (2.7), we know that the two most important components of the VQAs are: the cost function and the circuit

ansatz. The former uses the idea of deep learning to encode a problem into a function. And the latter corresponds to the quantum circuits with parameters that can be trained to optimize the cost function. More specifically, computing the cost function $C(\cdot)$ is performed on a quantum computer, while updating the parameters $\boldsymbol{\theta}$ is running on a classical computer. This hybrid quantum-classical framework can simultaneously leverage the advantages of quantum computers and classical computers to achieve potential quantum advantage. We note that there are currently a variety of circuit ansatzes [98, 68, 170], each with different expressive capabilities, that is, different abilities to find the optimal solution. Therefore, designing a better circuit ansatz has become a hot topic worthy of research.

2.6 Variational Quantum Eigensolver

In this section, we will introduce a specific VQA for quantum chemistry using near-term quantum computers. The name of this VQA is Variational Quantum Eigensolver (VQE). And its core task is to solve the ground state energy and its corresponding quantum state of the Hamiltonian $\hat{H}$ within a closed physical system. The main implementation method is to prepare a parameterized wave function ansatz $|U(\boldsymbol{\theta})\rangle$ on the quantum device and then combine it with optimization algorithms from classical machine learning (such as gradient descent) to continuously adjust and optimize the parameters $\boldsymbol{\theta}$ to minimize the expectation value $\langle\psi|U^{\dagger}(\boldsymbol{\theta})\hat{H}U(\boldsymbol{\theta})|\psi\rangle$. The basic principle of this approach is based on the Rayleigh-Ritz variational principle [65]:

$$E_0 = \min_{\boldsymbol{\theta}} \langle\psi|U^{\dagger}(\boldsymbol{\theta})\hat{H}U(\boldsymbol{\theta})|\psi\rangle \tag{2.8}$$

where E_0 denotes the ground state energy of this system, ψ refers to the initial quantum state. Generally speaking, the VQE has four main components:

- A problem specification, which is corresponding to the Hamiltonian $\hat{H}$.
- An ansatz, which will affect the $U(\cdot)$ of Eq. (2.8).
- A classical optimizer, which optimizes the parameters $\boldsymbol{\theta}$ in the QNN.
- Classical post-processing for computing the solution to the problem.

In addition to these four components, an optional fifth component could be quantum error mitigation (QEM) [23], which may enhance the accuracy of the results from each iteration.

Due to VQE's computational advantages, it has been widely used to solve combinatorial optimization and machine learning problems. However, the training of VQE also suffers from a barren plateau phenomenon, which seriously hinders us to achieve quantum advantage. To overcome the barren plateau and improve VQE's trainability, more efforts are needed to pave our way to achieve quantum advantage.

CHAPTER 3

The Design of Quantum Kernels

This chapter mainly focuses on how to design a problem-specific quantum feature map that can provide computational advantages over classic kernels. In the process of designing kernels, we will encounter many challenges, such as the exponential search space and the lack of prior knowledge to guide our kernel design. To address the above issues we will provide a feasible solution in this chapter. The main content of this chapter was published in [99], and the remainder is organized as follows: in Sec. 3.1, we introduce the relevant background and research motivation of quantum kernel. Afterwards, we elaborate the specific mathematical expression of quantum kernel learning in Sec. 3.2. In Sec. 3.3, we explain our proposed method QuKerNet in detail. Next, we conduct numerical experiments in Sec. 3.4 to verify the performance of QuKerNet. At last, some conclusions are given in Sec. 3.5.

3.1 Introduction

Quantum computing presents a compelling prospect for revolutionizing machine learning, harnessing the unique attributes of quantum mechanics such as superposition and entanglement [56, 16]. While many studies have showcased the ability of quantum computers to reach quadratic or even exponential runtime improvements for solving linear equations and matrix factorizations [70, 140, 112, 71, 50, 62, 48], their practical realization for real-world applications poses significant challenges primarily due to the demanding error-corrected qubit requirements [10, 66]. Presently, quantum hardware has entered the noisy intermediate-scale quantum (NISQ) era [136], characterized by limited qubit counts, shallow circuit depth, inherent system noise, and constrained topologies. Consequently, great efforts have been directed

towards exploring quantum machine learning (QML) algorithms that can effectively operate on NISQ machines [28, 15]. Among these efforts, quantum neural networks [122, 1, 47, 51, 161, 137] and quantum kernels [73, 149, 17] stand out as leading proposals. Celebrated by the flexibility of quantum kernels, proof-of-principle experiments have been conducted to exhibit their feasibility in solving tasks in various domains [174, 72, 129, 91].

The experimental advance spurs the exploration of computational advantages of quantum kernels when executed on NISQ machines. In pursuing this inquiry, a recent theoretical study pointed out that quantum advantages may arise if two conditions are met [92]: the reproducing kernel Hilbert space formed by quantum kernels encompasses functions that are computationally hard to evaluate classically, and the target concept resides within this class of functions. According to the construction rule of quantum kernels, this means that their power tightly hinges on the quantum feature map, typically composed of input-dependent gates to encode classical data into the quantum state feature space and trainable quantum gates with a specified layout. This statement has been verified when quantum kernels are applied to tackle well-structured tasks, encompassing classification of synthetic datasets [80], discrete logarithmic problems [111], quantum phase recognition [176], and specific decision problems [84].

Different from well-structured problems, identifying quantum feature maps that satisfy both two conditions concurrently poses a significant challenge for most realistic datasets. As a result, quantum kernels associated with inappropriate quantum feature maps tend to be inferior to that of classical kernels [80]. Even worse, when the quantum feature map involves deep circuit depth, a large number of qubits, and too many noisy gates, the corresponding quantum kernel may encounter the vanishing similarity [159] and the degraded generalization ability [168], precluding any potential computational advantages.

To address this issue, several initial approaches have been explored, including the design of problem-specific quantum feature maps [83], rescaling the bandwidth of input data [152, 25], and mitigating the negative impact of system noise and shot error [168, 151]. These approaches are complementary to one another and aim to enhance the performance of quantum kernels. Specifically, in the realm of designing problem-specific quantum feature maps, two

primary streams of research have emerged: tuning variational parameters within the quantum feature maps [83, 63, 61] and employing evolutionary or Bayesian algorithms to continually adjust the arrangement of quantum gates within the feature map [7, 132, 162]. However, both methods encounter severe caveats. The former overlooks the influence of quantum gate layout (see Table 3.1 for supporting evidence), while the latter requests high computational overhead solely by adjusting the layout. Besides, none of them addresses the vanishing similarity issue encountered when processing high-dimensional data and struggles to adapt to the limited topology of NISQ machines. Given these considerations, a crucial question arises: *How can we design a generic method to efficiently construct problem-specific quantum feature maps capable of (i) adapting to high-dimensional data into modern quantum machines without encountering the vanishing similarity issue; (ii) accommodating both layout and parameter optimization with satisfactory performance?*

TABLE 3.1. **Evidence for the effect of gate layout in performance of quantum kernels**. Test accuracies of 1000 quantum kernels on tailored MNIST dataset whose feature map consists of the different gate layouts.

max acc: 82.67%	min acc: 44.67 %	avg acc: 68.70 %	std: 8.23%

In this work, we utilize advanced deep-learning techniques to overcome the limitations aforementioned. Specifically, we frame the quantum kernel design as a *discrete-continuous joint optimization* problem and then propose a data-driven method dubbed quantum kernel design by neural networks (QuKerNet) to solve this optimization problem, which amounts to effectively and automatically designing problem-specific quantum feature maps. To our best knowledge, this is the first framework that enables the automatic design of kernels by simultaneously considering both circuit layouts and variational parameters. Besides, QuKerNet is more resource-friendly to modern quantum hardware as it takes into account the topology of qubits and the available number of qubits. Despite that neural architecture search and quantum architecture search also orient to the discrete-continuous joint optimization, adapting these proposals to design quantum kernels is not straightforward, because of the expensive computational costs of quantum kernels compared to (quantum) neural networks in collecting accurate predictions of training data. In this regard, we devise a surrogate loss to efficiently optimize QuKerNet.

Two core components of QuKerNet are the feature-selection technique and a neural predictor. The incorporation of the feature-selection technique enables QuKerNet to handle high-dimensional data and overcome the limitations posed by near-term quantum machines with limited qubits, thereby ensuring the practical utility and suppressing the vanishing similarity issue. Moreover, the exploitation of a neural predictor allows QuKerNet to distill knowledge from different quantum kernels on a given dataset, guaranteeing efficient and accurate performance prediction for the novel quantum feature map. Extensive numerical simulations are conducted to validate the efficacy of QuKerNet.

3.2 Problem Setup

In this section, we first introduce the principles of classical kernels, then introduce the mechanism of quantum kernels. After that, we elaborate the mathematical expression of quantum kernel learning, and finally give our solution.

3.2.1 Mechanism of Quantum Kernels

Kernel methods provide a powerful framework to perform nonlinear and nonparametric learning, attributed to their universal property and interpretability. Suppose that both the training and test examples are sampled from the same domain $\mathcal{X} \times \mathcal{Y}$. The training dataset is denoted by $\mathcal{D} = \{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=1}^{n} \subset \mathcal{X} \times \mathcal{Y}$, where $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$ and $y^{(i)} \in \mathbb{R}$ refer to the i-th example with the feature dimension d and the corresponding label, respectively. A general construction rule of kernel methods is embedding the given input $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$ into a high-dimensional feature space, i.e., $\phi(\cdot) : \mathbb{R}^d \to \mathbb{R}^q$ with $q \gg d$, which allows that different classes of data points can be readily separable. Considering that explicitly manipulating $\phi(\boldsymbol{x}^{(i)})$ becomes computationally expensive for large q, kernel methods construct a kernel matrix $\mathrm{K} \in \mathbb{R}^{n \times n}$ to effectively accomplish the learning tasks in the feature space. Specifically, the elements of K represent the inner product of feature maps with $\mathrm{K}_{ij} = \mathrm{K}_{ji} = \langle \phi(\boldsymbol{x}^{(i)}), \phi(\boldsymbol{x}^{(j)}) \rangle$ for $\forall i, j \in [n]$, where such an inner product can be evaluated by a positive definite function $\kappa(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$ in $O(d)$ runtime. The aim of kernel learning is using the employed kernel

$\mathrm{K} \in \mathbb{R}^{n \times n}$ to infer a hypothesis $h(\boldsymbol{x}^{(i)}) = \langle \boldsymbol{\omega}^*, \phi(\boldsymbol{x}^{(i)}) \rangle$ with a high test accuracy, where $\boldsymbol{\omega}^*$ are optimal parameters minimizing the loss function

$$\mathcal{L}(\boldsymbol{\omega}) = \lambda \langle \boldsymbol{\omega}, \boldsymbol{\omega} \rangle + \sum_{i=1}^{n} (\langle \boldsymbol{\omega}, \phi(\boldsymbol{x}^{(i)}) \rangle - y^{(i)})^2. \tag{3.1}$$

The performance of kernel methods heavily depends on the utilized embedding function $\phi(\cdot)$, or equivalently $\kappa(\cdot, \cdot)$. As such, various kernels such as the radial basis function kernel, Gaussian kernel, circular kernel, polynomial kernel, and *quantum kernel* have been proposed to tackle various tasks.

The mechanism of quantum kernels differs from the classical kernels in designing the feature map $\phi(\cdot)$. For an N-qubit quantum kernel, the prepared quantum state for the i-th example yields $|\varphi(\boldsymbol{x}^{(i)}, \boldsymbol{\theta})\rangle = U_E(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}) |0\rangle^{\otimes N}$, where $U_E(\cdot, \boldsymbol{\theta})$ is the specified encoding quantum circuit with trainable parameters $\boldsymbol{\theta} \in \Theta$ and Θ being the parameter space. Denote $\rho(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}) = |\varphi(\boldsymbol{x}^{(i)}, \boldsymbol{\theta})\rangle \langle \varphi(\boldsymbol{x}^{(i)}, \boldsymbol{\theta})|$. The aim of quantum kernel learning is seeking a quantum kernel $\mathrm{W}(\boldsymbol{x}, \boldsymbol{\theta}) \in \mathbb{R}^{n \times n}$ with $\boldsymbol{\theta} \in \Theta$, i.e.,

$$\mathrm{W}_{ij}(\boldsymbol{x}, \boldsymbol{\theta}) = \mathrm{Tr}[\rho(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}) \rho(\boldsymbol{x}^{(j)}, \boldsymbol{\theta})], \ \forall i, j \in [n], \tag{3.2}$$

to infer a hypothesis formulated in Eq. (3.1) subjecting to a higher test accuracies than other kernels. In this regard, the power of quantum kernels is dominated by the exploited $U_E(\cdot, \boldsymbol{\theta})$. Notably, the implementation of $U_E(\cdot, \boldsymbol{\theta})$ is flexible and diverse. For an N-qubit quantum computer with the basis gate set $\mathcal{G}$, the generic form of the encoding circuit is

$$U_E(\boldsymbol{x}, \boldsymbol{\theta}) = \prod_{j=1}^{L} U_j(\boldsymbol{\alpha}_j) V_j \in SU(2^N), \ \forall j \in [L], \tag{3.3}$$

where $U_j(\boldsymbol{\alpha}_j)$ refers to any parameterized gate in $\mathcal{G}$ (e.g., R_X, R_Y and R_Z gates) acting on an arbitrary qubit, $\boldsymbol{\alpha}_j$ is an element in the set $\{\boldsymbol{x}, \boldsymbol{\theta}\}$, and $V_j \in \{\mathrm{CNOT}, \mathbb{I}_2\}$ refers to the fixed gate in $\mathcal{G}$ acting on arbitrary two qubits, and L represents the total number of quantum gates.

3.2.2 Quantum Kernel Learning

Let us first revisit the optimization perspective of quantum feature map design to highlight the shortcomings of previous approaches before moving to elaborate our proposal. As stated in Sec. 3.2.1, the performance of quantum kernels heavily depends on the exploited quantum feature map $|\varphi(\boldsymbol{x}, \boldsymbol{\theta})\rangle$ in Eq. (3.2), which has enormous choices attributed to the diversity of Θ and the gate arrangement of $\{(U_j, V_j)\}$ in Eq. (3.3). Taking account into such diversity, the loss function of quantum kernel learning in Eq. (3.1) should be reformulated as

$$\underset{\boldsymbol{\omega}, S\in\mathcal{S}, \boldsymbol{\theta}\in\Theta}{\arg\min}\ \mathcal{L}(\boldsymbol{\omega}, S, \boldsymbol{\theta}) = \lambda \langle \boldsymbol{\omega}, \boldsymbol{\omega} \rangle + \sum_{i=1}^{n} (\langle \boldsymbol{\omega}, \varphi_S(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}) \rangle - y^{(i)})^2, \tag{3.4}$$

where $|\varphi_S(\boldsymbol{x}^{(i)}, \boldsymbol{\theta})\rangle = U_E(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}; S) |0\rangle^{\otimes N}$ and $U_E(\cdot, \cdot; S)$ amounts that the gate arrangement $S \in \mathcal{S}$ is used to build the encoding circuit. Note that locating the global optima $(S^*, \boldsymbol{\theta}^*)$ to minimize $\mathcal{L}(\boldsymbol{\omega}, S, \boldsymbol{\theta})$ is difficult, caused by the two aspects: (i) the loss landscape with respect to $\boldsymbol{\theta}$ is non-convex; (2) the number of feasible gate arrangements $|\mathcal{S}|$ exponentially scales with N and $|\mathcal{G}|$.

Towards the optimization difficulty, initial attempts have been made to seek sub-optimal solutions of Eq. (3.4) in an efficient manner. Concretely, [7] discarded the parameter space $\boldsymbol{\theta}$ and only focuses on searching the optimal gate arrangement $S \in \mathcal{S}$ to form U_E; [113, 83] adopted the fixed circuit layout S and focus on optimizing $\boldsymbol{\theta}$. Unlike prior studies, our proposal *jointly optimizes the gate layout and parameters to approach the global minima of $\mathcal{L}$ in Eq. (3.4)*. In this point of view, all the attempts aforementioned are special cases of our method.

Another aspect overlooked in prior literature is the design of effective quantum kernels for high-dimensional datasets, which presents even greater challenges compared to the low-dimensional case. That is, an improper selection of gate arrangements can lead to the vanishing similarity issue, also known as kernel concentration [80, 159] in which the off-diagonal elements of matrix W progressively diminish as the number of qubits N and the total number of quantum gates L increase. Consequently, quantum kernels exhibit lower test accuracy compared to other kernels. Additionally, modern quantum machines with limited

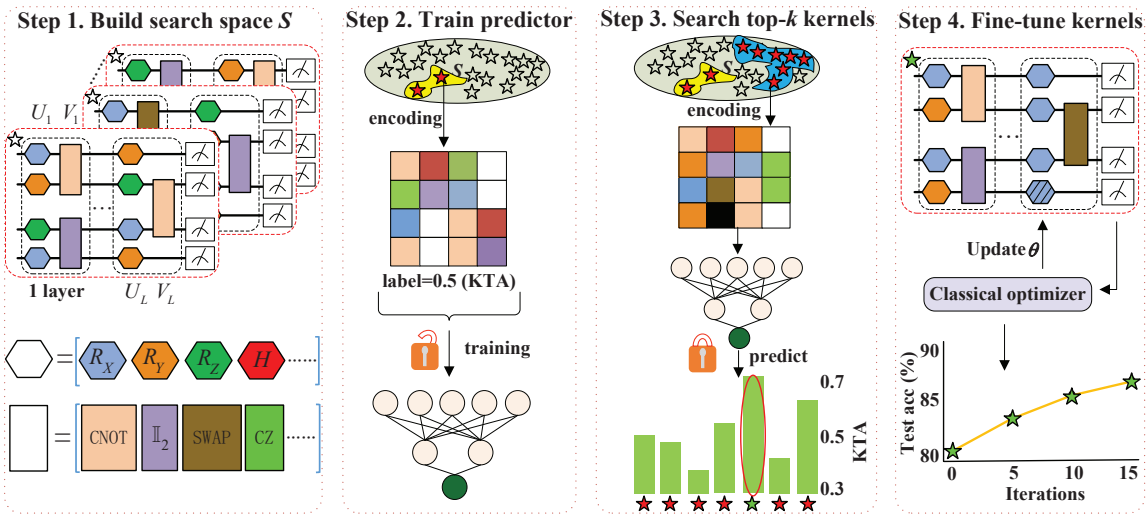


FIGURE 3.1. **The workflow of QuKerNet.** In step 1, QuKerNet sets up the search space $\mathcal{S}$ via the accessible basic quantum gate set $\mathcal{G}$. For example, the set of single-qubit gates includes R_X, R_Y, R_Z represented by colored hexagons, and the set of two-qubit gates contains $\mathrm{CNOT}, \mathrm{CZ}, \mathrm{SWAP}$ represented by the colored rectangles. In step 2, a pentagram represents a feasible candidate circuit in the search space $\mathcal{S}$. In the training process, M candidate circuits (highlighted by the red pentagrams) are collected and transformed to an image via the proposed encoding method. Meanwhile, Kernel-Target Alignment (KTA) of these sampled candidate circuits is calculated according to Eq. (3.5). These training data are used to train an MLP-based neural predictor. In step 3, the optimized neural predictor is employed to predict the performance of a large number of candidate circuits sampled from the search space $\mathcal{S}$. Afterward, QuKerNet ranks the predicted KTA and selects the top k candidate circuits. In step 4, for each candidate circuit, QuKerNet randomly replaces m parameterized gates, which are used to encode data features, by the variational parameters $\boldsymbol{\theta}$, highlighted by the hexagon with shadow. These parameters are optimized to maximize KTA. Then we compare their classification accuracy and the circuit with the highest accuracy is selected as the output circuit.

and noisy quantum resources make it challenging to directly embed high-dimensional data features into the encoding circuit. This is because when an extremely large value of L is used to encode the data, executing such quantum kernels on NISQ machines introduces a significant amount of errors, resulting in degraded performance [168].

3.3 Implementation of QuKerNet

Here we devise a data-driven approach to automatically design the near-optimal quantum kernel for the specified learning task by minimizing $\mathcal{L}$ in Eq. (3.4). Our approach, dubbed quantum kernel design by neural networks (QuKerNet), consists of four steps: search space

setup, neural-predictor training, top-k quantum kernels search, and fine-tune. An intuition is depicted in Fig. 3.1. Conceptually, QuKerNet completes the optimization of Eq. (3.4) via a two-stage learning strategy. In the first stage, QuKerNet estimates the optimal circuit layout $S^* \in \mathcal{S}$, accomplished by the first three steps. In the second stage, QuKerNet estimates the optimal $\boldsymbol{\theta} \in \Theta$ under the searched circuit layout, as completed by the last step. Through decoupling the discrete optimization from the continuous optimization, QuKerNet enables an efficient and scalable way to automatically design enhanced quantum kernels. In the remainder of this chapter, we first introduce the mathematical form implementation of each step, followed by the implementation details of these steps.

Search space setup. This step aims to design a suitable search space $\mathcal{S}$ in Eq. (3.4), determined by N and L. Concretely, for a large N and L, most of the candidate circuit layouts in $\mathcal{S}$ yield a poor performance, due to the vanishing similarity issue. To this end, QuKerNet applies the feature selection (FS) method, i.e., Max-Relevance and Min-Redundancy (mRMR) [133], to the dataset in the preprocessing step, where a dimension-reduction function $g : \mathbb{R}^d \to \mathbb{R}^p$ operates with each example $\boldsymbol{x}^{(i)}$ to extract p entries from d entries and form the new example $\hat{\boldsymbol{x}}^{(i)}$ with $p \ll d$. This approach not only mitigates the vanishing similarity issue but also facilitates the manipulation of high-dimensional data on NISQ machines. Once the new dataset $\hat{\mathcal{D}} = \{\hat{\boldsymbol{x}}^{(i)}, y^{(i)}\}_{i=1}^n$ is prepared, QuKerNet establishes $\mathcal{S}$ with the constraint $NL \sim O(p)$.

Neural-predictor training. This step targets to train a neural predictor to capture the relation between $S \in \mathcal{S}$ and the performance of the corresponding quantum kernel. Specifically, the neural predictor takes a set of circuit layouts in $\mathcal{S}$ as input and predicts their training accuracy on the dataset $\hat{\mathcal{D}}$. The optimization of neural predictor follows the supervised learning paradigm, where the weights of the neural network are optimized to minimize the discrepancy between its predictions and the true training accuracies via gradient descent methods. Through this training process, the neural predictor is expected to provide accurate predictions for the performance of unseen circuit layouts in $\mathcal{S}$.

Nevertheless, unlike QAS, the evaluation of the training accuracy of a quantum kernel with a specified circuit is computationally expensive. Therefore, an alternative strategy is needed to efficiently collect a labeled dataset $\mathcal{T}$ for training neural predictors. In our approach, we

replace the training accuracy with the Kernel-Target Alignment (KTA) as the label that the neural predictor predicts [38] since KTA is a reliable surrogate for classification accuracy and can be effectively calculated. Given a circuit layout S and the preprocessed dataset $\hat{\mathcal{D}}$, its KTA yields

$$\mathsf{K}(\hat{\mathcal{D}}; S) = \frac{1}{\mathcal{N}} \sum\nolimits_{ij} y^{(i)} y^{(j)} \, \mathrm{W}_{i,j}(\hat{\boldsymbol{x}}, \boldsymbol{\theta}; S), \tag{3.5}$$

where $\mathrm{W}_{i,j}(\hat{\boldsymbol{x}}, \boldsymbol{\theta}; S)$ refers to the (i, j)-th entry of the quantum kernel W in Eq. (3.2) whose circuit layout is S, $y^{(i)} \in \{0, 1\}$ is the binary label of the i-th example, and $\mathcal{N} = n \| \, \mathrm{W} \, \|_F$ denotes the normalization factor. For multi-class datasets with R classes, its KTA is computed by replacing $y^{(i)} y^{(j)}$ with $J_{i,j}$, i.e, $J_{i,j} = 1$ if $y^{(i)} = y^{(j)}$; otherwise, $J_{i,j} = -1/(R-1)$ [24]. Refer to Section 3.4.3 for the comparison of the runtime cost between KTA and the direct optimization. To facilitate calculation, we set $\Theta = \emptyset$ so that all parameterized quantum gates in S encode the data feature without the trainable parameters $\boldsymbol{\theta}$. By calculating KTA of $M \sim O(poly(p))$ different circuit layouts from $\mathcal{S}$ in parallel, we obtain the labeled dataset $\mathcal{T} = \{S^{(i)}, \mathsf{K}(\hat{\mathcal{D}}; S^{(i)})\}_{i=1}^{M}$.

The neural predictor utilized in QuKerNet comprises three essential components: input vectorization, model implementation, and optimization strategy. The input vectorization step addresses the conversion of the unstructured format of the circuit layout S into a structured representation that can be processed by deep neural networks. In this regard, we employ the transformation strategy proposed by [185] to accomplish this task. For the implementation of the neural predictor, we adopt the Multilayer Perceptron as the underlying architecture [64]. The specific configuration of the neural network, including the number of hidden layers, is determined based on the characteristics of the datasets being considered.

Top-k quantum kernels search. The purpose of this step is to select the most promising quantum kernels in the search space. Firstly, we randomly sample M' circuit layouts $\mathcal{S}'$ from $\mathcal{S}$ with $\mathcal{S}' \subset \mathcal{S}$ and $|\mathcal{S}'| = M'$. Then, the trained neural predictor is used to predict the $\mathsf{K}(\hat{\mathcal{D}}; S)$ with $S \in \mathcal{S}'$. Afterward, these M' circuits are sorted according to $\mathsf{K}(\hat{\mathcal{D}}; S)$ and the top k circuit layouts are preserved to construct the candidate layout set $\mathcal{S}_c$.

Fine-tune. This step aims to improve the performance of quantum kernels in $\mathcal{S}_c$. To do so, given $S \in \mathcal{S}_c$, we randomly reset m encoding gates in $U_E(\boldsymbol{x})$ as tunable parameters, i.e., $U_E(\boldsymbol{x}; S)$ turns to be $U_E(\boldsymbol{x}, \boldsymbol{\theta}; S)$ and $\Theta \neq \emptyset$. Besides, to ensure the performance of quantum kernels, the parameters $\boldsymbol{\theta}$ are initialized using the average value of corresponding features in the relevant gates. Then we fine-tune the quantum kernels to maximize the KTA with the fixed S by updating $\boldsymbol{\theta}$. After training, we choose the circuit layout with the highest classification accuracy from K fine-tuned candidates as the searched quantum kernel.

Then the quantum kernel W^* can be obtained based on the $(S^*, \boldsymbol{\theta}^*)$. Consequently, the optimal $\boldsymbol{\omega}^*$ can be achieved through $\boldsymbol{\omega}^* = \sum_{i=1}^{n} \sum_{j=1}^{n} \phi(\boldsymbol{x}^{(i)})((\mathrm{W}^* + \lambda I)^{-1})_{ij} y^{(j)}$, which can be used for kernel learning.

3.3.1 Implementation Details of QuKerNet

In this section, we elaborate on the missing details of QuKerNet. In particular, we separately present the implementation details of each stage of QuKerNet, followed by the summarization of its Pseudocode.

Search space setup. We visualize the construction rule of search space in Fig. 3.2. Specifically, given the dataset $\hat{\mathcal{D}} = \{\hat{\boldsymbol{x}}^{(i)}, y^{(i)}\}_{i=1}^{n}$, there are several strategies to load the classical data into an N-qubit quantum system. Suppose that the feature dimension of $\hat{\boldsymbol{x}}^{(i)}$ is p. In the case of $N = p$, the first way is element-wise encoding, where the variational quantum gates operating with j-th qubit load the j-th feature of $\hat{\boldsymbol{x}}^{(i)}$, i.e., $\boldsymbol{\alpha}_j = \hat{\boldsymbol{x}}_j^{(i)}$ with $\boldsymbol{\alpha}_j$ is defined in Eq. (3.3). An alternative way is random encoding, where each variational quantum gate loads one non-repetitive feature of $\boldsymbol{x}^{(i)}$.

We next consider the scenario with $p < N$ and design two encoding approaches. First, for the variational quantum gates acting on the j-th qubit with $j \in [N]$, the encoded data feature is $\hat{\boldsymbol{x}}_{j'}^{(i)}$ with $j' = j \pmod{p}$. The second approach involves random parameters. For the variational quantum gate operating with the first p qubits, we set $\boldsymbol{\alpha}_j = \hat{\boldsymbol{x}}_j^{(i)}$ for $\forall j \in [p]$. Besides, for the rest $N - p$ qubits, the parameters of the variational quantum gates are randomly and uniformly sampled from the interval $[0, 2\pi)$.

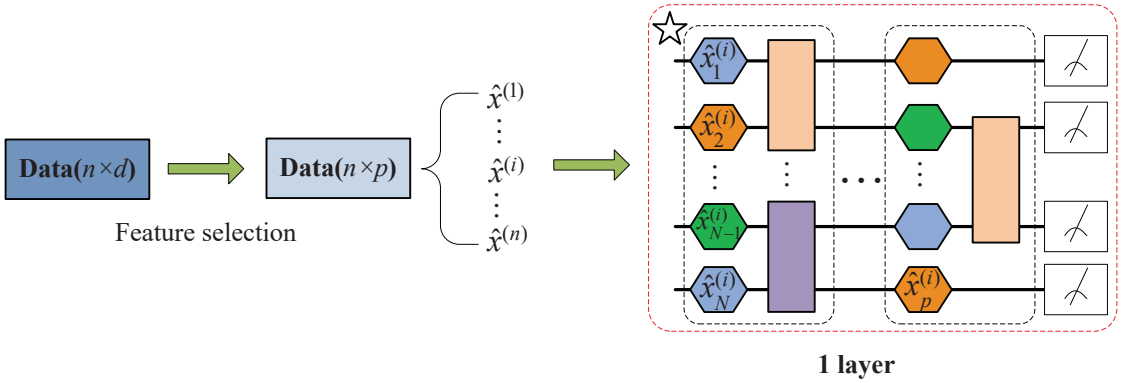
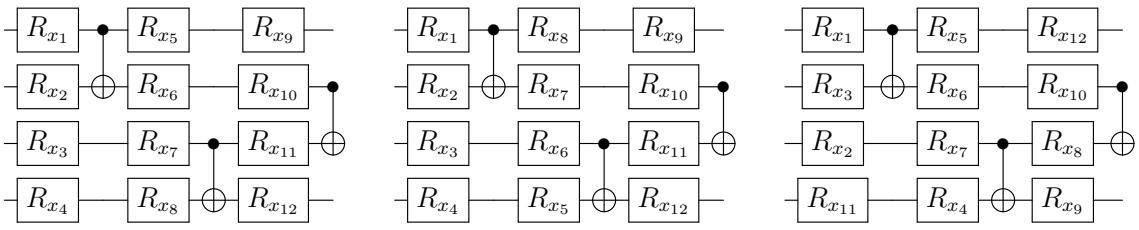


FIGURE 3.2. **The step 1 of QuKerNet**. First, we preprocess the data through feature selection. Next, we design the search space for the circuits based on relevant rules. Then, we encode the preprocessed data using different circuits sampled from the search space.

Last, in the case of $p > N$, we design three encoding methods as exhibited in Fig. 3.3. The first way is sequential encoding, which encodes features sequentially. The second way is chain encoding, which utilizes the chain queue to load features. The third way is random encoding. In this case, the $\boldsymbol{\alpha}_j$ is set to the value of the feature randomly selected from $\hat{\boldsymbol{x}}^{(i)}$.

We note that there are multiple ways of using random encoding to form a multi-layer encoding layout. The first solution is replicating the entire layout of a single block multiple times. Another solution is expanding each dotted area in Fig. 3.4 multiple times. It is also feasible to repeat the layout of the individual dotted areas in Fig. 3.4 for the varied times.



(A) Sequential encoding (B) Chain encoding (C) Random encoding

FIGURE 3.3. Various encoding methods. The x_j is corresponding to the j-th feature in sample $\hat{\boldsymbol{x}}^{(i)}$.

Neural-predictor training. Neural-predictor adopted in QuKerNet is composed of two parts: image encoder and the deep neural networks. The former intends to convert the information circuit layout into a format that can be processed by neural networks. The latter is employed to

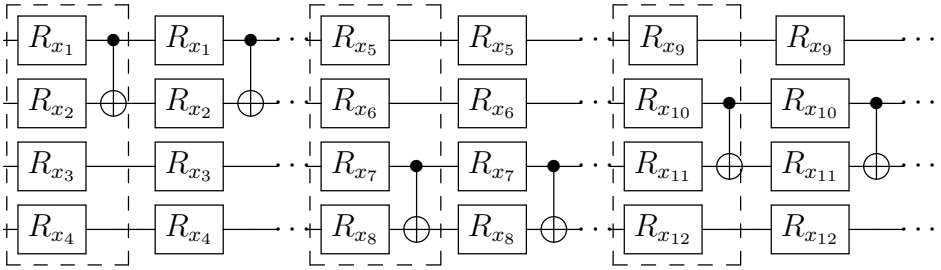


FIGURE 3.4. Repeated encoding scheme. The x_j is corresponding to the j-th feature in sample $\hat{\boldsymbol{x}}^{(i)}$.

distill the knowledge between circuit layout (i.e., quantum feature map) and the corresponding performance. In the following, we separately explain their implementations.

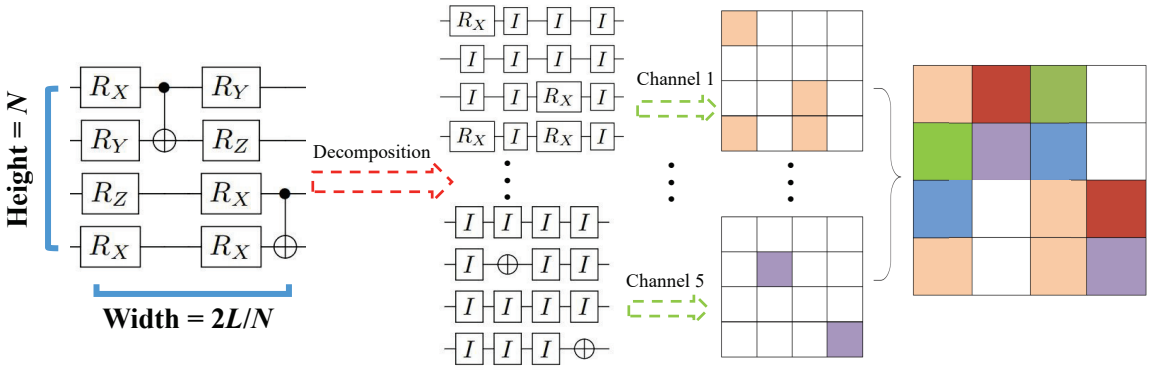


FIGURE 3.5. **Representation of quantum circuit layout**. The number of qubits $N = 4$, and the total number of rotation gates $L = 8$. First, the input circuit can be divided into five separate layouts, each layout containing only one type of gate. Then, these layouts are individually encoded into images, with each image corresponding to a channel. Finally, the images from the five channels are combined to form the final output image.

The implementation of the image encoder is shown in Fig. 3.5. In particular, we encode circuits into images based on the scheme proposed in [185]. For circuits with varied depth, we fill the encoded images with blank pixels to the maximum width of images in this set to unify the size of these images (refer to Fig. 3.6(a)). Specifically, N is seen as the height of the image, and $2L/N$ corresponds to the width of the image. In addition, each type of gate associates with an image channel. Note that CNOT gate corresponds to 2 channels, because it includes a target qubit and a control qubit. And there are only two pixel values in each channel, that is, 0 and 1, which indicates whether a gate has been applied or not (1 represents a gate has been applied, 0 means there is either no gate or $\mathbb{I}_2$ gate has been applied on this

position), and the position of the pixel is identical with the position of this gate in quantum circuit.

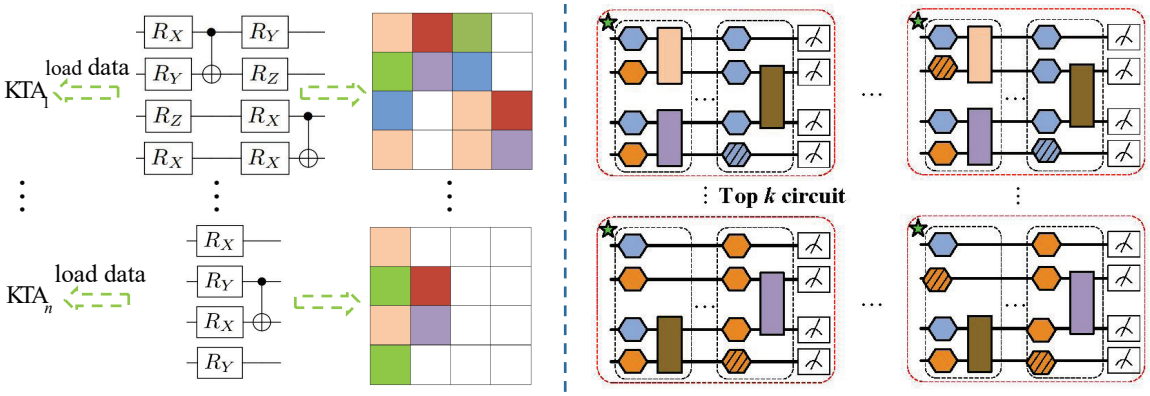


FIGURE 3.6. **Some details of step 2 and 4 in QuKerNet**. (a) The left area represents data collection for neural network training. First, all the circuits sampled from $\mathcal{S}$ will be encoded into images of a uniform size (with the width determined by the maximum depth among all circuits). Then, these circuits also are used to load the data and calculate the KTA using Eq. (3.5). Finally, the pairs of images and KTA can be used to train the neural network. (b) The right area is the fine-tuning stage. For each of the k circuits found through the QuKerNet-1, the QuKerNet performs $|\boldsymbol{\theta}|$ different parameterized gate replacements. In each replacement, the number and positions of replaced gates remain the same for the k circuits.

The implementation of the neural network adopted in QuKerNet is depicted in Fig. 3.7. That is, a simple Multilayer Perceptron (MLP) is employed as the regression model. This model is a 2-layer MLP model, including one hidden layer and one output layer. And the total number of trainable parameters of this model is $1280L + 257$, where L refers to the total number of quantum rotation gates. Adam optimizer is applied in our experiments and the learning rate is set to 0.01, and the batch size is 32. Besides, in order to achieve better prediction results, we magnify the value of KTA 10 times in each data pair.

Fine-tune. Recall that QuKerNet exploits two random methods in this stage to enlarge the parameter space to improve the learning performance. The first random operation refers to the number ($m \in \{1, 2 \ldots, L/L_0 - 1\}$) of reset parameterized gates in $U_E(\boldsymbol{x})$ that is random in each processing of fine-tune. And the second random operation is that the positions of reset gates are randomly selected. Combined with the above two random operations, QuKerNet can explore larger parameter space. The details of fine-tuning is shown in Fig. 3.6(b).

For clarification, we summarize the pseudocode of QuKerNet.

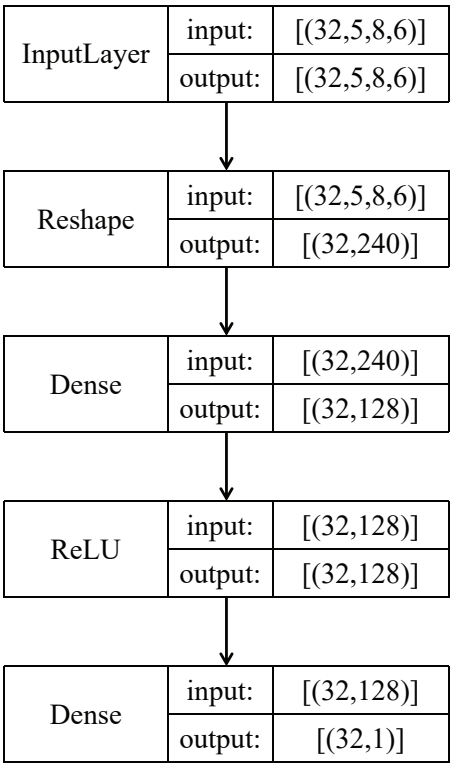


FIGURE 3.7. **MLP based model for regression**. [32,5,8,6] corresponding to the batch size, the number of gate types, the number of qubits N, and the depth of the circuit (refer to the width in Fig. 3.5), respectively.

Algorithm 1: Pseudo code of QuKerNet.

Input: $\boldsymbol{x} \in \mathbb{R}^{n\times d}$, p, M, M', L_0, N, k and $|\Theta|$;
Output: S^* and $\boldsymbol{\theta}^*$;

1 Conducting feature selection on $\boldsymbol{x}$ to get $\hat{\boldsymbol{x}} \in \mathbb{R}^{\boldsymbol{n}\times \boldsymbol{p}}$;
2 Sampling M circuits from $\mathcal{S}$ based on L_0, N, and encoding M circuits to images as well as calculating KTA by Eq. (3.5);
3 Train the predictor on training set of M pairs (image, KTA);
4 Sampling M' circuits from $\mathcal{S}$ based on L_0, N, and encoding them into images ;
5 Utilizing the predictor to predict the KTA of M' circuits ;
6 Selecting k circuits with the top predicted KTA from M' circuits. Fine-tuning $|\Theta|$ different $\boldsymbol{\theta}$ for each of the k circuits ;
7 To validate the test accuracy of $k|\Theta|$ kernels on $\hat{\boldsymbol{x}}$, and pick the kernel with the highest test accuracy.

3.4 Numerical Results

In this chapter, we conduct a set of experiments to evaluate the performance of QuKerNet across various datasets. Our primary objectives are to explore the potential merits of the quantum kernel designed by QuKerNet compared to prior quantum kernels and classical kernels, assess the effectiveness of QuKerNet in addressing the vanishing similarity issue for large-scale quantum kernels, and evaluate the robustness of QuKerNet in noisy environments. All simulations are conducted using Python, utilizing the PennyLane [14], PyTorch [130], and the JAX library [19]. All experiments are run on AMD EPYC 7302 16-Core Processor (3.0GHz) with 188G memory (Ubuntu system).

3.4.1 Data Sets

We benchmark QuKerNet on three datasets, each representing a distinct domain: computer vision, finance, and learning tasks relevant to the demonstration of quantum advantages. The first dataset utilized is a tailored version of the MNIST dataset [96], consisting of handwritten digit images. To ensure computational efficiency, we distill 300 images from the original MNIST dataset, focusing on the labels from 0 to 4, and reduce the feature dimension of each example from 784 to 40 using mRMR. The second dataset employed is a tailored Credit Card (CC) dataset [40], commonly used for fraud detection and risk assessment in the financial industry. For this dataset, we extracted 200 (100 for each class) samples from the CC dataset and reduce the dimension of each example from 28 to 24 using Principal Components Analysis (PCA). Last, we utilize the method proposed in [80] to relabel the tailored MNIST dataset (class 0 and 1), creating a synthetic dataset that allows us to evaluate the potential advantages of QuKerNet in a controlled manner.

3.4.2 Experimental Setting

In the section, we adopt three kernels, i.e., radial basis function kernel (RBFK), hardware efficient ansatz-based quantum kernel (HEAK), and training embedding quantum kernel (TEK), to benchmark the performance of our proposal. For completeness, here we provide the

necessary backgrounds of three kernels. We also describe the more specific implementation details of QuKerNet.

Radial basis function kernel. RBFK is a common kernel used in machine learning. The mathematical form of RBFK is

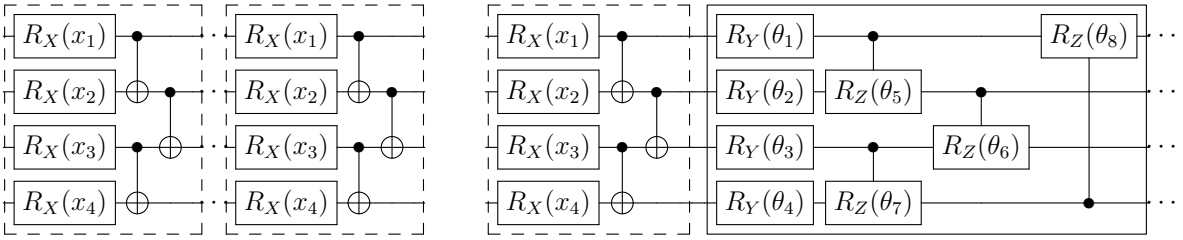
$$\mathrm{K}_{ij} = \exp(-\gamma||\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}||^2) \tag{3.6}$$

where γ is a hyper-parameter. RBFK is commonly used in various domains such as image classification [32], face detection [11], text classification [58], protein structure prediction [116], and many other areas [86].

Hardware efficient ansatz-based kernel.

$$U_E(\boldsymbol{x}, \boldsymbol{\theta}) = \prod_{j=1}^{B} \hat{U}_j(\boldsymbol{\beta}) \tag{3.7}$$

where $\hat{U}_j(\boldsymbol{\beta})$ refers to the j-th block taking the form as $\hat{U}_j = (\otimes_{i=1}^{N} R_a(\boldsymbol{\beta}_{ij}))\hat{V}$, $a \in \{X, Y, Z\}$, $\boldsymbol{\beta}_{ij}$ is an element in the set $\{\boldsymbol{x}, \boldsymbol{\theta}\}$, and $\hat{V}$ refers to the entangled layer consisting of CNOT gate and $\mathbb{I}_2$ gate. And HEAK is widely used in the fields of quantum machine learning [124], quantum chemistry [88], and combinatorial optimization [101] due to its implementability and expressibility on NISQ devices.



(A) HEAK layout (B) TEK layout

FIGURE 3.8. **HEAK layout and TEK layout**. The $\boldsymbol{x}_j$ is corresponding to the j-th feature in sample $\hat{\boldsymbol{x}}^{(i)}$. (a) The layout of HEAK. And each dashed region is a block that is used to encode up to N features of $\hat{\boldsymbol{x}}^{(i)}$. (b) The layout of TEK. The layout of the dashed region is used for encoding data, while the solid line portion represents variational parameters used to adjust the performance of the kernel. A trainable module is connected after each block of HEAK. The dashed area and the solid line area together form a block.

Training embedding kernels. TEK is a method that only optimizes the $\boldsymbol{\theta}$ for a given S. In this paper, TEK is designed based on HEAK, and their layouts are shown in Fig. 3.8. Therefore, it takes the following mathematical form:

$$U_E(\boldsymbol{x}, \boldsymbol{\theta}) = \prod_{j=1}^{B} \hat{U}_j(\boldsymbol{\beta}) \widetilde{U}_j(\boldsymbol{\gamma}) \tag{3.8}$$

where $\hat{U}_j(\boldsymbol{\beta})$ is the same to the definition in Eq. (3.7). While $\widetilde{U}(\cdot)$ is a flexible layout used to adjust the quantum states of encoding, and $\boldsymbol{\gamma}$ represents the variational parameters. In this study, we use the circuit layout proposed in [83] to construct $\widetilde{U}(\cdot)$. So $\widetilde{U}_j(\boldsymbol{\gamma}) = (\otimes_{i=1}^{N} R_Y(\boldsymbol{\gamma}_{ij}))\widetilde{V}$, $\boldsymbol{\gamma}_{ij}$ is an entry of $\boldsymbol{\theta}$, and $\widetilde{V}$ refers to the entangled layer. And for the each index of qubit i, we apply the CR_Z gate to the i-th and $(i+1) \pmod N$-th qubits. TEK is suitable for image classification [113] or regression tasks [110].

In summary, RBFK and HEAK are static kernels with deterministic feature mapping. While HEAK and TEK are quantum kernels and these kernels are built upon the same circuit layout S.

Specific implementation of QuKerNet. Throughout the entire experiment, we utilize the Hardware Efficient Ansatz (HEA) [88] as the backbone to construct the search space of QuKerNet. In particular, as shown in Fig. 3.9(a), HEA subsumes a block-wise layout, and each block is composed of parameterized single-qubit gates and fixed two-qubit gates. In this context, the encoding circuit $U_E(\boldsymbol{x}, \boldsymbol{\theta}) = \prod_{j=1}^{L} U_j(\boldsymbol{\alpha}_j) V_j = \prod_{j'=1}^{B} \hat{U}_{j'}(\boldsymbol{\beta})$, where $\hat{U}_{j'}(\boldsymbol{\beta})$ refers to the j'-th block taking the form as $\hat{U}_{j'} = (\otimes_{i=1}^{N} R_a(\boldsymbol{\beta}_{ij'}))\hat{V}$, $a \in \{X, Y, Z\}$, $\boldsymbol{\beta}_{ij'}$ is an element in the set $\{\boldsymbol{x}, \boldsymbol{\theta}\}$, and $\hat{V}$ refers to the entangled layer consisting of CNOT gate and $\mathbb{I}_2$ gate. The construction rule of the search space is as follows. For the parameterized single-qubit gates in each block, when the index of qubits i satisfies $i \pmod 2 = 0$, we apply the same parameterized gate R_a (each gate in R_a appears at a probability of 1/3); otherwise, we adopt another type of parameterized gate $R_{a'}$. For the entangled gates in each block, when the index of qubit i satisfies $i < N$, we pick a two-qubit gate from $\{\mathrm{CNOT}, \mathbb{I}_2\}$ uniformly random and apply it to the i-th and $(i+1)$-th qubits. This strategy not only facilitates the balance of the size of the search space and the expressivity but also enables an efficient

implementation of the circuit on real quantum devices. With a slight abuse of notations, in the remained section, we denote L_0 as the layer number, contrasting with the original definition of gate number in Eq. (3.3). For instance, when the U_E encodes all the features of $\boldsymbol{x}$, the layer number of U_E is seen as 1 and denoted by $L_0 = 1$ and the relationship between L_0 and the number of blocks B, is $B = L_0 * \lceil p/N \rceil$.

3.4.2.1 Hyper-parameter Settings

In this section, we detail the hyper-parameter settings of classical and quantum kernels in comparison.

RBFK. If there is no special explanation, we set $\gamma \in \{1, 2, 3, 4, 5\}/(p \text{ Var}[\boldsymbol{x}_j^{(i)}])$ and $\text{Var}[\boldsymbol{x}_j^{(i)}]$ is the variance of all the features $j = 1, \ldots, d$ from all the data points $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}$.

HEAK. In our experiments, $a \equiv X$. For the entangled gates in each block, when the index of qubit i satisfies $i < N$, we apply the CNOT gate to the i-th and $(i + 1)$-th qubits. While the $\boldsymbol{\beta}_{ij}$ corresponds to one feature in $\hat{\boldsymbol{x}}^{(i)}$.

TEK. We randomly initialize each $\boldsymbol{\gamma}_{ij}$ with a value uniformly sampled from the interval $[0, 2\pi)$, and set the epoch to 30 for optimization. Meanwhile, we employ gradient descent optimization with a learning rate of 0.2 to minimize the negative KTA of TEK.

QuKerNet. Here we only introduce the general hyper-parameter settings and defer the specific hyper-parameter settings to the corresponding experiments. Except for the KTA experiment, $N = 8, L_0 \in \{1, 2, 3, 4, 5\}, M' = 50000, k = 10, |\Theta| = 20$. And the neural predictor is optimized by Adam with 0.01 learning rate for 30 epochs, and the criterion used is Smooth L1 Loss. Each setting is repeated with 5 times to collect the statistical results.

Both TEK and HEAK employ a sequential encoding strategy (refer to Fig. 3.3(a)), with $L_0 = 1$ to encode data.

3.4.2.2 Performance metrics

In this section, we will first introduce three metrics for evaluating the performance of different algorithms and then provide some basic knowledge about the phenomenon of exponentially concentrated value in quantum kernels to help us better understand this phenomenon.

Specifically, we use test accuracy (Acc) to evaluate the generalization ability of QuKerNet, and employ the Pearson correlation coefficient (PCC) to measure the correlation between the performance of a predictor trained using KTA and the performance of a predictor trained using training accuracy, and utilise the kernel variance (KV) to inspect the degree of vanishing similarity of the specified kernel.

- **Acc.** We define classification accuracy as follows:

$$\mathrm{Acc} = n_{correct}/n \tag{3.9}$$

  where *n* is the number of samples in test data and $n_{correct}$ means correctly classified samples.
- **PCC.** We use the PCC (higher is better) to evaluate the validity of KTA. The Pearson correlation coefficient (PCC) is defined as:

$$\mathrm{PCC} = \frac{\mathrm{cov}(\boldsymbol{X}, \boldsymbol{Y})}{\sigma_{\boldsymbol{X}}\sigma_{\boldsymbol{Y}}} \tag{3.10}$$

  where $\boldsymbol{X}$ and $\boldsymbol{Y}$ are a given pair of random variables (for example, test accuracy and KTA), the cov is the covariance, $\sigma_{\boldsymbol{X}}$ and $\sigma_{\boldsymbol{Y}}$ are the standard deviation of $\boldsymbol{X}$ and $\boldsymbol{Y}$, respectively. The value of PCC ranges from -1 to 1, which implies the degree of correlation between $\boldsymbol{X}$ and $\boldsymbol{Y}$. The larger the absolute value of PCC, the stronger the correlation (-1 means they are perfectly negatively correlated, while 1 indicates they are perfectly positively correlated).
- **KV.** The mathematical expression of KV for the kernel W is $\mathrm{Var}(\mathrm{W}) = \mathbb{E}(\mathrm{W}^2) - (\mathbb{E}(\mathrm{W}))^2$, where the expectation is taken over the input data $\boldsymbol{x}$.

Vanishing similarity of quantum kernels. The vanishing similarity of quantum kernels is that as the size of the problem increases, the difference between kernel values becomes

increasingly small, which means that kernel values will tend towards the same value. More precisely, vanishing similarity can be formally defined as follows:

$$\mathrm{P}[|\,\mathrm{W}_{ij} - \mathbb{E}(\mathrm{W}_{ij})| \geq \delta] \leq \frac{\beta^2}{\delta^2}, \beta \in O(1/b^N) \tag{3.11}$$

where $b > 1$. If $\mathrm{Var}[\mathrm{W}_{ij}] \in O(1/b^n)$ and for all W_{ij}, Eq. (3.11) holds true, we identify that this is a phenomenon of vanishing similarity. In general, there are four main triggers that lead to vanishing similarity, including high expressibility, global measurements, entanglement and noise [159]. When this phenomenon occurs, it becomes challenging to extract meaningful information from the quantum states in a reasonable amount of time. This is because distinguishing subtle differences between quantum states to evaluate quantum kernels requires an exponential number of measurements. Therefore, vanishing similarity leads to the poor performance of quantum kernels. To address this issue, one should be avoid to design a highly expressible, entangled quantum kernel. For the global measurements-induced vanishing similarity, one can employ problem-specific quantum kernel to remit it [111].

3.4.3 Experimental Results

In the section, we present our numerical simulation results to exhibit the effectiveness of QuKerNet.

FS is necessary to alleviate the vanishing similarity issue. Here we show how feature selection affects the performance of the tailored MNIST (with 150 training examples sampled from the distilled dataset). The focus on the tailored MNIST rather than the rest two datasets is because of its high-dimensionality feature. The results are visualized in Fig. 3.9(b). The left line chart indicates that when L_0 is fixed, as p increases, KV continuously decreases. More precisely, when $L_0 = 5$, during the process of increasing p from 8 to 40, KV decreases from approximately 0.05 to 0.001. The right line chart reveals that without feature selection, even if $L_0 = 1$, the KV is close to 0.0001. By contrast, after feature selection, it is around 0.001 even when $L_0 = 9$.

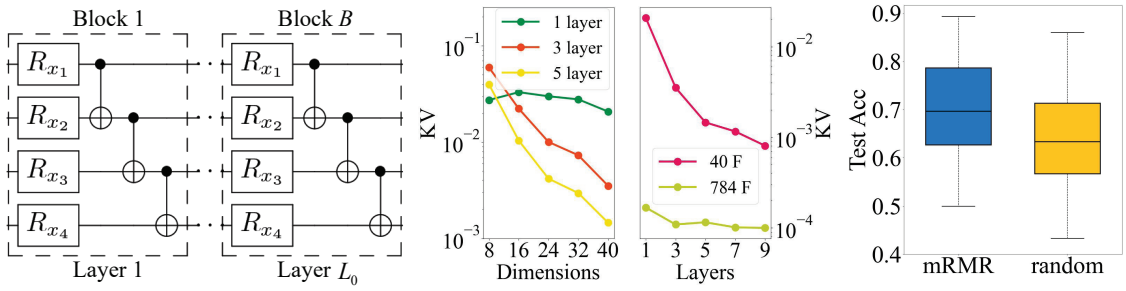


FIGURE 3.9. **Role of feature selection of QuKerNet.** (a) The depiction of hardware efficient ansatz (HEA). Each layer is composed of R_X gates where the rotational angle on the i-th qubit amounts to the i-th feature of the input data, and CNOT gates acting on the adjacent qubits. (b) Alleviation of vanishing similarity by feature selection. The label of "KV" represents kernel variance. Both "F" and 'dimensions' refer to the feature dimension. (c) Comparison with two feature selection methods, i.e., mRMR and random selection.

FS versus random pick. To further explore the role of feature selection, we searched for a 40-dimensional feature subset from the tailored MNIST using two methods (mRMR and random pick) to construct two new datasets. Additionally, we randomly generated 300 kernels with $L_0 \in \{1, 2, 3, 4, 5\}$ to conduct kernel learning on these two datasets. The box chart in Fig. 3.9(c) highlights that the data preprocessed by feature selection attains a higher classification accuracy. Taken together, it can be concluded that the feature selection technique can alleviate the vanishing similarity issue to a certain extent and improve the learning performance of quantum kernels.

KTA is a good surrogate of training accuracy. To avoid the expensive computational overhead, QuKerNet adopts KTA instead of training accuracy to construct the labeled dataset to train the neural predictor, as elucidated in Sec. 3.3. To verify the effectiveness of this replacement, we investigate the correlation between the performance of our proposal under the loss functions in Eq. (3.4) and Eq. (3.1) on the same tiny MNIST at two learning stages i.e., QuKerNet-1 (only optimize the circuit layout S) and QuKerNet-2 (simultaneously optimize the circuit layout S and variational parameters $\boldsymbol{\theta}$), respectively. The tiny MNIST contains 50 train and 50 test examples, with labels ranging from 0 to 4. Each example consists of 4 features selected by mRMR. In this case, $N = 4, L_0 = 1, |\Theta| = 50, |\mathcal{S}| = 72$ and $\mathcal{S}$ contains in total $3 * 3 * 2^3 = 72$ different circuit layouts. The restricted search space allows us to explore the whole search space with a thorough

analysis. The simulation results are exhibited in Fig. 3.10. In this case, we use the Pearson correlation coefficient (PCC, higher is better) to evaluate the validity of KTA. From Fig. 3.10, the PCCs between the training accuracy of kernels found by QuKerNet and the train accuracy of kernels selected by Eq. (3.1), are 0.9918 (QukerNet-1) and 0.9766 (QukerNet-2), respectively. This indicates we can employ Eq. (3.4) instead of Eq. (3.1) to implement kernel learning. Besides, KTA has been demonstrated to be a reasonable surrogate as the PCCs between the KTA of kernels selected by QuKerNet and the training accuracy of kernels found through Eq. (3.1) are both close to 1 (i.e., 0.9881 in QukerNet-1 and 0.9821 in QukerNet-2). These results validate the feasibility of using KTA instead of training accuracy to construct the labeled dataset to train the neural predictor.

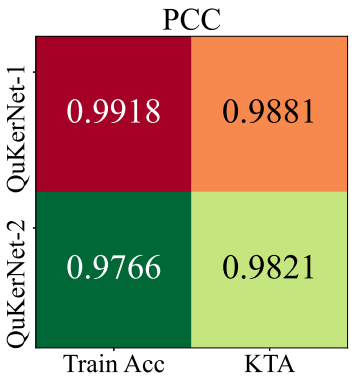


FIGURE 3.10. Correlation of kernels optimized under different loss functions.

Moreover, we analyze the time complexity of obtaining KTA and the training accuracy of a kernel to demonstrate the computational efficiency of calculating KTA.

For a given dataset $\hat{\mathcal{D}} = \{\hat{\boldsymbol{x}}^{(i)}, y^{(i)}\}_{i=1}^{n}$, as stated in [163], the time complexity of training and test processes of the kernelised Support Vector Machines are all $O(n^3)$, where n is the number of the data. So the time complexity of calculating train accuracy is $O(n^3)$. While the time complexity of computing KTA is $O(n^2 * p)$, as there are $n * n$ entries in kernel matrix W and getting each entry needs $O(p)$ time complexity, where p refers to the number of features of $\hat{\boldsymbol{x}}^{(i)}$. In general, $p \ll n$, which indicates that the computational efficiency of KTA is much higher than that of train accuracy.

Enhanced performance after each step of QuKerNet. We next exploit the performance of QuKerNet at different learning stages when learning the three datasets introduced in '**Datasets**'. We set $M = 1000, 500, 1000$ for tailored MNIST, tailored CC, and synthetic dataset, respectively. As shown in Table 3.2, QuKerNet is able to find kernels that perform better than those in the training set, indicating that it has the ability to genuinely grasp certain rules to infer kernel performance based on the circuit layout. Furthermore, it can be

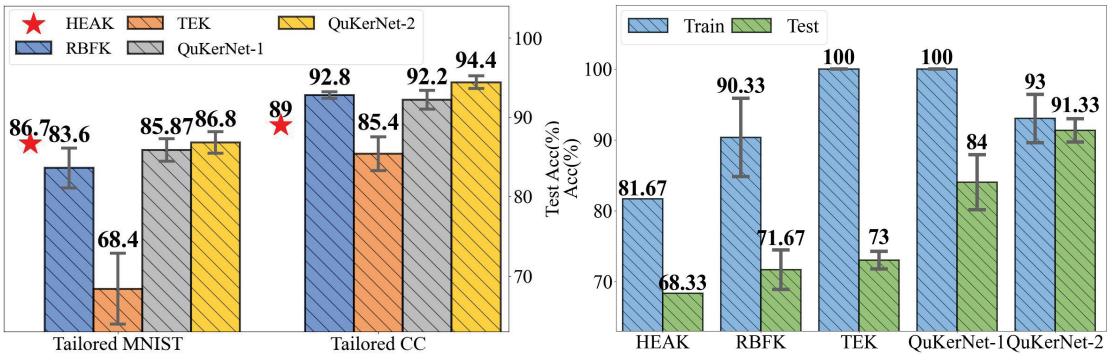


FIGURE 3.11. **Numerical results of QuKerNet.** (a) The best kernels discovered by TEK, QuKerNet-1, and QuKerNet-2 on tailored MNIST and tailored CC datasets. (b) The training and test accuracy for the best kernels searched by TEK, QuKerNet-1, and QuKerNet-2 on the synthetic dataset. Note that the results of HEAK have no standard deviation as it is a fixed kernel without any adjustable parameters.

observed that the top performance of the kernels selected by QuKerNet keeps improving in both QuKerNet-1 and QuKerNet-2, demonstrating that optimizing both S and $\boldsymbol{\theta}$ is necessary and effective.

TABLE 3.2. The top test accuracy for three datasets on different stages.

	Training data	Candidate	Fine-tune
Tailored MNIST	82.13 ± 0.27	85.87 ± 1.43	$\mathbf{86.80} \pm 1.36$
Tailored CC	87.00 ± 0.00	92.20 ± 1.17	$\mathbf{94.40} \pm 0.80$
Synthetic dataset	80.33 ± 0.67	84.00 ± 3.89	$\mathbf{91.33} \pm 1.63$

Comparison with other kernels. To further verify the performance of kernels found by QuKerNet is superior to unoptimized or individually optimized (S or $\boldsymbol{\theta}$) algorithms, we compare the performance of five different methods, Radial Basis Function Kernels (RBFK) [21], HEA-based quantum kernel (HEAK) [160], Training Embedding Kernels (TEK) [83], QuKerNet-1, and QuKerNet-2 in learning three datasets aforementioned. The statistical results of five random experiments conducted on tailored MNIST, tailored CC, and synthetic datasets are shown in Fig. 3.11. Compared to the other algorithms, QuKerNet-2 has achieved the highest test accuracy on three datasets (tailored MNIST: 86.8%, tailored CC: 94.4%, synthetic data: 91.33%), showcasing its superiority.

Potential advantages. We then apply HEAK, RBFK, TEK, QuKerNet-1, and QukerNet-2 to learn the synthetic dataset, with the purpose of exploring whether the kernels designed by QuKerNet may possess certain quantum advantages. From Fig. 3.11(b), it can be observed

that there is a huge performance gap between TEK and the kernel designed by QuKerNet (i.e., 73% versus 91.33%), which demonstrates the importance of optimizing S. Besides, the RBFK achieves a very low test accuracy compared to quantum kernels, including QuKerNet (i.e., 71.67% versus 91.33%), implying the potential of QuKerNet compared to classical kernels. Moreover, the small gap between train and test accuracy hints good generalization ability of QuKerNet compared to other kernels. Furthermore, the results of QuKerNet-1 and QuKerNet-2 (16% versus 1.67%) suggest that optimizing $\boldsymbol{\theta}$ is crucial to improve the generalization of QuKerNet. At last, we provide the optimal circuit layouts obtained through QuKerNet on the systhetic dataset, which is shown in Fig. 3.12.

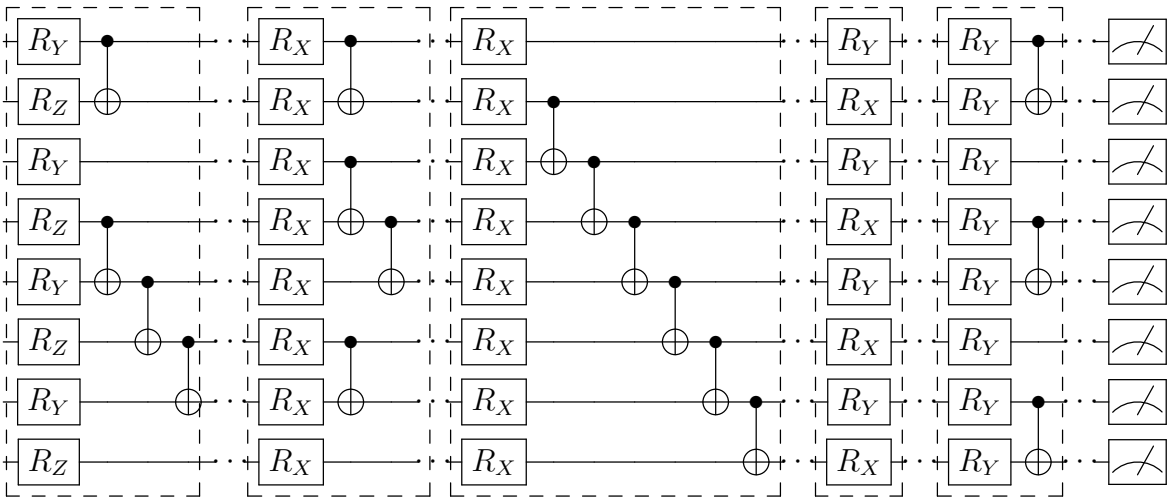
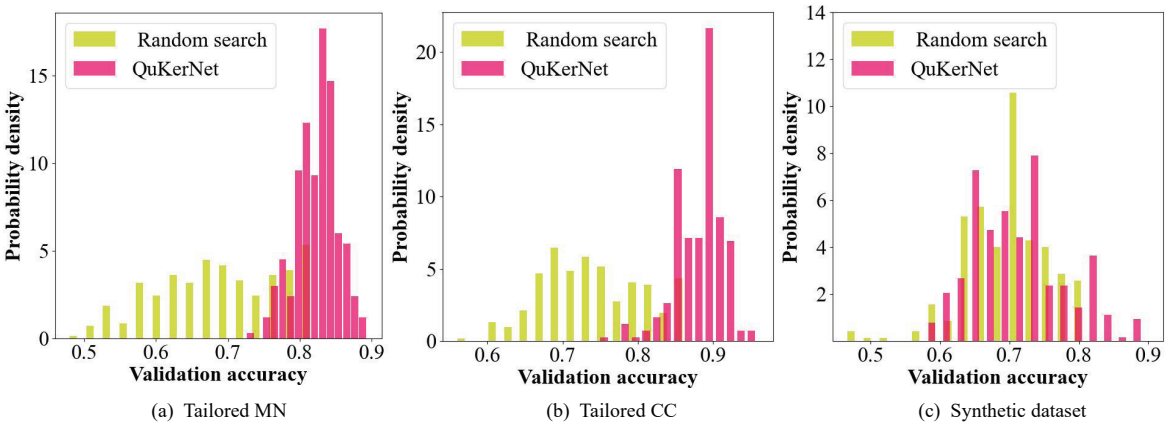


FIGURE 3.12. The layout of selected best encoding circuit for synthetic dataset (the layout of each dashed section will be repeated five times).

QuKerNet versus Random Search.

To further demonstrate the superiority of the QuKerNet over the random search method, we conducted a specific validation on the three datasets introduced in '**Datasets**' to evaluate the performance of the kernels based on circuit layouts searched by QuKerNet-1 and random search. In this case, $N = 8, L_0 \in \{1, 2, 3, 4, 5\}, |\mathcal{S}| = 50000, k = 200$. The performance of 200 kernels selected by QuKerNet-1 and random search are shown in Fig. 3.13. On tailored MNIST and tailored CC datasets, most of the kernels selected by QuKerNet-1 have better performance than the randomly searched kernels (i.e., [75%, 88%] versus [50%, 75%] on tailored MNIST). For the synthetic dataset, although the performance of the majority



(a) Tailored MN (b) Tailored CC (c) Synthetic dataset

FIGURE 3.13. Validation accuracy histogram between circuit layout from random search and QuKerNet. QuKerNet performs best.

of kernels obtained through these two methods overlaps, there are still a few kernels that outperform those found by random search. The above findings are sufficient to demonstrate that our method is indeed capable of discovering circuit layouts that are superior to those found through random search for constructing better kernels.

Performance under real device noise. To further verify the adaptability of QuKerNet, we compared the performance of the quantum kernel searched by QuKerNet in real device noise and noiseless cases. And we employ ibmq_quito to conduct experiments, the noisy parameters are shown in Table 3.3. Due to the time-consuming nature of noise experiments, in order to obtain experimental results within a reasonable time, we took one-third of the synthetic data to generate new synthetic data. Furthermore, PCA is used to reduce the dimensionality of the data to 8 dimensions. In this case, $M = 500, N = 4, L_0 \in \{1, 2, 3\}, |\mathcal{S}| = 20000, k = 10, |\Theta| = 10$. All experiments were run three times with different random seeds ($\gamma \in \{1, 2, 3\}/(p \text{ Var}[\boldsymbol{x}_j^{(i)}])$ for RBFK). Experimental results are shown in Fig. 3.14(a). Fig. 3.14(a) shows that the performance gaps of the kernels between the noise and noiseless cases identified by QuKerNet are not greater than 3.5% (i.e., 61.67% versus 63.33% for QuKerNet-1, 70% versus 73.33% for QuKerNet-2). Although HEAK does not show any difference in performance between the noise and noiseless conditions, its performance is significantly lower than the kernels selected by QuKerNet (i.e., 40% versus 70% in noise

situations). These results indicate that QuKerNet has good adaptability to real noise conditions and can be effectively used on practical devices.

TABLE 3.3. **The noisy parameters of ibmq_quito**. T_1 and T_2 are the longitudinal and transverse relaxation time respectively. "F" represents the qubit frequency. "RE" refers to readout error of the given qubit.

Parameter	Q1	Q2	Q3	Q4
$T_1(\mu s)$	48.21	60.20	30.10	70.56
$T_1(\mu s)$	26.82	89.08	14.46	13.10
F(GHz)	5.30	5.08	5.32	5.16
RE	0.0443	0.0220	0.1111	0.0450

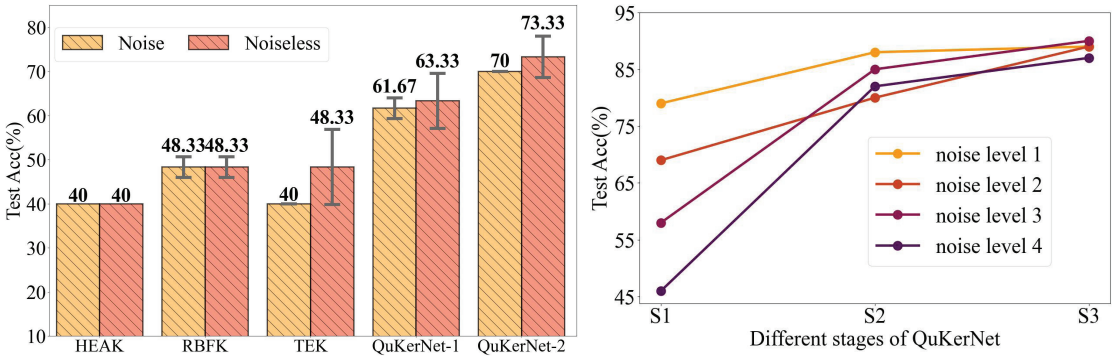


FIGURE 3.14. **Noise results of QuKerNet.** (a) The best kernels discovered by TEK, QuKerNet-1, and QuKerNet-2 on systhetic dataset in real device noise and noiseless situations. Note that the RBFK has the same results in noise and noiseless cases as it is a classical kernel method. (b) The effect of different noise levels for QuKerNet on a small part of the tailored CC dataset. S1, S2 and S3 represent the training stage, candidate stage, and fine-tuning stage, respectively. The optimal performance of kernels in each stage is shown in the line chart.

Performance under different noise levels. To further verify the adaptability of QuKerNet, we inspected the performance of the quantum kernel searched by QuKerNet in different noise levels. As noise simulation is time-consuming, we employ the PennyLane to simulate the noisy quantum models, and we only select 40 samples on labels 0 and 1 from the tailored CC with a 20/20 train/test split to construct the toy CC. Then we employ PCA to reduce the dimension of toy CC to 12 dimensions. And we set up 4 noise levels are shown in Table 3.4. Besides, we set $M = 1500, N = 4, L_0 \in \{1, 2, 3\}, |\mathcal{S}| = 50000, k = 10, |\Theta| = 10$ to conduct our experiments. And the statistical results of five random trials are shown in Fig. 3.14(b). From Fig. 3.14(b), we know that although the performance of the kernels in training data will decrease with the increasing noise (i.e., from 79% to 46%), QuKerNet can

still find kernels with high performance (i.e., around 90% test accuracy) in all noise levels, indicating that QuKerNet has better robustness and adaptability.

TABLE 3.4. The details of four noise levels (DQEP refers to depolarizing quantum error probabilities).

	level 1	level 2	level 3	level 4
DQEP of 1-qubit gate	0.005	0.01	0.015	0.02
DQEP of 2-qubit gate	0.05	0.1	0.15	0.2

Adaptive classical kernel vs QuKerNet. To better demonstrate the adaptability of QuKerNet, we compare the performance of Neural Kernel Network (NKN) and QuKerNet. NKN is a adaptive kernel construction method proposed in paper [155]. We conducted 5 random experiments with the default parameters based on the official code of [155], and the results are summarized in the table below. The test accuracy are provided in the Table 3.5. The performance of NKN is lower than our method on all three datasets (i.e., 47.67% vs 94.4% on Tailored CC). Additionally, we observed that the training accuracy of NKN on the three dataset is close to 100%, indicating potential overfitting. This could be attributed to two factors: 1) We used default parameters without any optimization, and 2) The limited size of the training samples. These results demonstrate that QuKerNet's adaptive capability is superior to those of classic kernel methods.

TABLE 3.5. The test accuracy for three datasets of different methods.

	RBFK	QuKerNet	NKN
Tailored MNIST	83.60 ± 2.52	$\mathbf{86.80} \pm 1.36$	19.73 ± 2.78
Tailored CC	92.80 ± 0.40	$\mathbf{94.40} \pm 0.80$	47.67 ± 9.23
Synthetic dataset	71.67 ± 2.79	$\mathbf{91.33} \pm 1.63$	50.60 ± 5.35

Activate learning. Our proposed scheme in QuKerNet offers a general framework that can be seamlessly integrated with various advanced methods and techniques. Here, we implement the method of Bayesian optimization (BO) for quantum kernel search using Optuna [5] to show the flexibility of our approach. For the BO, We set the number of iterations to 1000, and the experimental results are shown in the Table 3.6. The kernel searched by Bayesian optimization did not surpass the random search approach in search space (i.e., 85.6% vs 86.8% on Tailored MNIST), which may be due to the limited number of iterations. And this experiment fully demonstrates the flexibility of QuKerNet.

TABLE 3.6. BO vs random search using neural predictor.

	Training data	Candidate	Fine-tune
Tailored MNIST BO	82.13 ± 0.27	83.20 ± 1.66	85.60 ± 1.44
Tailored MNIST Ours	82.13 ± 0.27	85.87 ± 1.43	**86.80** ± 1.36
Tailored CC BO	87.00 ± 0.00	89.80 ± 1.72	93.00 ± 1.41
Tailored CC Ours	87.00 ± 0.00	92.20 ± 1.17	**94.40** ± 0.80
Synthetic dataset BO	80.33 ± 0.67	76.67 ± 6.15	89.34 ± 2.26
Synthetic dataset Ours	80.33 ± 0.67	84.00 ± 3.89	**91.33** ± 1.63

20 qubits simulation results. The experimental results of 20 qubits on three datasets are shown in Table 3.7. We set $M = 50, L_0 = 1, k = 5, |\Theta| = 5$, to conduct three random experiments. From the table, we also find that our method works well. Since our method can still search the kernels with good performance on the three datasets (e.g., from 84.89% to 86.45% on Tailored MNIST dataset). This once again demonstrates the effectiveness of QuKerNet, even in large-scale quantum systems.

TABLE 3.7. 20 qubits simulation results on three datasets.

	Training data	Candidate	Fine-tune
Tailored MNIST	84.89 ± 0.31	86.45 ± 0.32	86.45 ± 0.32
Tailored CC	92.00 ± 0.00	92.20 ± 0.00	95.00 ± 0.00
Synthetic dataset	70.00 ± 0.00	73.33 ± 4.71	75.55 ± 3.13

Experiments on higher-dimensional MNIST data. To better demonstrate the effectiveness of our algorithm, we increased the dimensionality of the Tailored MNIST dataset to 80, to test our algorithm. The results in Table 3.8 demonstrate our method can still find kernels with better performance on higher-dimensional data. In addition, we found that the performance of both HEAK (86.7 vs 73.33) and TEK (68.4 vs 34.68) decreases with increasing dimensionality. This may because they all encounter the vanishing similarity. On the other hand, as more information is available, the performance of both RBFK (83.6 vs 87.6) and QuKerNet (86.8 vs 90.27) are improved. This once again demonstrates that QuKerNet is universal and can overcome the vanishing similarity.

Trade-off between the performance gain and optimization overhead. The main trade-off between the performance gain and optimization overhead originated from finding quantum feature map with good performance within a large exponential search space.

TABLE 3.8. 40 and 80 dimension simulation results on Tailored MNIST.

Method / Dimension	RBFK	HEAK	TEK	QuKerNet
40	83.60 ± 2.52	86.70 ± 0.00	68.40±4.46	**86.80** ± 1.36
80	87.60 ± 0.03	73.33 ± 0.00	34.68±17.84	**90.27** ± 0.53

We have conducted relevant experiments to investigate this trade-off. The main results is that, in most cases, increasing the number of fine-tuning iterations is equivalent to increase the optimization overhead, can lead to performance improvements. Detailed results are displayed in the Table 3.9. We observes that in most of cases when $|\Theta|$ is fixed, the performance of the selected kernel increases with an increase in k (91.8% vs 93% on Tailored CC when $|\Theta|$=5). Conversely, when k is fixed, the larger the value of $|\Theta|$, the better the performance of the searched kernel (84% vs 91.33% on Synthetic dataset when k=10). These findings suggest that higher optimization overhead can lead to greater performance gain, within a certain range.

TABLE 3.9. Trade-off results on three datasets.

	$\|\Theta\| = 0$	$\|\Theta\| = 5$	$\|\Theta\| = 10$	$\|\Theta\| = 20$
Tailored MNIST $k = 5$	84.53 ± 2.44	86.27 ± 1.72	86.27 ± 1.72	86.80 ± 1.36
Tailored MNIST $k = 10$	85.87 ± 1.43	86.53 ± 1.49	86.53 ± 1.49	86.80 ± 1.36
Tailored CC $k = 5$	91.20 ± 1.47	91.80 ± 1.72	93.20 ± 0.75	94.40 ± 0.80
Tailored CC $k = 10$	92.20 ± 1.17	93.00 ± 1.10	93.40 ± 1.02	94.40 ± 0.80
Synthetic dataset $k = 5$	78.67 ± 3.86	87.00 ± 4.14	89.00 ± 1.70	89.00 ± 1.70
Synthetic dataset $k = 10$	84.00 ± 3.89	90.00 ± 3.16	91.33 ± 1.63	91.33 ± 1.63

3.4.4 Discussion

In this section, we will start with a concise literature review, following with the variants of our proposal and end with the experimental summary of QuKerNet.

3.4.4.1 Related work

Prior literature most related to our work can be classified into two categories: quantum kernels and Quantum Architecture Search (QAS). Here we separately review the connections and differences between these studies and our work.

Kernel-based quantum algorithms. Previous studies of quantum kernels can be divided into two groups. The first group is data-independent quantum kernels [17, 80, 160, 81], which use a fixed circuit to embed the data. These quantum kernels may not have good prediction as the quantum feature map may be inappropriate for the manipulated dataset. Another group is data-dependent quantum kernels [63, 113, 83], where the quantum feature is customized to the given dataset. The approaches of designing data-dependent quantum features either focus on modifying the circuit layout or tuning variational parameters, indicating that these methods can only achieve local optimal solutions. QuKerNet differs from prior works in the pursuit of the global optima, by simultaneously considering the circuit layout and variational parameters. In this perspective, our work is a highly generalized version of prior kernel algorithms, where they are all special cases of our algorithm.

QAS. QAS aims to automatically design the architecture of quantum neural networks that can attain high performance for a specified learning task. Essentially, QAS is a discrete-continuous joint optimization problem. Depending on the different optimization methods, QAS can be classified into heuristic-based QAS [180, 156], reinforcement learning based QAS [127], Bayesian-based QAS [52], and other methods [76, 185, 75, 184, 114, 175, 49, 108]. Different from QAS, QuKerNet orients to enhance quantum kernels. The fundamental difference between neural networks and kernels hints the hardness of directly employing QAS to automatically design quantum kernels.

3.4.4.2 Variants of QuKerNet

Our proposed scheme in QuKerNet offers a general framework that can be seamlessly integrated with various advanced methods and techniques. For instance, the gate set $\mathcal{G}$ used to construct the search space can be modified to accommodate different quantum hardware platforms, enabling flexibility and adaptability. Furthermore, different feature selection methods such as mRMR [133], Principal Component Analysis (PCA) [2] and Locality Preserving Projections (LPP) [74] and other methods [100, 178, 104, 188, 187] can be employed to handle diverse high-dimensional datasets effectively. In terms of the neural predictor, alternative backbones and encoding methods can be explored to enhance its

performance in capturing the relationship between circuit layouts and kernel performance, i.e., graph neural network (GNN) backbone plus graph encoding strategy [76], Convolutional Neural Network (CNN) backbone with image encoding [185]. Additionally, advanced training methods can be employed to optimize the neural predictor with improved performance, i.e., data augmentation [85, 82], Dropout [154], early stopping [27]. Moreover, the fine-tuning process can benefit from the incorporation of more sophisticated heuristic methods, enhancing the overall optimization process.

3.4.4.3 Experimental summary

In this section, we first demonstrated the necessity of feature selection for alleviating the vanishing similarity issue. Then, We have elaborated that KTA is a good surrogate for training accuracy in kernel design. After that, We verified the necessity of each step in the framework of QuKerNet. Finally, we showcased the superiority of QuKerNet on different baselines (diverse kernels, noise settings, large-scale quantum system and so on). These findings fully demonstrate the effectiveness and practicality of QuKerNet.

3.5 Summary

In this chapter, we propose QuKerNet to automatically design problem-specific quantum feature maps, which greatly improves the power of quantum kernels under NISQ settings. In contrast with prior advantageous quantum kernels established on well-structured problems, our proposal does not require prior information on tasks at hand. This characteristic underpins the potential of QuKerNet to use NISQ machines to conquer realistic tasks with computational merits.

CHAPTER 4

Learning the Parameters of Quantum Neural Networks with Graph HyperNetworks

---

Variational Quantum Algorithms (VQAs) are gaining prominence in the era of near-term quantum computing, as it can simultaneously utilize quantum and classical resources to obtain possible quantum advantage. However, the trainability of VQAs becomes a significant obstacle to achieving potential quantum advantage, especially on large-scale quantum circuits. In this chapter, we propose a universal parameter initialization approach, named QuGHN, to generate near-optimal parameters to initialize the quantum circuit, thereby enhancing the trainability of Quantum Neural Networks (QNNs). The remainder of this chapter is organized as follows: in Sec. 4.1, we mainly introduce three mainstream solutions for improving the trainability of QNNs, the idea of parameter initialization, and the matters of the current parameter initialization methods. Subsequently, the mathematical form of the problem to be solved in parameter prediction will be given in Sec. 4.2. In Sec. 4.3, we elaborate the implementation details of QuGHN module by module. Then we will evaluate the performance of various aspects of the parameters generated by QuGHN in Sec. 4.4, such as effectiveness, robustness, and convergence. Lastly, we conclude the content of this chapter in Sec. 4.5.

4.1 Introduction

Variational Quantum Algorithms (VQAs) are a class of hybrid algorithm using classical optimizers to train Quantum Neural Networks (QNNs) that can handle complex tasks. Practically, with the advantages of quantum computing, VQAs can be useful in solving optimization and eigenvalue problems, showing great application prospects [8, 145]. Therefore, VQAs has

gradually become a leading strategy for solving various valuable problems in various fields including chemistry [117], finance [53], material sciences [12] and logistics [34].

Although the introduction of VQAs provides new possibilities for using the advantages of quantum computing to solve problems that cannot be handled by classical resources, its training is a challenging because of the notorious phenomenon of barren plateaus (BPs) in the training landscapes of QNNs [118]. When barren plateaus appears, the variance of gradients diminishes exponentially with the number of qubits in QNNs increases, making it impossible to train QNNs and thus the QNNs are unable to be used for various subsequent tasks.

In order to alleviate the BPs phenomenon that occurs during QNNs training, many scholars have made different attempts. There are currently three main solutions, namely, designing different ansatz [68, 172, 49], smart parameter initialization schemes [109, 93, 138, 182], and over-parameterization [94, 177]. The design of ansatz usually weakens the expressive power of QNN, thereby affecting the performance of the model [78]. On the other hand, the over-parameterization method cannot be applied to the case where the depth of the circuit is exponential in qubits, because the number of parameters required for over-parameterization is comparable to the scale of the parameters in QNNs [94, 169]. Therefore, parameter initialization stands out as a more practical solution.

The core idea of parameter initialization is to set the parameters of neural networks to those parameters with large gradients and close to the optimal solution, thereby accelerating the convergence of the QNNs. This type of initialized parameters is also beneficial for QNNs without BP. At present, there are many parameter initialization strategies in the community, and these methods are mainly divided into distribution-based initialization and meta-learning-based initialization methods. [182] as a representative work of the former, it designed a more suitable Gaussian distribution for parameter initialization. While methods based on meta-learning include [166, 31, 146]. Despite their initial attempts, these methods still have some problems. For example, the former is a static initialization strategies cannot adapt to distinct tasks and data, while the latter is usually only applicable to QNNs with the same number of parameters, which greatly weakens the practicality of this method. Therefore, how to design a flexible and universal parameter initialization scheme is particularly important.

This method needs to have two characteristics: *(i) it can adapt to different circuit layout and different parameter amounts; (ii) it can generate high quality initialization parameters.*

In this work, we devise a general approach to initialize the parameters of the QNNs. Specifically, we seem the parameter initialization problem as an optimal parameter prediction problem, and then propose a flexible method dubbed parameter initialization based on the Quantum Graph HyperNetwork [181] (QuGHN) to solve this learning problem, which can automatically predict the best initialization parameters based on the layout information of the circuit itself and the specific task to dynamically initialize the QNNs. In addition, it can be flexibly applied to systems with different number of qubits without retraining a new model.

The core components of QuGHN are graph neural network (GNN) and hypernetwork. GNN is used to encode the directed acyclic graph converted from QNNs into graph embedding, thereby depicting the structural information of the circuit. This kind of GNN can continuously adjust the embedding of the graph according to the circuit layout, which is equivalent to continuously adjusting the embedding in the latent space based on the circuit layout and the optimal parameters, so that it can approach the embedding of the optimal parameters. By training GNNs to realize message passing at the graph level, the structural information of the circuit is introduced, thereby improving the quality of the embedding. On the other hand, the hypernetwork is used to decode the graph embedding into the parameters of the QNNs. The use of hypernetwork allows us to predict the parameters of QNNs with different circuit layouts and different scales of quantum system. Extensive experiments are conducted to demonstrate the versatility, robustness and efficacy of QuGHN.

4.2 Problem Setup

In this section, we first introduce the principles of parameter prediction for unseen architecture, then introduce the mechanism of Graph Hypernetwork. Finally, we elaborate the mathematical expression of our solution.

4.2.1 Principles of Parameter Prediction

The objective function for initializing QNN parameters using the hypernetwork has the following form:

$$\arg\min_{\boldsymbol{\phi}} \sum\nolimits_{j=1}^{n} \sum\nolimits_{i=1}^{m} \mathcal{L}(f(\boldsymbol{x}_j; a_i, H(a_i; \boldsymbol{\varphi})), y_j) \tag{4.1}$$

where the a_i is a quantum neural network, $\boldsymbol{\varphi}$ is the corresponding parameters in the hypernetwork H, $H(a_i; \boldsymbol{\varphi})$ represents the predicted parameters related to the a_i. While $\boldsymbol{x}_j$ and y_j refer to the input data of specific task.

4.2.2 Mechanism of Graph Hypernetwork

Due to recent advancements of graph neural networks on graph-structured data [4, 6, 144], also in order to achieve better prediction results, we are considering using Graph Hypernetwork (GHN for short) to replace the hypernetwork. GHN mainly consists of an encoder and a decoder. The encoder is a GNN used to encode the neural network structure into an embedding, while the decoder is a hypernetwork used to decode the embedding into the parameters of the neural network. Therefore, GHN mainly solves the following problems:

$$\begin{aligned} \boldsymbol{\theta} &= \{\boldsymbol{\theta}_v | v \in \mathcal{V}\} \\ &= \{H(\boldsymbol{h}_v^{(T)}; \boldsymbol{\varphi}) | v \in \mathcal{V}\} \\ &= \{H(\boldsymbol{h}; \boldsymbol{\varphi}) | \boldsymbol{h} \in G^{(T)}(\{\boldsymbol{h}_v^{(0)} | v \in \mathcal{V}\}; \boldsymbol{\phi})\} \\ &= GHN(\mathcal{A}; \boldsymbol{\phi}, \boldsymbol{\varphi}) \end{aligned} \tag{4.2}$$

where $\boldsymbol{\theta}$ represents the predicted parameters of parameterized quantum gates in the QNN, and $\mathcal{A}$ represents the directed acyclic graph (DAG) of the QNN, $\mathcal{V}$ is the node set with all the parameterized quantum gates in DAG, $\boldsymbol{\varphi}$ is the parameter of the hypernet H, and $\boldsymbol{\phi}$ is the parameter of the GNN. We use DAG to represent QNN because each quantum gate can be viewed as a node in a DAG, and the order of operations is represented by the edges between nodes. Moreover, the final operation of QNN is measurement, which yields a classic result, thus forming a DAG. While $\boldsymbol{h}_v^{(T)}$ represents the embedding of node v after T steps propagation through GNN, and G is the GNN used to update the embedding of the node. By using GNN,

one can capture the information flow in QNN, and this encoder-decoder architecture can use the layout information of QNN to predict parameters, thereby achieving better performance.

4.3 Implementation of QuGHN

When we replace the hypernetwork in Eq. (4.1) with GHN, we naturally introduce the following objective function:

$$\arg\min_{\boldsymbol{\phi},\boldsymbol{\varphi}} \sum\nolimits_{j=1}^{n} \sum\nolimits_{i=1}^{m} \mathcal{L}(f(\boldsymbol{x}_j; a_i, GHN(\mathcal{A}_i; \boldsymbol{\phi}; \boldsymbol{\varphi})), y_j) \tag{4.3}$$

where, $f(\cdot;\cdot,\cdot)$ is the output of QNNs. Since this algorithm involves the training of QNNs, we first review the basics of the QNNs. In general, QNNs mainly solves the following problems:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} C(\boldsymbol{\theta}) \tag{4.4}$$

where $\boldsymbol{\theta}$ is the trainable parameters in QNNs. And $C(\boldsymbol{\theta}) = \sum_{j=1}^{n} \ell(\mathrm{Tr}(OU(\boldsymbol{\theta})\rho_j U^{\dagger}(\boldsymbol{\theta})), y_j)$. ρ_j is a set of input quantum state, $U(\boldsymbol{\theta})$ denotes the quantum circuits that acting on the ρ_j, while O refers to a measurement at the end of the circuit. Here, $U(\cdot)$ corresponds to a_i in Eq. (4.3), and $\boldsymbol{x}_j$ can be applied to either O or ρ, depending on the tasks. For example, in a classification task on MNIST [96], $\boldsymbol{x}_j$ is applied to ρ, while in a task like finding the ground state energy of Hamiltonian with different coupling parameters, $\boldsymbol{x}_j$ is applied to O. Therefore, more generally, we substitute $U(\cdot)$ into Eq. (4.3) and assign $\boldsymbol{x}_j$ to both O and ρ, obtaining the following objective function:

$$\begin{aligned} \arg\min_{\boldsymbol{\phi},\boldsymbol{\varphi}} \sum\nolimits_{j=1}^{n} \sum\nolimits_{i=1}^{m} \mathcal{L}(\mathrm{Tr}[O_i(\boldsymbol{x}_j^O) U_i(GHN(\mathcal{A}_i; \phi, \varphi)) \\ \rho_{i,j}(\boldsymbol{x}_j^{\rho}) U_i^{\dagger}(GHN(\mathcal{A}_i; \boldsymbol{\phi}, \boldsymbol{\varphi}))], y_j) \end{aligned} \tag{4.5}$$

where, $\mathrm{Tr}[\cdot]$ refers to the output from the quantum circuit, the $\boldsymbol{x}_j^O$ and $\boldsymbol{x}_j^{\rho}$ represent the parameters acting on the observable O and quantum state ρ, respectively. While the parameters $\boldsymbol{\theta}$ of the quantum circuit are updated through the classical neural network, $GHN(\cdot; \boldsymbol{\phi}, \boldsymbol{\varphi})$.

To facilitate the understanding of our algorithm, we display the intuition of QuGHN in Fig. 4.1. Specifically, in order to solve the problem that the previous parameter initialization method

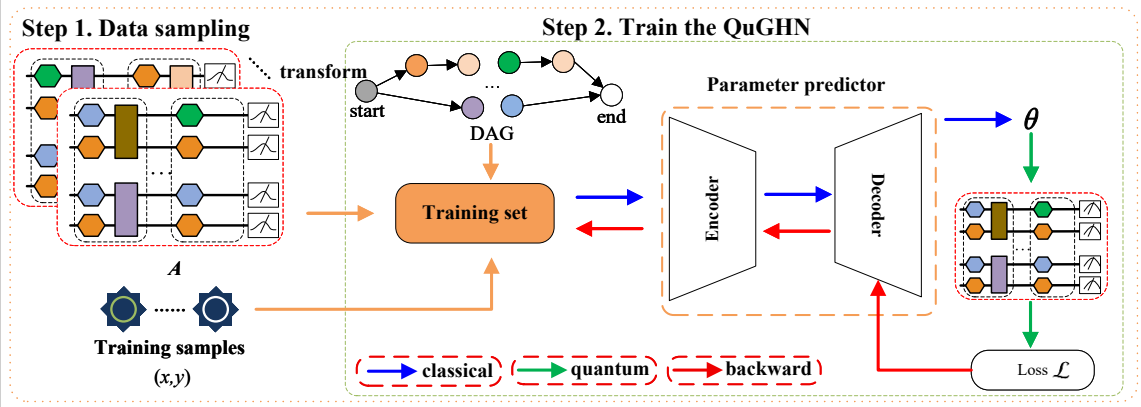


FIGURE 4.1. **The workflow of QuGHN.** In step 1, we randomly sample the circuits built with a certain ansatz, and then collect two types of data. One is to convert the circuit into DAG as the training data $\mathcal{A}$, and the other is the features $\boldsymbol{x}$ and label y of a certain task (in this chapter, $\boldsymbol{x}$ corresponds to the coupling parameters of the Hamiltonians, y refers to the ground state energy of these parameterized Hamiltonians, while in quantum machine learning tasks, $\boldsymbol{x}$ can be images and y refers to the labels related to these images), and then this triplet of data ($\mathcal{A}$, $\boldsymbol{x}$, y) can be used as training data for training QuGHN based on Eq. (4.3). In step 2, we utilise ($\mathcal{A}$, $\boldsymbol{x}$, y) to train the decoder and encoder. The encoder first encodes the quantum gates into a feature vector according to the Eq. (4.6), and the decoder decodes the processed feature vector into parameters for initialization of the quantum circuit, which corresponds to the Eq. (4.7). This encoding and decoding process is the mechanism of the parameter predictor, corresponding to the process traversed by the **blue arrow** in the figure. Here, the predicted parameters $\boldsymbol{\theta}$ are loaded into the corresponding quantum circuit to calculate the loss value on the certain task, which corresponds to the **green arrow** in the figure. The loss calculated at the quantum circuits is used to update parameters of QuGHN. After continuous backpropagation (**red arrow**), the goal of minimizing the error between the predicted parameters and the optimal parameters is achieved. Then, the trained QuGHN can be used to generate the parameters of a QNN with new circuit layout, and these parameters can be used as initial parameters for the training of a QNN.

cannot adapt to quantum systems of various scales, we introduced the DAG to represent the circuits uniformly. In this case, circuits of different scales can be uniformly converted into DAG, so that unified predictions can be made. On the other hand, in order to improve the performance of parameter initialization, we employ GNN to capture the global and local information of the circuits, thereby facilitating the model training. Moreover, we consider training the encoder and decoder at the same time, so that the model can more effectively use the information contained in the quantum circuit layout itself to predict parameters.

As showed in Fig. 4.1, QuGHN mainly consists of three parts, namely data collection, parameter predictor and predictor training.

Data collection. The first step of our algorithm is to collect enough training data. And the data used for training mainly includes two parts, namely, the classical part and the quantum part. The first part is sampling samples for corresponding task training, which is denoted by $\mathcal{D} = \{\boldsymbol{x}_{(i)}, y_{(i)}\}_{i=1}^{n}$, where $\boldsymbol{x}_{(i)} \in \mathbb{R}^d$ and $y_{(i)} \in \mathbb{R}$ refer to the i-th example with the feature dimension d and the corresponding label, respectively. And the quantum part mainly includes QNNs $\boldsymbol{A} = [a_1, \ldots, a_m]$ and directed acyclic graphs transformed from $\boldsymbol{A}$ are $\mathcal{A} = [\mathcal{A}_1, \ldots, \mathcal{A}_m]$. In this paper, the $\boldsymbol{x}$ is the coupling parameters of the Hamiltonians, and the y is the ground state energy of these Hamiltonians. Although the $\boldsymbol{A}$ can be a QNN with different ansatz structures, in this paper, it is quantum circuits mainly constructed by Hardware-efficient ansatz and Hamiltonian variational ansatz. The $\mathcal{A}$ is a set of DAGs. Please refer to the Sec. 4.3.1 for details on how to convert a QNN into a DAG.

Parameter predictor. The parameter predictor consists of an encoder and a decoder. The encoder $G(\cdot, \boldsymbol{\phi})$ maps the discrete circuit layout into continuous feature embeddings $\boldsymbol{e}$:

$$\boldsymbol{e} = G(\mathcal{A}_i, \boldsymbol{\phi}) \tag{4.6}$$

where $\mathcal{A}_i$ is the DAG of the quantum circuit. $\boldsymbol{\phi}$ denotes the parameters of the GNN encoder. Next, we can use the decoder composed of the hypernet H to decode the feature embeddings of each parameterized gate in the quantum circuit into the parameters corresponding to the gate.

$$\boldsymbol{\theta} = H(\boldsymbol{e}, \boldsymbol{\varphi}) \tag{4.7}$$

where $\boldsymbol{\varphi}$ refers to the parameters of the hypernet H. $\boldsymbol{\theta}$ denotes the predicted parameters of the parameterized gate in quantum circuit. When we substitute Eq. (4.6) into Eq. (4.7), we get Eq. (4.2), that is, $\boldsymbol{\theta} = GHN(\mathcal{A}_i; \boldsymbol{\phi}, \boldsymbol{\varphi})$, which corresponds to the mechanism of the parameter predictor. Next, we will introduce encoder and decoder in detail.

GNN-based encoder— GNN-based encoder is a crucial module in our method as it can update node features by passing messages between nodes and has strong learning capabilities in graph-structured data [106]. Essentially, all quantum circuits can be viewed as a DAG. This property makes it natural to use the GNN encoder to update node features of DAG of quantum circuits. In this paper, we will first encode different quantum gates into vectors $\boldsymbol{h}_v \in \mathbb{R}^D$

with uniform length as the node feature. Then we employ GNN to update the node feature according to the following formula:

$$\boldsymbol{h}_v^{(t+1)} = \begin{cases} GRU(\boldsymbol{h}_v^{(t)}, \boldsymbol{m}_v^{(t)}), & \text{if node } v \text{ is active.} \\ \boldsymbol{h}_v^{(t)}, & \text{otherwise.} \end{cases} \tag{4.8}$$

where GRU is the gated recurrent unit (GRU) [33] and $\boldsymbol{m}_v^{(t)} = \sum_{u \in N_{in}(v)} M(\boldsymbol{h}_u^{(t)})$ is the message that node v received at time step t. $N_{in}(v)$ refers to the set of neighbors with incoming edges pointing towards v. $M(\cdot)$ represents the message function that used to pass messages to neighboring nodes, essentially, it can gather all neighbor information of a node to that node, thereby achieving the effect of aggregating information. And it can be implement through multi-layer perceptron (MLP). Given a graph $\mathcal{A}_i$, and the encoder $G(\cdot, \boldsymbol{\phi})$, the node embeddings can be updated with the following formula:

$$\{\boldsymbol{h}_v^{(t)} | v \in \mathcal{V}\} = G^{(t)}(\{\boldsymbol{h}_v^{(0)} | v \in \mathcal{V}\}, \phi) \tag{4.9}$$

where $\boldsymbol{h}_v^{(0)} = \boldsymbol{h}_v$ is the initial node features. Through this message passing between nodes, we can simultaneously capture the global information of the graph structure (aggregate graph level information) and local information (transmission between neighboring nodes), which can improve the performance of predicted parameters on models.

MLP-based decoder— The MLP-based decoder is a hypernetwork that can decode the node embeddings into parameters of neural networks. The input of this decoder is the output of the encoder $G(\cdot, \boldsymbol{\phi})$. Therefore, the predicted parameters can be obtained through:

$$\boldsymbol{\theta} = H(\boldsymbol{h}_v{}^{(t)}, \boldsymbol{\varphi}) \tag{4.10}$$

where $\boldsymbol{\varphi}$ refers to the parameters of the hypernetwork H. And there are many choices for H, such as MLP [143], convolutional neural network (CNN) [97], and so on. For simplicity, in this paper, we choose MLP as the H to decode node embeddings.

Predictor training. The parameter predictor is trained on a small number of coupling parameter pairs of Hamiltonians and their corresponding ground state energy in an end-to-end manner, i.e., the parameters of the encoder and decoder are simultaneously trained for one

loss function. Since we employ mean square error (MSE) as the loss function, the optimal parameters $\boldsymbol{\phi}^*$ and $\boldsymbol{\varphi}^*$ of the encoder and decoder can be achieved through the bi-level optimization paradigm.

$$\begin{aligned}\boldsymbol{\phi}^*, \boldsymbol{\varphi}^* = \arg\min_{\boldsymbol{\phi},\boldsymbol{\varphi}} \tfrac{1}{Bs} \textstyle\sum_{i=1}^{Bs} \Big(\mathrm{Tr}[O_i(\boldsymbol{x}_i^O) U_i(GHN(\mathcal{A}_i; \boldsymbol{\phi}, \boldsymbol{\varphi})) \\ \rho_i U_i^\dagger(GHN(\mathcal{A}_i; \boldsymbol{\phi}, \boldsymbol{\varphi}))] - y_i\Big)^2\end{aligned} \tag{4.11}$$

where ρ_i is $|0\rangle^{\otimes N}$ (N is the number of qubit in the quantum circuit), and Bs refers to the batch size. The $\boldsymbol{\phi}^*$ and $\boldsymbol{\varphi}^*$ are the optimal parameters of the encoder G and decoder H, respectively.

After the training is completed, we can use QuGHN to predict the parameters of QNNs based on the DAG of QNNs, that is $\boldsymbol{\theta} = GHN(\mathcal{A}; \boldsymbol{\phi}^*, \boldsymbol{\varphi}^*)$.

4.3.1 Implementation Details of QuGHN

In this section, we introduce the missing details of QuGHN. Specifically, we mainly illustrate the data collection, implementation details of the encoder and decoder of QuGHN, as well as training details, followed by the summarization of its pseudocode.

Data Construction. The data used for training QuGHN consists of three parts in our experiments. The first two are data points and labels, which constitutes the dataset $\mathcal{D} = \{\boldsymbol{x}_{(i)}, y_{(i)}\}_{i=1}^n$, where $\boldsymbol{x}_{(i)} \in \mathbb{R}^d$ and $y_{(i)} \in \mathbb{R}$ denote the i-th point with the feature dimension d and the corresponding label, respectively. More precisely, $\boldsymbol{x}_i = [d_1, d_2]$ is a two-dimensional vector whose values are randomly selected from the distribution of Hamiltonian coupling parameters, $d_1, d_2 \sim p(J)$. For the Ising model, $d_1 \sim p(J_{\mathrm{Ising}})$, $d_2 \sim p(J_x)$. And for the Heisenberg model, $d_1 \sim p(J_{xy})$, $d_2 \sim p(J_z)$. While on the Cluster model, $d_1 \sim p(J_{\mathrm{I}})$, $d_2 \sim p(J_{\mathrm{C}})$. The label y is divided into exact label and approximate label. When $N \leq 12$, we collect the exact ground state energy of parameterized Hamiltonian as the label y. For $N > 12$, we obtain the approximate ground state energy through VQE, Matrix Product States (MPS) and Matrix Product Operators (MPO), as the true label y. It is worth noting that in the HEA experiment, in order to figure out how the algorithm performance changes with

the increase in the number of qubits, we uniformly use approximate ground state energy as label y to train all the models, thereby eliminating the impact of inconsistent label acquisition methods.

The final part of training data is $\mathcal{A}$, the directed acyclic graph (DAG) of quantum circuit. A toy example of transforming a quantum circuit into a DAG is depicted in Fig. 4.2. These are six kinds of nodes in the Fig. 4.2, a start node, an end node, and four types of gate nodes. The start node (colored in gery) and the end node (colored in white) represent the input (initial state) and output (measurement) of a circuit, respectively. The remaining nodes are the four types of gates that are commonly used in quantum circuit. They are R_X, R_Y, R_Z and CNOT gate. The directed edges between two neighboring quantum gates indicate the temporal order of quantum gate execution on the quantum circuit. Since circuits with distinct number of qubits can be transformed into closed DAGs with only one start node and one end node, which allows us to employ the unified representation (nodes and edges) to symbolize quantum circuits with different number of qubits. The node feature vector contains two types information: gate type and node index, which will be used by GNN-based encoder.

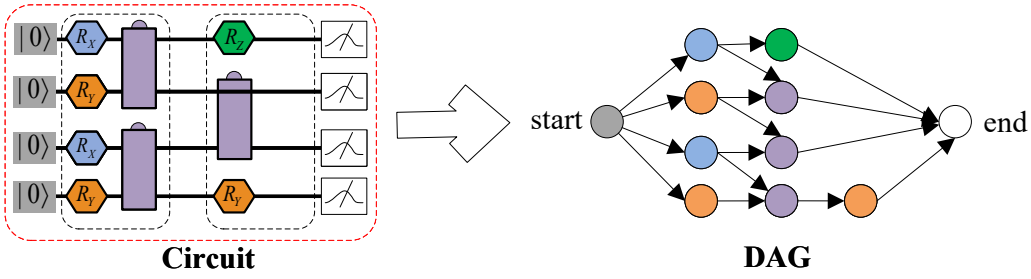


FIGURE 4.2. **An example of the circuit encoding scheme.**

GNN-based encoder. GNN-based encoder plays two roles in our algorithm. 1) It encodes the quantum gates of different types into vectors. 2) It can update the encoded vectors through message passing on the DAG. The workflow of the GNN-based encoder is shown in Fig. 4.3. The input of GNN-based encoder is the DAG of a circuit, $\mathcal{A}$. It contains the node features and edges of DAG. First, we will extract the gate type from the node features and encode it into a vector using Embedding layer (in this paper, we employ the nn.Embedding layer of PyTorch). As Embedding layer is trainable, this dynamic encoding method has better generalization than

one-hot encoding and can also capture the relationship between different quantum operations. The green dotted box in the Fig. 4.3 is designed to update embeddings. This module consists of a MLP and GRU, which are used to transmit messages between neighboring nodes. When the update is completed, we can get the final circuit embeddings, $\boldsymbol{e} \in \mathbb{R}^{k \times l}$. Where k refers to the number of parameterized quantum gates in QNNs, and l represents the length of vector that encoded from parameterized quantum gate. This corresponds to the function of Eq. (4.6). In general, you can choose MLP and GRU of any structure to form the message passing module. In this paper, MLP is a 4-layer neural network. It starts with the Fully Connected (FC) layer, followed by Rectified Linear Unit (Relu) layer [3]. Then another FC layer and Relu layer. And the input and output of this MLP have the same dimensions. The detailed architecture of MLP can be seen in Fig. 4.4. GRU is a normal GRU with input and output of the same dimensions. And this GRU shared across all nodes in a graph.

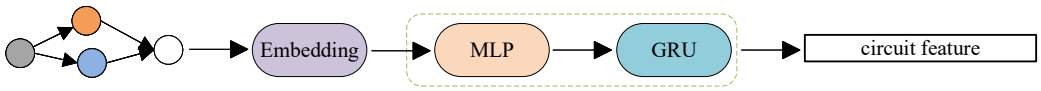


FIGURE 4.3. **The pipeline of the GNN-based encoder.** The block 'Embedding' represents a embedding layer that can transform the discrete node information to a continuous vector. The block 'MLP' is a multi-layer perceptron and the block 'GRU' refers to a Gated Recurrent Unit.

MLP-based decoder. The architecture of MLP-based decoder is shown in Fig. 4.5. In order to reduce the training time and improve the stability of the GHN, we first perform layer normalization [9] on the node embeddings obtained from the GNN-based encoder as the input of MLP-based decoder. The mathematical form of the layer normalization is shown below:

$$\tilde{\boldsymbol{x}} = \frac{\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}]}{\sqrt{\mathrm{Var}[\boldsymbol{x}] + \varepsilon}} * \gamma + \beta \tag{4.12}$$

where $\varepsilon > 0$ is a very small number. γ and β are learnable affine transform parameters. And the output of MLP-based decoder is the parameter related to the parameterized quantum gate. The role of this decoder corresponds to Eq. (4.7).

Training details. The training phase of QuGHN consists of two stages. The first stage is the parameter prediction stage, which only uses classical resources. While the second stage will conduct on quantum computer (in this paper, we only use classical computer to

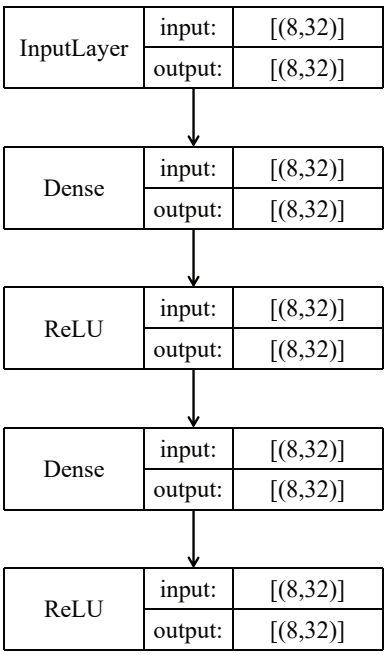


FIGURE 4.4. **The architecture of MLP in the encoder of QuGHN**. [8,32] corresponding to the number of parameterized quantum gates k, the dimension of the gate type vector l.

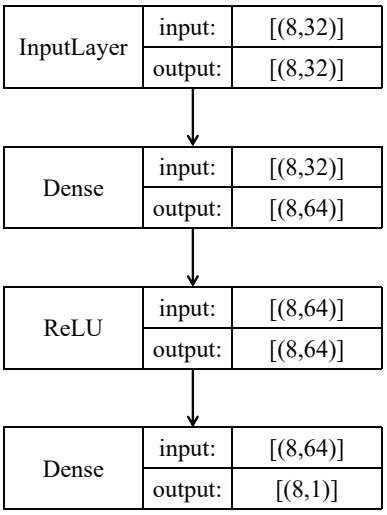


FIGURE 4.5. **The architecture of MLP-based decoder**. [8,32] corresponding to the number of parameterized quantum gates k, the dimension of the gate type vector l.

simulate the results of quantum computer). In the first stage, input is the DAGs of circuits $\mathcal{A}$. Then it passes through a parameter predictor consisting of the GNN-based encoder and the MLP-based decoder, and outputs the parameters $\boldsymbol{\theta}$. While in the second stage, we will input

parameters $\boldsymbol{\theta}$ and data $\boldsymbol{x}$ into the quantum circuits $\boldsymbol{A}$ to get the predicted ground state energy:

$$\hat{y}_i = \langle \psi_i | O(\boldsymbol{x}_i) | \psi_i \rangle \tag{4.13}$$

where $|\psi_i\rangle \equiv |\psi(\boldsymbol{\theta}, a_i)\rangle = U_{a_i}(\boldsymbol{\theta})|0\rangle^{\otimes N}$. And $U_{a_i}(\cdot)$ is the unitary of quantum circuit a_i. In fact, $\hat{y}_i = f(\boldsymbol{x}_i; a_i, GHN(\mathcal{A}_i; \boldsymbol{\phi}; \boldsymbol{\varphi}))$. So, in each training batch, we first obtain the predicted $\hat{y}$ from the quantum terminal, and then optimize the Eq. (4.3) on the classical computer.

For clarification, we summarize the pseudocode of QuGHN.

Algorithm 2: Pseudo code of QuGHN.

Input: $\boldsymbol{x} \in \mathbb{R}^{n \times d}$, $y \in \mathbb{R}^n$, and $\mathcal{A}$;
Output: $\boldsymbol{\phi}^*$ and $\boldsymbol{\varphi}^*$;

repeat
  Encoding the $\mathcal{A}$ to embeddings $\boldsymbol{e}$ based on the Eq. (4.6);
  Performing the layer normalization on the embeddings through the Eq. (4.12);
  Decoding the final $\boldsymbol{e}$ into predicted parameters $\boldsymbol{\theta}$ according to the Eq. (4.7);
  Conducting measurement on quantum computer to get the predicted labels $\hat{y}$ by Eq. (4.13);
  Updating the $\boldsymbol{\phi}$ and $\boldsymbol{\varphi}$ of the GHN according to Eq. (4.3);
until *The stopping criterion is satisfied*;

4.4 Numerical Results

In this chapter, we conduct extensive experiments to evaluate the performance of QuGHN across various tasks, diverse scenarios. Our primary objectives are to explore the effectiveness, robustness and versatility of QuGHN compared to prior advanced initialization methods, assess the effectiveness of QuGHN in addressing the barren plateaus in QNNs, and validate the convergence of parameters generated by QuGHN. All simulations are conducted using Python, utilizing the TensorCircuit [183], PyTorch [130], and the JAX library [19]. All experiments are run on AMD EPYC 7302 16-Core Processor (3.0GHz) with 188G memory (Ubuntu system).

4.4.1 Experimental Setting

In the section, we adopt two ansatzes, i.e., Hardware-Efficient Ansatz (HEA) [88] and Hamiltonian Variational Ansatz (HVA) [170], three diverse Hamiltonians as well as four comparison algorithms to conduct comprehensive evaluation of the performance of our algorithm. For completeness, here we provide the necessary backgrounds of these ansatzes, Hamiltonians and methods.

Ansatzes. To verify the performance of QuGHN, we mainly test it on two different ansatz, namely problem-agnostic ansatz HEA and problem-inspired ansatz HVA, which are commonly used in variational quantum algorithms (VQA) [28].

(1) **HEA.** The circuit of the HEA used in this work is:

$$U(\boldsymbol{\theta}) = \prod_{j=1}^{B} \hat{U}_j U_j(\theta_j) \tag{4.14}$$

where B is the number of blocks, $U_j(\theta_j)$ refers to the j-th block taking the form as $U_j = (\otimes_{i=1}^{N} R_a(\boldsymbol{\theta}_{ij}))$, $a \in \{X, Y, Z\}$, N is the number of qubits, R_a is a single-qubit rotation gate. While $\hat{U}_j$ refers to the entangled layer has the following form:

$$\hat{U}_j = \prod_{i=1}^{N-1} \mathrm{CNOT}_{i,i+1} \tag{4.15}$$

(2) **HVA.** The HVA uses operators in the Hamiltonian to generate an ansatz. Generally speaking, a given Hamiltonian can be represented as a linear combination of operators: $H = \sum_{k=1}^{K} H_k$, so the HVA has the form:

$$U(\boldsymbol{\theta}) = \prod_{j=1}^{B} \left(\prod_{k=1}^{K} e^{-i\theta_{j,k} H_k} \right) \tag{4.16}$$

Hamiltonians. For the purpose of exploring the universality of QuGHN, we conduct experiments on three different models, namely, transverse field Ising model (TFIM), Heisenberg model and transverse field cluster model (TFCM). Their Hamiltonians are given by the following formula:

(1) **Ising.**

$$H_{\text{Ising}} = -J_{\text{Ising}} \sum_{i}^{N-1} Z_i Z_{i+1} - J_x \sum_{i}^{N} X_i \tag{4.17}$$

where J_{Ising} refers to the spin-coupling strength in the z-axis, while J_x represents the relative strength of the transverse field along the x-axis.

(2) **Heisenberg.**

$$H_{\text{Heis}} = -J_{xy}(\sum_{i}^{N-1} X_i X_{i+1} + \sum_{i}^{N-1} Y_i Y_{i+1}) - J_z \sum_{i}^{N-1} Z_i Z_{i+1} \tag{4.18}$$

where J_{xy} refers to the spin-coupling strength in the x-axis and y-axis. And J_z represents the spin-coupling strength in the z-axis.

(3) **Cluster.**

$$H_{\text{cluster}} = -J_{\text{I}} \sum_{i}^{N-1} Z_{i-1} Z_i - \sum_{i}^{N} X_i - J_{\text{C}} \sum_{i=1}^{N-1} Z_{i-1} X_i Z_{i+1} \tag{4.19}$$

where J_{I} dominates the ferromagnetic ordered phase, and J_{C} dominates the topological phase.

Comparison algorithms. In order to better verify the performance of QuGHN, we select 4 parameter initialization methods for comparative experiments. These methods are FLIP [146], Meta-VQE [31], Learning to learn with quantum neural networks via classical neural networks (L2L) [166], as well as Random Initialization (RI) as the baselines. The basic principles of these algorithms are described below.

(1) **FLIP.** To our best knowledge, FLIP is the first method to achieve parameters initialization for arbitrarily-sized parametrized quantum circuits. It includes a fixed encoder and a trainable decoder. The core idea of FLIP is that it is to first encode the parameterized quantum gates into a vector of uniform length containing fixed information (i.e., gate type, the number of qubits, number of layers), and then use the idea of meta-learning to train a decoder to output the predicted parameters. Its loss function is:

$$\mathcal{L}(\boldsymbol{\phi}) = \int p(C) C_{\mathcal{T}}(\boldsymbol{\theta}_{\mathcal{T}}) dC \tag{4.20}$$

where $\boldsymbol{\phi}$ is parameters of the decoder, $p(C)$ denotes a distribution of a certain problem, $\boldsymbol{\theta}_{\mathcal{T}}$ refers to the predicted parameters by FLIP, that is $\boldsymbol{\theta}_{\mathcal{T}} = \mathcal{F}(\boldsymbol{e}; \boldsymbol{\phi})$. The $\mathcal{F}(\cdot; \boldsymbol{\phi})$ represents the decoder of FLIP, and $\boldsymbol{e} \in \mathbb{R}^{k \times l}$ denotes the embeddings of parameterized quantum gates, where k refers to the number of parameterized quantum gates in QNNs, and each parameterized quantum gate is encoded as a l-dimensional vector. And $C_{\mathcal{T}} \sim p(C)$. The aim of training FLIP is to obtain the optimal parameters $\boldsymbol{\phi}^*$ of the decoder. Then we can use this decoder to predict parameters of arbitrarily-sized quantum circuits according to the equation $\boldsymbol{\theta} = \mathcal{F}(\boldsymbol{e}; \boldsymbol{\phi}^*)$. And FLIP can be used for tasks including but not limited to state preparation [142], Quantum Approximate Optimization Algorithm (QAOA) [55], Fermi-Hubbard model [22].

(2) **Meta-VQE.** This approach is a variant of VQE, called meta-variational quantum eigensolver. And it focuses on learning the ground-state energy profile of a parameterized Hamiltonian. The main idea of Meta-VQE is that it encodes the training samples into a meta-quantum circuit, then trains this meta-quantum circuit, and finally uses the parameters of this meta-quantum circuit as the initialization parameters of other quantum circuits. Therefore, it can only initialize quantum circuits with the same number of parameters in meta-quantum circuit. The cost function of Meta-VQE is shown in the following:

$$\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\varphi}) = \sum\nolimits_{i=1}^{n} \langle \psi_i | H(\lambda_i) | \psi_i \rangle \tag{4.21}$$

where $\boldsymbol{\phi}$ and $\boldsymbol{\varphi}$ are the variational parameters of encoding circuits and processing circuits, respectively. λ denotes the parameters of parameterized Hamiltonian, n is the number of training set. While $|\psi_i\rangle \equiv |\psi(\lambda_i, \boldsymbol{\phi}, \boldsymbol{\varphi})\rangle = U(\boldsymbol{\phi}) S(\lambda_i, \boldsymbol{\varphi}) |0\rangle^{\otimes N}$. $U(\cdot)$ is the unitary of encoding circuit, and $S(\cdot)$ denotes the unitary of processing circuits. N is the number of qubit in quantum system. In test phase, we can use $\boldsymbol{\phi}^*$ and $\boldsymbol{\varphi}^*$ to initialize other QNNs that meet the requirements of Meta-VQE. This method can usually only be used for VQE tasks, such as XXZ model [115], quantum chemistry [98] and transmon simulation [87].

(3) **L2L.** This is the method that first to introduce meta-learning into the quantum field and use it for parameter initialization. The core idea of L2L is to train a classical recurrent neural network (RNN) to optimize the QNNs parameters. L2L mainly solves the following problems:

$$\mathcal{L}(\boldsymbol{\phi}) = \sum\nolimits_{i=1}^{n} \left(f(\boldsymbol{\theta}^t) - y_i\right)^2 \tag{4.22}$$

where $\boldsymbol{\phi}$ is the parameters of the RNN, n denotes the number of training samples, $\boldsymbol{\theta}^t$ is the predicted parameters of QNNs at time step t ($\boldsymbol{h}^t, \boldsymbol{\theta}^t = R(\boldsymbol{h}^{t-1}, \boldsymbol{\theta}^{t-1}; \boldsymbol{\phi})$, R refers to the RNN, as the decoder, h^t denotes the hidden state from the previous time step h^{t-1}, and $\boldsymbol{h}^0$ is an all-zero vector as the initial hidden state). While $f(\boldsymbol{\theta}^t) = \langle \hat{H} \rangle_{\boldsymbol{\theta}^t}$ represents the output of QNNs on the Hamiltonian operator $\hat{H}$. After the training is done, we can obtain the meta-parameters through $\boldsymbol{\theta}_m = R(\boldsymbol{h}^t, \boldsymbol{\theta}^t; \boldsymbol{\phi}^*)$. Since we obtain parameters are the meta-parameters, this method is only applicable to quantum circuits with the same number of parameters. This approach is able to initialise parameters on QAOA and VQE.

(4) **RI.** This is a general comparison algorithm as it initialise the parameters of QNNs based on random guesses, which makes it suitable for any scenario and very convenient. The principle of this algorithm is to randomly sampling parameters $\boldsymbol{\theta}$ ($\boldsymbol{\theta} \sim U(0, 2\pi)$) to initialise QNNs. Due to its efficiency and convenience, it has become an indispensable comparison algorithm.

4.4.1.1 Hyper-parameter Settings

In this section, we detail the hyper-parameter settings for data collection and model training. Our experiment involves three kinds of hyperparameters: coupling parameters, learning rate and the number of qubits N. Since the HEA circuit and the HVA circuit have different architectures, we will set different hyperparameters. In addition, we set the batch size to 16 and the number of training epochs to 10. The number of GNN layers is set to 1.

HEA settings. We perform numerical simulations with $J_{\text{Ising}} \in [-1, 0]$, $J_x \in [-1, 0]$ for TFIM, $J_{xy} \in [-1, 0]$, $J_z \in [-1, 0]$ for Heisenberg model and $J_{\text{I}} \in [-1, 0]$, $J_{\text{C}} \in [-1, 0]$ for

TFCM. Besides, the one layer of HEA circuit consists of a layer of sing-qubit rotation gate R_X, followed by a layer of CNOT gate, acting on all qubits.

Since our goal is to train a model that can be used to initialize the parameters of QNN with different Hamiltonians. In other words, our aim is to construct the relationship between the optimal parameters of QNN and the coupling parameters in quantum models. To this end, we first randomly sampled 100 different pairs of coupling parameters to generate Hamiltonians as training samples, and also randomly sampled another 100 different Hamiltonians as test samples. And all algorithms except random initialization will be trained for 10 epochs, and the learning rate of QuGHN, FLIP, Meta-VQE, L2L are 0.001, 0.001, 0.1, 0.1, respectively. In order to comprehensively evaluate the performance of QuGHN in quantum systems of different size, we set the number of qubits $N \in \{10, 15, 20, 25, 30, 40, 50, 60\}$. On the other hand, for the sake of saving time in collecting data and training models, we simulate and measure quantum circuits by Matrix Product States (MPS) and Matrix Product Operators (MPO) when $N > 15$. For quantum systems with more than 15 qubits, the ground state energy collected by training with variational quantum eigensolver (VQE) [134] for 200 epochs as the true ground state energy.

HVA settings. In this case, we set $J_{\text{Ising}} \in [0, 5]$, $J_x \in [0, 5]$ for TFIM, $J_{xy} \in [0, 3]$, $J_z \in [0, 3]$ for Heisenberg model and $J_{\text{I}} \in [0, 3]$, $J_{\text{C}} \in [-3, 0]$ for TFCM. And the learning rate of QuGHN, FLIP, Meta-VQE, L2L are set to 0.01, 0.001, 0.1, 0.1, respectively. On the other hand, we set the number of blocks $B = N$. In order to complete the experiment in a reasonable time, we only choose the number of qubits from $N \in \{6, 8, 10, 12\}$.

It is worth noting that we will defer the specific hyper-parameter settings to the corresponding experiments.

4.4.1.2 Performance metrics

In this section, we adopt two metrics, mean squared error (MSE) and average variance of gradient (AVG), to evaluate the performance of various algorithms. In addition, we also give a

detailed explanation of BP in QNNs to help us understand the practical difficulties in training QNNs.

- **MSE.** For a predictor, the MSE is defined as:

$$\mathrm{MSE} = \frac{1}{n}\sum\nolimits_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \tag{4.23}$$

  where n denotes the number of samples, Y_i is the label of the i-th sample, while $\hat{Y}_i$ refers to the predicted label by the predictor. And $\mathrm{MSE} \in [0, +\infty)$, which indicates the quality of a predictor. The smaller the value of MSE, the better the performance of the predictor.

- **AVG.**

$$\mathrm{AVG} = \frac{1}{n}\sum\nolimits_{i=1}^{n}\mathrm{Var}[\nabla_{\boldsymbol{\theta}} C_i(\boldsymbol{\theta})] \tag{4.24}$$

  where n refers to the number of parameters of QNNs, $\nabla_{\boldsymbol{\theta}} C_i(\boldsymbol{\theta}) = (\partial_{\boldsymbol{\theta}_1} C \partial_{\boldsymbol{\theta}_2} C \vdots)$ is the gradient of QNN. And $\mathrm{AVG} \geq 0$ holds true in any case. When evaluating the trainability of a QNN, the larger AVG is, the more trainable the QNN is and the less likely barren plateaus will occur.

Barren plateaus in QNNs. The barren plateaus phenomenon in QNNs is that the loss differences vanish exponentially with the size of the quantum system, which means that the loss landscape will gradually flatten, making optimization impossible. More precisely, vanishing loss differences can be formally defined as follows:

$$\mathrm{P}(|\mathcal{L}(\boldsymbol{\theta}) - \mathbb{E}[\mathcal{L}(\boldsymbol{\theta})]| \geq \delta) \in \mathcal{O}(\frac{1}{b^N}) \tag{4.25}$$

where $b > 1$, $\delta > 0$. N refers to the number of qubit in quantum system. $\mathbb{E}[\mathcal{L}(\boldsymbol{\theta})]$ denotes the expectation value over the parameter landscape. And if for some of μ, there is $\mathrm{Var}[\partial_\mu \mathcal{L}(\boldsymbol{\theta})] \in \mathcal{O}(\frac{1}{b^N})$, Eq. (4.25) holds true for these μ. In this situation, the partial derivative of the loss will concentration exponentially. To our knowledge, there are five factors that can lead to the emergence of barren plateaus in QNNs. They are related to the expressiveness of the QNNs [57], the entanglement of the input data [126], the locality of the observable [45], the choice of loss function [29, 164] and the presence of hardware noise [95, 139, 167]. When

the barren plateaus occurs, training QNNs will become infeasible as the differences between gradients of parameters of QNNs become exponentially small that making gradient-based optimization methods ineffective. Even using the gradient-free optimization methods is infeasible, as the loss at different points vanishing exponentially. This will make various QNN-based algorithms unable to be applied to practical tasks. Essentially, barren plateaus originated from a curse of dimensionality of the Hilbert space. So one should utilize shallow circuits [20], smart initialization [169], mid-circuit measurements [60] to reduce the unitary space explored to mitigate the barren plateaus phenomenon in QNNs.

4.4.2 Experimental Results

In the section, we present extensive numerical simulation results to demonstrate the effectiveness, robustness and versatility of QuGHN.

Effectiveness of QuGHN.

Performance on HEA backbone. We repeated each experiment three times for the purpose of offsetting the impact of randomness. And the detailed experimental results are listed in Table 4.1, Table 4.2 and Table 4.3. As can be seen from the above tables, the parameters predicted by our method all achieved the best performance. For instance, our method achieves perfect prediction on 10, 15, 20 qubits system in TFIM, whereas Meta-VQE and FLIP achieve 46.69 and 21.44 MSE on 20 qubits system, which are far from the perfect performance that our algorithm achieved. This observation continues across other scenarios. For the Heisenberg model, our method achieves MSE is less than 1 across all different number of qubits systems. Unlike the unstable performance of FLIP, our method demonstrates greater robustness. The same trend can be seen in TFCM. This is because FLIP only considers information with parameterized gates, while our method focuses on the entire circuit layout and the parameterized gates at the same time, so it has better performance and stability.

Performance on HVA backbone. All experiments were repeated three times to ensure the reliability of the experimental results. The detailed experimental results are listed in Table 4.4, Table 4.5 and Table 4.6. From the overall results, our method consistently delivers competitive

TABLE 4.1. The MSE of different parameter initialization methods on 100 different Hamiltonians for **HEA** circuits (**TFIM**).

Method \ Qubits	10	15	20	25	30	40	50	60
FLIP	7.5±9.30	22.20±30.74	21.44±22.92	6.02±8.30	1.10±1.51	169.84±234.56	1.49±1.54	391.96±553.73
Meta-VQE	5.5±3.82	24.55±5.24	46.69±6.30	69.73±3.37	103.35±11.47	179.73±5.60	343.44±21.65	478.90±34.87
L2L	66.01±0	168.92±3.31	287.85±4.33	458.54±5.44	487.24±260.16	1213.23±8.83	1911.49±11.10	2823.75±13.49
RI	29.41±1.13	71.34±1.40	125.23±2.39	191.57±2.09	282.35±10.72	500.77±2.96	791.54±23.29	1137.34±18.30
QuGHN	**0±0**	**0±0**	**0±0**	**0.07±0.07**	**0.04±0.02**	**0.47±0.52**	**0.44±0.38**	**1.67±2.30**

TABLE 4.2. The MSE of different parameter initialization methods on 100 different Hamiltonians for **HEA** circuits (**Heisenberg model**).

Method \ Qubits	10	15	20	25	30	40	50	60
FLIP	0.58±0.41	0.4±0.35	3.94±3.28	0.75±0.58	5.82±5.35	4.18±3.16	3.46±1.30	191.89±270.37
Meta-VQE	5.49±3.73	21.27±4.45	43.33±4.74	68.61±10.04	113.13±13.29	210.74±24.33	350.55±41.07	579.46±84.28
L2L	61.23±3.73	154.61±5.25	278.66±4.07	439.06±5.13	640.2±6.19	1161.61±8.33	1829.84±10.49	2694.13±25.39
RI	26.54±0.71	64.78±1.32	118.53±1.89	186.42±1.89	267.43±5.88	485.92±8.86	759.81±23.03	1093.48±0.82
QuGHN	**0±0**	**0.01±0**	**0±0**	**0.01±0.01**	**0.02±0.03**	**0.01±0.01**	**0.85±1.18**	**0.96±1.06**

TABLE 4.3. The MSE of different parameter initialization methods on 100 different Hamiltonians for **HEA** circuits (**TFCM**).

Method \ Qubits	10	15	20	25	30	40	50	60
FLIP	1.9±0.79	2.83±0.61	9.13±3.47	7.66±0.83	17.35±5.41	35.51±1.66	91.94±59.42	172.80±179.08
Meta-VQE	8.01±6.68	29.49±19.47	62.85±41.86	94.57±62.60	136.64±87.50	289.58±179.12	387.83±248.17	948.08±58.91
L2L	59.07±3.89	127.43±11.60	236.12±17.69	378.64±58.02	613.32±45.50	1084.23±9.83	1963.74±176.82	2443.32±157.56
RI	41.56±1.17	99.7±1.69	187.84±4.08	293.24±3.53	416.68±4.16	761.13±6.49	1186.24±5.19	1723.67±7.03
QuGHN	**1.05±0.07**	**2.57±0.43**	**4.29±0.60**	**6.19±0.06**	**9.65±0.90**	**17.45±2.69**	**24.35±1.45**	**39.29±8.01**

initialization parameters across all qubit numbers. For example, our method achieves 21.6 MSE for 6 qubits in TFIM, smaller than any other competitor. Although L2L achieves a relatively small MSE (28.8) for 6 qubits, the gap between L2L and our method becomes larger as the number of qubits increases, and the MSE value for 12 qubits is 1058.86 (L2L) vs 196.4 (QuGHN). We also observed similar phenomena in the Heisenberg model, such as the MSE obtained by L2L increases from 23.14 ($N = 6$) to 322.5 ($N = 12$), while our method only increases from 21.92 ($N = 6$) to 102.36 ($N = 12$) across all qubit numbers. This is enough to show the stability of our algorithm. On the other hand, although our method does not achieve the minimum MSE in TFCM, close to the result of L2L (2.22), we still achieved the minimum MSE on subsequent quantum system (8, 10, 12 qubits). Overall, our method demonstrates strong stability and superiority on different tasks.

Robustness of QuGHN.

TABLE 4.4. The MSE of different parameter initialization methods on 100 different Hamiltonians for **HVA** circuits (**TFIM**).

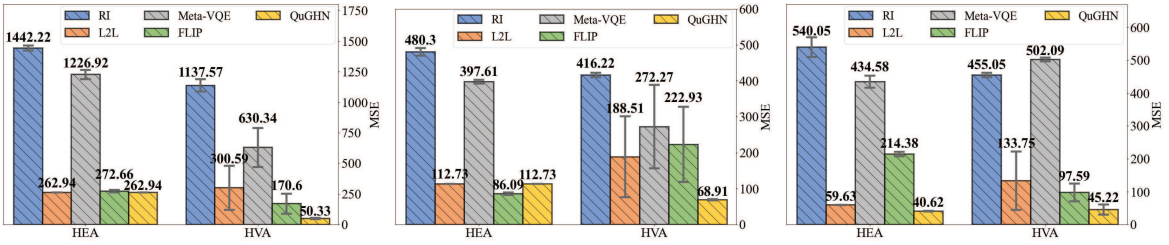
Method \ Qubits	6	8	10	12
FLIP	44.34±19.00	89.02±22.01	170.60±81.15	440.26±348.55
Meta-VQE	292.52±36.03	426.60±131.18	630.34±159.41	1014.72±21.86
L2L	28.8±1.15	103.79±48.68	300.59±180.17	1058.86±952.18
RI	392.74±26.98	749.64±20.73	1137.57±50.00	1716.18±33.89
QuGHN	**21.60±5.41**	**33.07±1.94**	**50.33±4.99**	**196.4±4.94**

TABLE 4.5. The MSE of different parameter initialization methods on 100 different Hamiltonians for **HVA** circuits (**Heisenberg model**).

Method \ Qubits	6	8	10	12
FLIP	46.17±21.54	98.42±28.68	222.93±104.64	567.93±18.66
Meta-VQE	57.51±9.73	119.83±41.01	272.27±116.03	375.92±189.79
L2L	23.14±1.77	56.85±10.63	188.51±112.71	322.50±116.88
RI	115.97±3.81	240.78±4.23	416.22±6.3	652.02±5.62
QuGHN	**21.92±0.14**	**48.18±6.59**	**68.91±1.90**	**102.36±4.87**

TABLE 4.6. The MSE of different parameter initialization methods on 100 different Hamiltonians for **HVA** circuits (**TFCM**).

Method \ Qubits	6	8	10	12
FLIP	20.98±7.93	44.17±29.98	97.59±26.81	207.22±92.21
Meta-VQE	144.48±18.79	291.92±11.20	502.09±5.81	751.69±4.54
L2L	**2.22±0.50**	184.00±98.97	133.75±88.85	630.22±348.18
RI	129.2±2.17	269.74±22.09	455.05±6.97	693.68±7.91
QuGHN	6.40±1.64	**18.93±2.60**	**45.22±15.37**	**61.85±19.67**

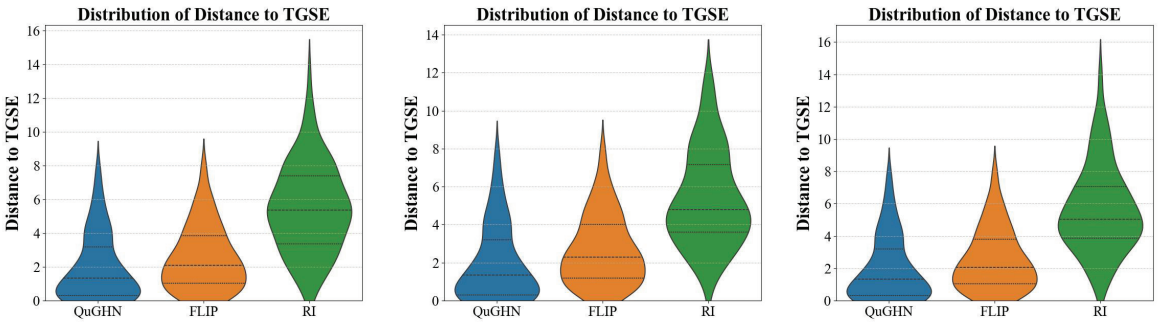


(a) Ising (b) Heisenberg (c) Cluster

FIGURE 4.6. Performance comparison on the three benchmark Hamiltonians with different ansatz on 10-qubit quantum system. The label of "HEA" represents the Hardware-efficient ansatz, while "HVA" refers to the Hamiltonian variational ansatz. The "MSE" means average results collected on 100 test Hamiltonians (lower is better). On the "HVA", QuGHN performs best.

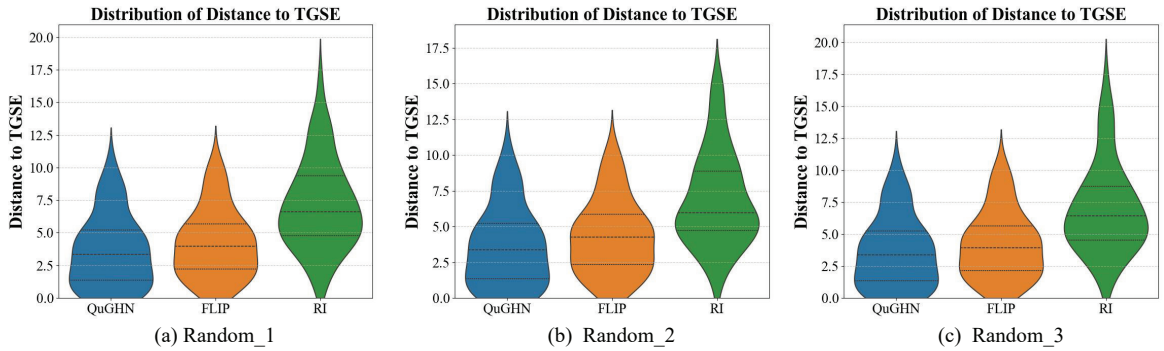
To further verify the robustness of our algorithm, we chose to test the performance of both HEA and HVA using the same setup on a 10-qubit system. And we collected exact ground state energy as labels to train and test the model. We also set $J_{\text{Ising}} \in [0, 5]$, $J_x \in [0, 5]$ for TFIM, $J_{xy} \in [0, 3]$, $J_z \in [0, 3]$ for Heisenberg model and $J_{\text{I}} \in [0, 3]$, $J_{\text{C}} \in [-3, 0]$ for TFCM. The results are presented in the Fig. 4.6. Our method consistently achieved competitive MSE across all ansatzes and all tasks. For example, our method achieves minimal MSE on Ising model, for HEA is 262.94 and for HVA is 50.33. The reason why the MSE on HEA is greater than that on HVA is that the layout of HEA and HVA are different. Another more important reason is that, in this case, the circuit constructed with HVA has 10 layers, while the circuit of HEA has only 1 layer, so the expression ability of HEA circuits is weaker than that of HVA circuits. In addition, we observe that the robustness of other algorithms is weaker than ours. For instance, on the Heisenberg model, the standard deviations of FLIP, L2L, and Meta-VQE are very large on HVA circuits, while our method has almost no fluctuation on both HVA and HEA circuits. This demonstrates the robustness of our method. On the other hand, as shown in Tables 4.1 to 4.6, the standard deviation of our method does not increase significantly with the increase of qubit number in quantum system, which once again proves the robustness of our method. In summary, our method comprehensively demonstrates its robustness and stability on different number of qubit, distinct ansatz, and diverse tasks.

Versatility of QuGHN.



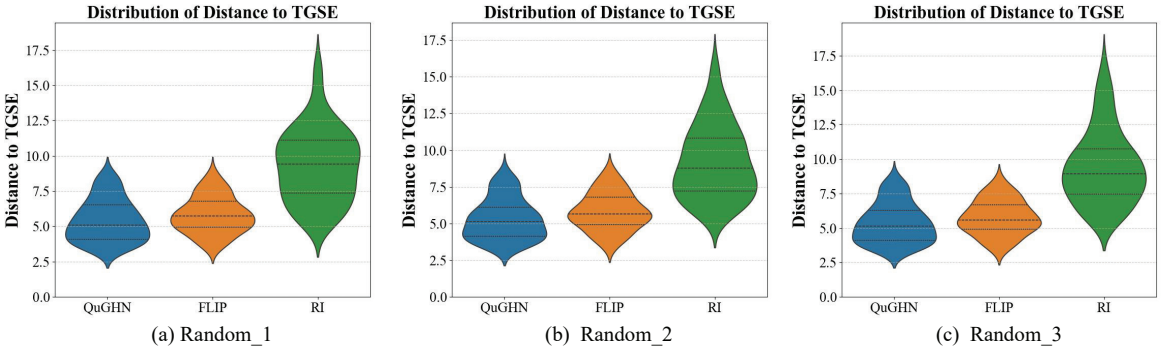
(a) Random_1 (b) Random_2 (c) Random_3

FIGURE 4.7. Performance of three random experiments on the **Ising model** with **HEA** circuits. The label of "TGSE" represents the true ground state energy. "Distance" refers to the absolute value of the difference between the predicted ground state energy and the true ground state energy, $d(E) = |E - E_0|$.



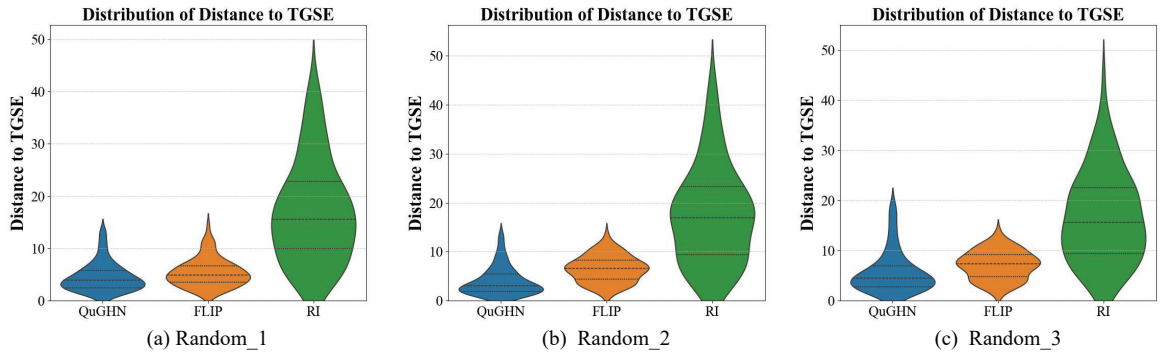
(a) Random_1 (b) Random_2 (c) Random_3

FIGURE 4.8. Performance of three random experiments on the **Heisenberg model** with **HEA** circuits.



(a) Random_1 (b) Random_2 (c) Random_3

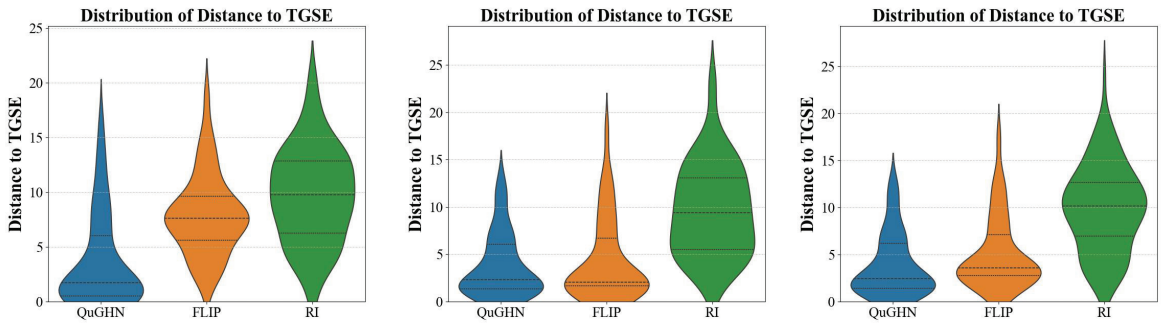
FIGURE 4.9. Performance of three random experiments on the **Cluster model** with **HEA** circuits.



(a) Random_1 (b) Random_2 (c) Random_3

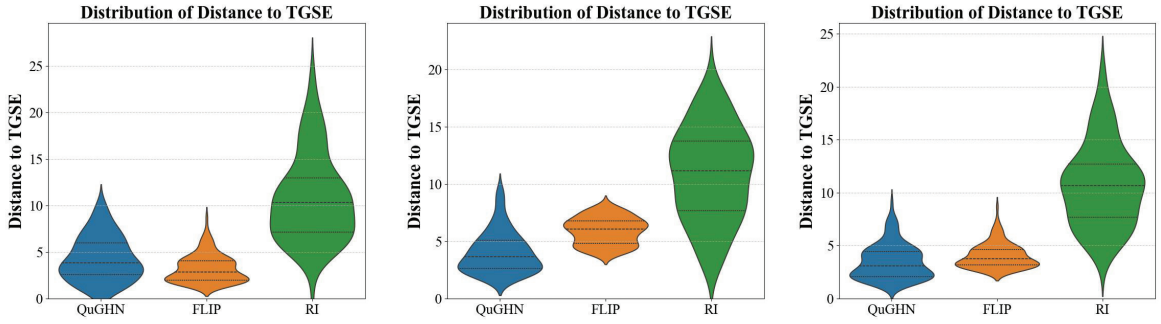
FIGURE 4.10. Performance of three random experiments on the **Ising model** with **HVA** circuits ($N = 6$).

In order to further verify the versatility of our algorithm. Here, we designed two experiments. One is to use circuits with different numbers of qubits to train and test RI, FLIP and our method on the circuit constructed by HEA (Meta-VQE and L2L can not adapt to these tasks). The other experiment is to use circuits with different numbers of layers (with different parameter



(a) Random_1 (b) Random_2 (c) Random_3

FIGURE 4.11. Performance of three random experiments on the **Heisenberg model** with **HVA** circuits ($N = 6$).



(a) Random_1 (b) Random_2 (c) Random_3

FIGURE 4.12. Performance of three random experiments on the **Cluster model** with **HVA** circuits ($N = 6$).

amounts) on the HVA circuits to train and test the performance of the above three algorithms. In the HEA experiment, the number of qubits will be randomly selected from [6, 8, 10], and a total of 120 Hamiltonians will be collected for training, and another 120 Hamiltonians will be randomly generated for testing. In the HVA experiment, we will uniformly conduct experiment on a quantum system with 6 qubits, where the number of layers of HVA will be randomly selected from [2, 4, 6], and 120 Hamiltonians will be used for training, and another 120 will be generated for testing. The experimental results are shown in the Fig. 4.7 to Fig. 4.12. From Fig. 4.7 to Fig. 4.9, we observed that QuGHN significantly outperformed other algorithms in all three times experiments on the Ising and Heisenberg models, and was slightly better than other algorithms on the cluster model. It is worth noting that RI performed much worse than our algorithm on all three tasks, which once again illustrates the necessity of designing suitable parameters for QNN initialization. For the results on HVA circuits, we only

observed a significant gap between QuGHN and other methods twice in three experiments on all tasks. For this reason, we put the detailed results in the Table 4.7 and Table 4.8. And we find that our algorithm achieves the lowest MSE across all tasks and all ansatzes, which demonstrates the superiority of QuGHN. In summary, our algorithm is very universal and effective.

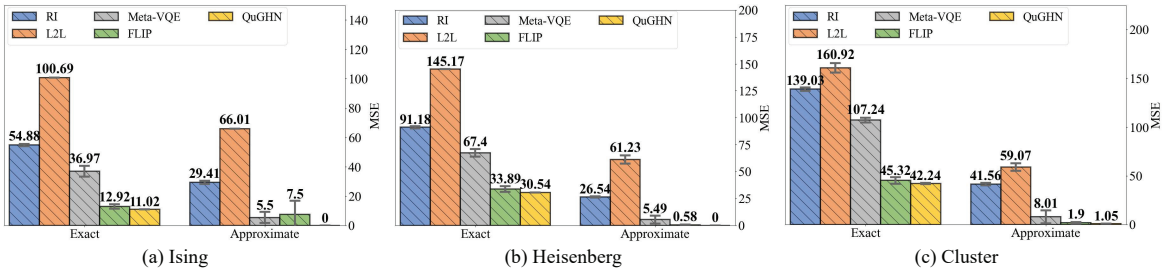
TABLE 4.7. The MSE of different parameter initialization methods on 120 different Hamiltonians with **HEA** circuits.

VQE task	Method	MSE
Ising	FLIP	10.13±0.27
	RI	37.07±1.67
	QuGHN	**7.7±0**
Heisenberg	FLIP	24.88±0.4
	RI	61.12±1.72
	QuGHN	**20.57±0**
Cluster	FLIP	35.17±0.29
	RI	93.92±2.29
	QuGHN	**30.79±0.6**

TABLE 4.8. The MSE of different parameter initialization methods on 120 different Hamiltonians with **HVA** circuits ($N = 6$).

VQE task	Method	MSE
Ising	FLIP	47.96±9.24
	RI	383.31±18.58
	QuGHN	**34.23±11.39**
Heisenberg	FLIP	52.03±17.80
	RI	117.08±2.24
	QuGHN	**27.96±2.77**
Cluster	FLIP	22.22±10.21
	RI	134.50±2.90
	QuGHN	**19.51±3.92**

Exact ground state energy vs Approximate ground state energy. As experiments on HEA involve large-scale quantum system, such as 50 qubits, collecting the exact ground state energy of the Hamiltonian is impractical. Therefore, we use VQE to obtain the approximate ground state energy of the Hamiltonian as labels for model training. To verify the effectiveness of these approximate labels, we compared the performance of training the model using exact ground state energy as labels with that of training using approximate labels in a 10-qubit quantum system. The experimental results are shown in Fig. 4.13. Overall, our algorithm



(a) Ising (b) Heisenberg (c) Cluster

FIGURE 4.13. Performance comparison on the three benchmark Hamiltonians with exact and approximate ground state energy obtained on 10-qubit quantum system. The label of "Exact" represents the exact energy ground state of the Hamiltonian obtained by matrix decomposition, while "Approximate" refers to the energy ground state of Hamiltonian obtained by VQE. The "MSE" means results collected on 100 test Hamiltonians (lower is better).

achieved the lowest MSE, demonstrating the best performance, regardless of whether using exact or approximate labels. Furthermore, we observed consistent performance rankings across different algorithms using both exact and approximate labels. For example, on both the Heisenberg model and Cluster model, the performance ranking from high to low is QuGHN $>$ FLIP $>$ Meat-VQE $>$ RI $>$ L2L. This validates that using approximate labels for model training and evaluation does not affect the final performance ranking, effectively reflecting the performance of each algorithm. On Ising model, we found that FLIP outperformed Meta-VQE on exact labels (12.92 vs 36.97), but underperformed Meta-VQE on approximate labels (7.5 vs 5.5). We speculate that this discrepancy is due to the instability of the FLIP, as it exhibits the largest standard deviations on approximate labels. In summary, using approximate labels to evaluate the performance of different algorithms is feasible, efficient and reasonable.

Mitigating barren plateaus in TFIM. To verify that our method can alleviate the barren plateaus during the training of QNNs, we used HEA as the backbone to construct quantum circuits and then conducted experiments on the one-dimensional transverse field Ising model. We gradually increased the number of qubits from 10 to 15, while setting the number of circuit layer to 4. We then initialized 100 random circuits using QuGHN and RI, and calculated the average variance gradient (AVG) of all circuits based on these initialized parameters. The detailed results is shown in Fig. 4.14. The AVG of QuGHN has an increasing trend as the number of qubit increases,

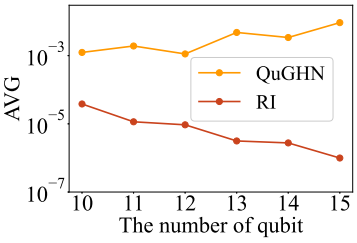


FIGURE 4.14. The "AVG" on 100 different circuits initialized by QuGHN and RI. "AVG" refers to the average variance of gradient.

while the AVG of RI shows a decreasing trend, which illustrates that our method can mitigate barren plateaus of QNNs.

Convergence of parameters generated by QuGHN. To explore the convergence of the

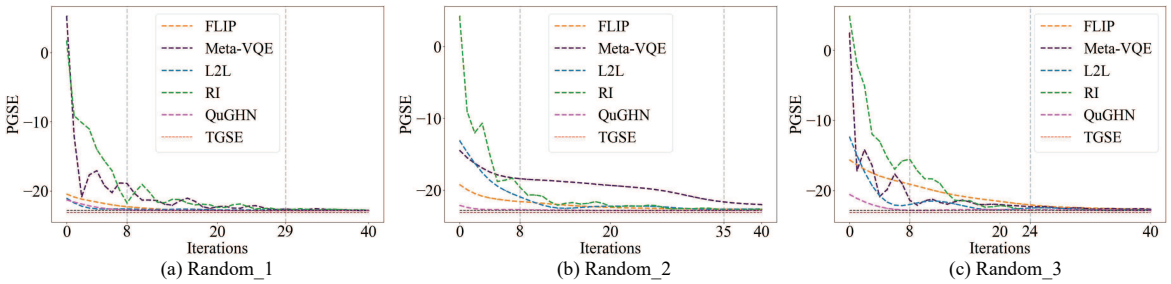


FIGURE 4.15. **Ising model** with 40 optimization steps of three random experiments on the **first** test Hamiltonian. The label of "PGSE" represents the predicted ground state energy. "TGSE" refers to the true ground state energy ($N = 10$).

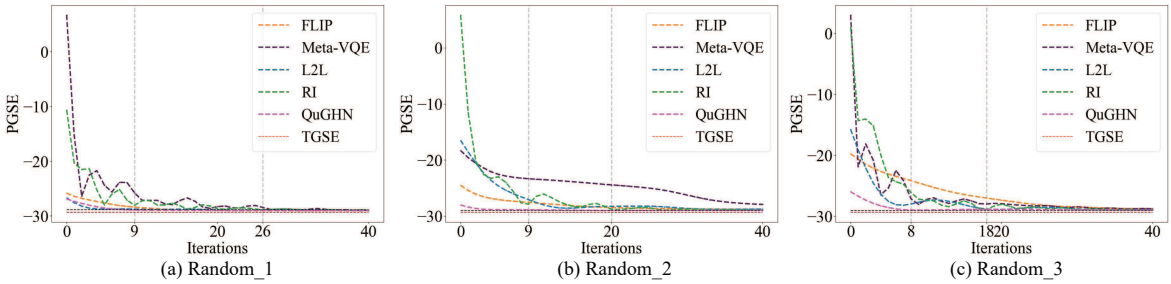


FIGURE 4.16. **Ising model** with 40 optimization steps of three random experiments on the **second** test Hamiltonian. The label of "PGSE" represents the predicted ground state energy. "TGSE" refers to the true ground state energy ($N = 10$).

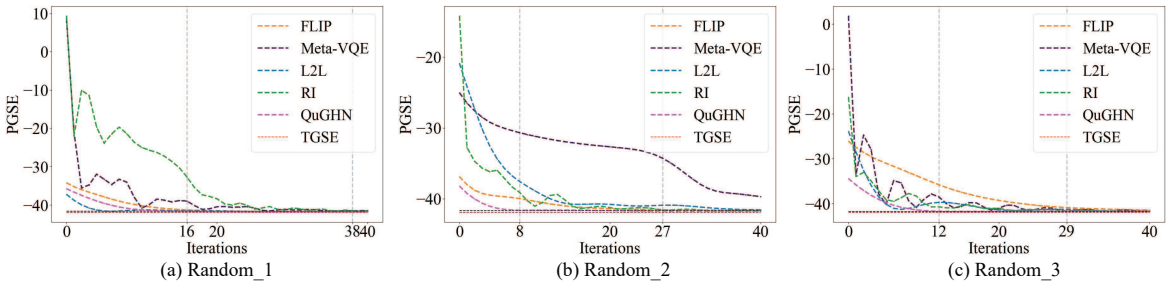
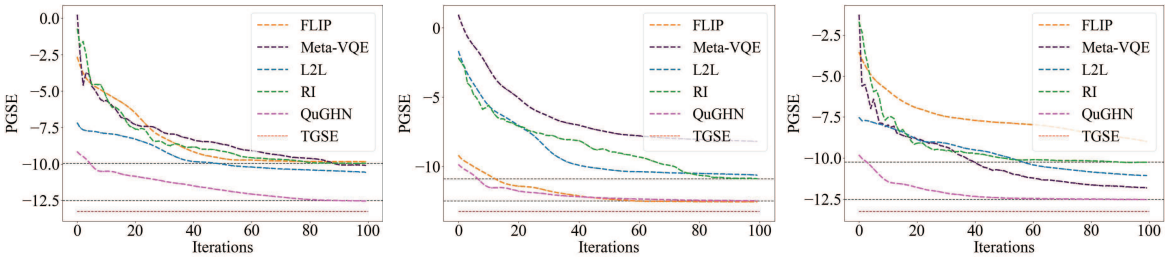
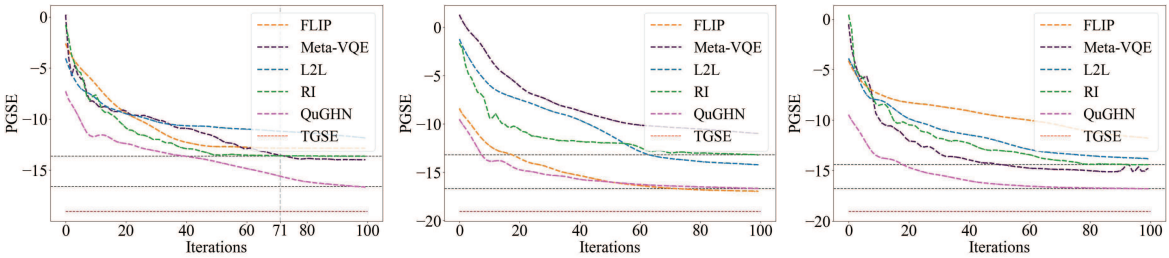


FIGURE 4.17. **Ising model** with 40 optimization steps of three random experiments on the **third** test Hamiltonian. The label of "PGSE" represents the predicted ground state energy. "TGSE" refers to the true ground state energy ($N = 10$).



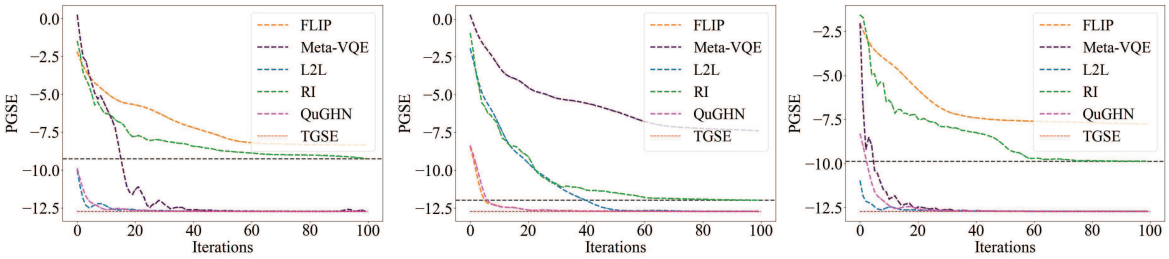
(a) Random_1 (b) Random_2 (c) Random_3

FIGURE 4.18. **Heisenberg model** with 100 optimization steps of three random experiments on the **first** test Hamiltonian. The label of "PGSE" represents the predicted ground state energy. "TGSE" refers to the true ground state energy ($N = 10$).



(a) Random_1 (b) Random_2 (c) Random_3

FIGURE 4.19. **Heisenberg model** with 100 optimization steps of three random experiments on the **second** test Hamiltonian. The label of "PGSE" represents the predicted ground state energy. "TGSE" refers to the true ground state energy ($N = 10$).



(a) Random_1 (b) Random_2 (c) Random_3

FIGURE 4.20. **Heisenberg model** with 100 optimization steps of three random experiments on the **third** test Hamiltonian. The label of "PGSE" represents the predicted ground state energy. "TGSE" refers to the true ground state energy ($N = 10$).

parameters predicted by QuGHN on the corresponding models, we conducted experiments on the circuits using the HVA backbone, where the quantum system was set to 10 qubits (because in experiments on the HEA circuits, the parameters predicted by the QuGHN have achieved MSE is close to 0 in a 10-qubit quantum system). We randomly selected three Hamiltonians for each task, then used QuGHN and other four comparison algorithms to obtain

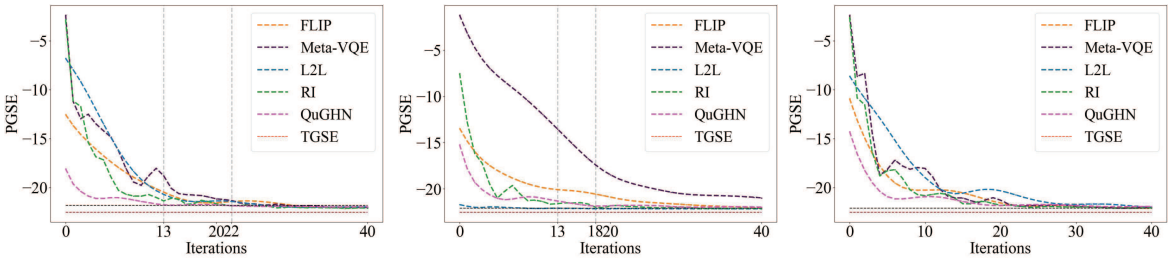


FIGURE 4.21. **Cluster model** with 40 optimization steps of three random experiments on the **first** test Hamiltonian. The label of "PGSE" represents the predicted ground state energy. "TGSE" refers to the true ground state energy ($N = 10$).

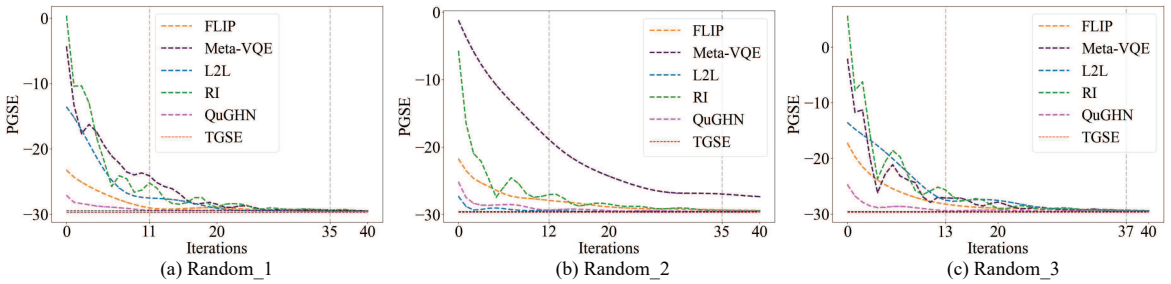


FIGURE 4.22. **Cluster model** with 40 optimization steps of three random experiments on the **second** test Hamiltonian. The label of "PGSE" represents the predicted ground state energy. "TGSE" refers to the true ground state energy ($N = 10$).

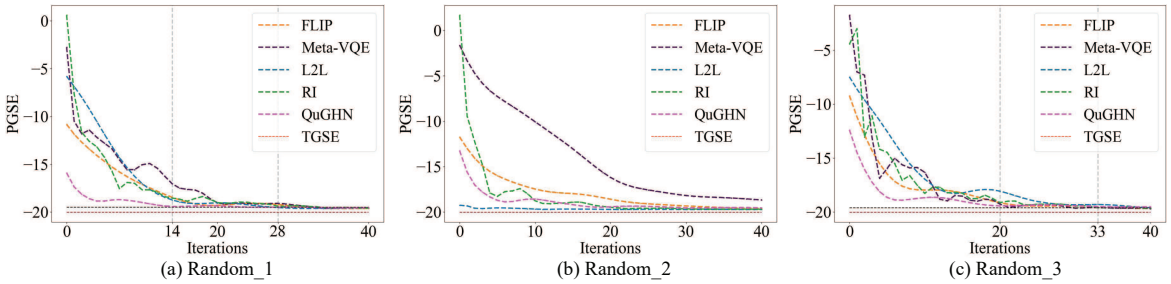


FIGURE 4.23. **Cluster model** with 40 optimization steps of three random experiments on the **third** test Hamiltonian. The label of "PGSE" represents the predicted ground state energy. "TGSE" refers to the true ground state energy ($N = 10$).

the parameters for model training (since we conducted three random experiments before, each method will get three initial parameters). After training for 100 epochs, we collected the loss values for analysis (in VQE, loss corresponds to the predicted ground state energy of the Hamiltonian). The detailed results are display on Fig. 4.15 to Fig. 4.23. Overall, the parameters initialized by our method have a faster convergence speed (the number of Adam

iterations to achieve a certain loss value), and can approach the better solutions. For example, in Fig. 4.15, the parameters generated by our method allow the model to converge in epochs 8, while the random initialized parameters only reach convergence in epochs 29, 35, and 24, respectively. This demonstrates that the parameters obtained by our method allow the model to converge about 3-4 times faster than the random initialization. From Fig. 4.16, Fig. 4.17, Fig. 4.21, Fig. 4.22 and Fig. 4.23, we also observe the same trend, that is, the parameters generated by QuGHN make the model reach convergence faster. From the experiments on Heisenberg model, we find that our method can find better solutions after training the model. For instance, in Fig. 4.18(a) and Fig. 4.18(c), the ground state energy that we predict after training 100 epochs is closer to the true ground state energy than that of predicted by model with random initialization (approximately -12.5 vs -10), and also better than other competitors. On the other hand, in Fig. 4.19(a), the parameters generated by random initialization can not be further optimized after epoch 71, while the parameters generated by our method can be continuously optimized, obtaining values closer to the true ground state energy. In addition, in Fig. 4.20, the parameters initialized by our method reach convergence at epoch 20, and the predicted ground state energy is also close to the true ground state energy. In summary, the parameters generated by our method can make the model converge faster and help the model find better solutions.

4.4.3 Discussion

In this section, we first summarize previous work on parameter initialization, then analyze the similarities and differences between these works and our approach. After that, we provide a detailed discussion of our algorithm. Finally, we summarized previous experiments to support our claims.

4.4.3.1 Related work

In order to better understand the work of this paper, we discuss the differences and connections between this paper and other related works. The previous works that are more related to

ours can be divided into three categories: distribution-based methods, meta-learning-based methods, and warm-start methods.

Distribution-based. This method mainly designs a clever sampling distribution, and then randomly sampling parameters from the designed distribution for QNNs initialization. Some methods take the input data into consideration to generate the sampling distribution [93], while others determine the distribution by the number of layers of the QNNs [182]. These methods can alleviate BP to some extent, but because more information is not used to design the sampling distribution, it is often difficult to obtain better performance. QuGHN is different from these works in that it uses more information such as the layout information of the QNNs and task information to generate the sampling distribution, hoping to output the optimal parameters for QNNs initialization. From the perspective of information theory, the distribution QuGHN generate has better robustness because this distribution has less uncertainty.

Meta-learning-based. The core idea of this method is to train a meta-learner to predict the parameters of QNN. This meta-learner hopes to incorporate all the knowledge about a task into a distribution, so that the corresponding knowledge can be directly retrieved when predicting the parameters. However, most of these methods cannot adapt to QNNs with different number of qubits or distinct layers, which greatly reduces their practicality [31, 166]. Although some methods can adaptively predict different QNNs with different numbers of parameters [146], in essence, this type of method is training a decoder to predict parameters with a fixed embedding scheme. In contrast to them, QuGHN trains both the encoder and the decoder to achieve the global optimal solution.

Warm-start. Warm-start method mainly borrows the idea of transfer learning, by training on a smaller size of QNN without BP and then transfer these parameters to a larger size of QNN. However, it has great limitations as it usually requires the base task and the target task to be related, so it currently can only focus on systems with translation invariance [120, 109]. In contrast, QuGHN can be used to initialize the parameters of QNN under any different tasks, so it is more practical.

4.4.3.2 Analysis of QuGHN advantages.

The above experiments demonstrate the effectiveness, robustness, and versatility of QuGHN. Here, we provide a more detailed analysis of the modules in our algorithm that contribute to these advantages. First, the effectiveness of this algorithm comes from using GNNs to pass messages across the DAG, capturing both global (layout) and local (quantum gates) information about the circuit. FLIP only utilizes local (quantum gates) information about the circuit, while Meta-VQE only uses global (meta-circuit) information, and L2L also only considers global (individual circuit) information. Random initialization does not use any valid information. Therefore, compared to these baseline algorithms that only introduce global or local information of circuits for model training, our method can achieve better results. On the other hand, the robustness of our algorithm originates from our dynamic encoding and decoding design, that is, the use of learnable encoders and decoders, which enables the parameter predictor to adapt to different scenarios. While the instability of FLIP is due to its fixed encoding scheme. Since Meta-VQE is too simple, it may be more susceptible to noisy data, reducing the robustness of the algorithm. And L2L fails to capture local information of the circuit structure, making it unable to accurately predict the parameters corresponding to the parameterized quantum gates, thus losing part of robustness. For the versatility of QuGHN, essentially, this is because QuGHN employs a scheme that can uniformly encode and decode quantum circuits. This scheme can encode each quantum gate as a vector of uniform dimension. And in the decoding stage, decoder is able to predict the parameters corresponding to each parameterized quantum gate individually, making it very universal. FLIP also utilizes encoding and decoding schemes, so it is suitable for different scenarios. Unlike QuGHN implements the predictions at the quantum gate level, Meta-VQE and L2L make predictions at the circuit level. Therefore, they are prone to less universal. In summary, we use GNN with DAG to improve performance, adopt a encoder and decoder solution to achieve versatility, and employ dynamic encoding and decoding to enhance the robustness.

Here, we will analyze the algorithm complexity of QuGHN to deepen our understanding of the QuGHN. According to the Eq. (4.5) in the main text, the meta taining cost of QuGHN is $\mathcal{O}(n \cdot m \cdot n_{epoch} \cdot T)$. Where n represents the number of training samples, m denotes the

number of quantum circuits, n_{epoch} is the number of training rounds, and T refers to the time required to execute a certain task once. T increases as the size of the quantum system increases. In addition, T is also the time complexity of random initialization.

4.4.3.3 Experimental Summary.

In this section, We first demonstrated the effectiveness of QuGHN on different ansatzes, quantum systems of differen scales, and diverse Hamiltonians. Then, we elaborated that QuGHN exhibits superior robustness compared to other comparative algorithms. Finally, we also verified the versatility of QuGHN on different settings. These findings strongly suggest that QuGHN is a general QNN parameter initialization method that combines efficiency, robustness, and versatility.

4.5 Summary

In this chapter, we propose QuGHN to automatically initialize parameters of QNNs, which greatly improves the practicality of VQAs. Different from the most of previous excellent methods can only be applied to fixed tasks or circuits, our proposal is adaptable to different tasks and parameter amounts. This enables us to better leverage VQAs to achieve potential quantum advantages on existing quantum devices.

CHAPTER 5

Conclusion and Outlook

Within this chapter, we outline the contributions of this thesis and provide potential avenues for future research.

5.1 Conclusion

In this thesis, we delve into two promising quantum machine learning (QML) algorithms, quantum kernels and variational quantum algorithms (VQAs).

Specifically, in Chapter 3, we devised a QuKerNet algorithm that used ideas similar to neural architecture search (NAS) to seek quantum kernels with better performance. This proposal mainly uses feature-selection technique to surmount the constraints imposed by near-term quantum machines, and to mitigate the vanishing similarity issue. Besides, it trained a neural predictor to evaluate the performance of quantum kernels. This search-and-evaluation mode enables QuKerNet to find quantum kernels with better performance. Compared with the general quantum kernels and the classical radial basis function kernels (RBFK), kernels searched by QuKerNet can show higher accuracy in various classification tasks. QuKerNet not only unlocks the potential of quantum kernels for enhancing real-world tasks but also highlights the substantial role of deep learning in advancing quantum machine learning.

In Chapter 4, we proposed the QuGHN architecture, which introduces the encoder-decoder framework into the VAQs. Motivation mainly comes from two aspects: on the one hand, most of the existing parameter initialization methods for quantum neural networks (QNNs) can only be used for QNNs with the same number of parameters or quantum systems with the same size,

or for certain tasks, which severely reduces the practicality of these methods; On the other hand, in generative models, models based on the encoder-decoder framework have achieved great success [43, 89, 157]. In addition, we introduced the graph neural network (GNN), which is commonly used in graph-structured data, to capture information from the circuit layout, thereby improving the performance of predicting parameters. Through numerical experiments on two types of ansatzes (problem-agnostic ansatz and problem-inspired ansatz), three tasks (Ising model, Heisenberg model and Cluster model) and various sizes of systems (the number of qubits varying from 6 to 60), we find that the parameters predicted by QuGHN exhibit enhanced robustness, convergence, and trainability. QuGHN not only paves the way for VQAs to achieve potential quantum advantage, but also makes it possible to apply VQAs on large-scale circuits.

In conclusion, this thesis conducts extensive research on quantum kernel design and parameter initialization, addressing common issues in quantum machine learning and introducing two novel strategies. In addition to improving the performance of the quantum kernel and the VQAs, the two proposed solutions demonstrate the indispensable role of deep learning in quantum machine learning, pointing the way for further development in this field.

5.2 Outlook

Quantum machine learning, a trending topic within quantum AI research, holds promising research and application prospects. Many scholars have contributed related applications and new research directions in recent years. This thesis also highlights various directions for future research, quantum kernel machine learning, and VQAs for science. It is worth noting that these directions are prospective ideas rather than extensions required to complete the current work.

Quantum kernel machine learning. We know that QuKerNet aims at machine learning tasks, so it is worth exploring how the kernels found by QuKerNet performs in clustering [186], dimensionality reduction [148], regression[158], generation [102] and other related tasks [37]. In addition, it is worth clarifying the performance of the kernels designed by QuKerNet under

imbalanced data [173]. Finally, it is valuable to investigate how the kernels generated by QuKerNet perform in the presence of adversarial samples [123].

VQAs for science. Variational algorithms [59] converts the original problem into a variational problem and solves it to obtain an approximate or optimal solution to the original problem. It is often used to solve complex problems in physics [128], engineering [141], economics [69], statistics [18] and other fields [67]. Therefore, with the help of parameters generated by QuGHN, we look forward to exploring the performance of VQAs in fields such as chemistry [12], biology [36], and high-energy physics [44]. Since it is very difficult for classical algorithms to deal with large-scale problems of these fields.

These identified research directions can deepen our understanding of quantum machine learning and give us the opportunity to see the boundaries of quantum kernel learning and the capabilities of VQAs. This will in turn promote the development of quantum machine learning, allowing us to obtain practical quantum advantages with the support of deep learning.

Bibliography

[1] Amira Abbas, David Sutter, Christa Zoufal, Aurélien Lucchi, Alessio Figalli and Stefan Woerner. 'The power of quantum neural networks'. In: *Nature Computational Science* 1.6 (2021), pp. 403–409.

[2] Hervé Abdi and Lynne J Williams. 'Principal component analysis'. In: *Wiley interdisciplinary reviews: computational statistics* 2.4 (2010), pp. 433–459.

[3] Abien Fred Agarap. 'Deep learning using rectified linear units (relu)'. In: *arXiv preprint arXiv:1803.08375* (2018).

[4] Hao Ai and Yu-xi Liu. 'Scalable Parameter Design for Superconducting Quantum Circuits with Graph Neural Networks'. In: *Physical Review Letters* 135.4 (2025), p. 040601.

[5] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta and Masanori Koyama. 'Optuna: A next-generation hyperparameter optimization framework'. In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019, pp. 2623–2631.

[6] Shamminuj Aktar, Andreas Bärtschi, Abdel-Hameed A Badawy, Diane Oyen and Stephan Eidenbenz. 'Predicting expressibility of parameterized quantum circuits using graph neural network'. In: *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Vol. 2. IEEE. 2023, pp. 401–402.

[7] Sergio Altares-López, Angela Ribeiro and Juan José García-Ripoll. 'Automatic design of quantum feature maps'. In: *Quantum Science and Technology* 6.4 (2021), p. 045015.

[8] David Amaro, Carlo Modica, Matthias Rosenkranz, Mattia Fiorentini, Marcello Benedetti and Michael Lubasch. 'Filtering variational quantum algorithms for combinatorial optimization'. In: *Quantum Science and Technology* 7.1 (2022), p. 015021.

[9] Jimmy Lei Ba, Jamie Ryan Kiros and Geoffrey E Hinton. 'Layer normalization'. In: *arXiv preprint arXiv:1607.06450* (2016).

[10] Ryan Babbush, Jarrod R McClean, Michael Newman, Craig Gidney, Sergio Boixo and Hartmut Neven. 'Focus beyond quadratic speedups for error-corrected quantum advantage'. In: *PRX Quantum* 2.1 (2021), p. 010103.

[11] Marian Stewart Bartlett, Gwen Littlewort, Ian Fasel and Javier R Movellan. 'Real Time Face Detection and Facial Expression Recognition: Development and Applications to Human Computer Interaction.' In: *2003 Conference on computer vision and pattern recognition workshop*. Vol. 5. IEEE. 2003, pp. 53–53.

[12] Bela Bauer, Sergey Bravyi, Mario Motta and Garnet Kin-Lic Chan. 'Quantum algorithms for quantum chemistry and quantum materials science'. In: *Chemical reviews* 120.22 (2020), pp. 12685–12717.

[13] Marcello Benedetti, Erika Lloyd, Stefan Sack and Mattia Fiorentini. 'Parameterized quantum circuits as machine learning models'. In: *Quantum science and technology* 4.4 (2019), p. 043001.

[14] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, Shahnawaz Ahmed, Vishnu Ajith, M Sohaib Alam, Guillermo Alonso-Linaje, B AkashNarayanan, Ali Asadi et al. 'Pennylane: Automatic differentiation of hybrid quantum-classical computations'. In: *arXiv preprint arXiv:1811.04968* (2018).

[15] Kishor Bharti, Alba Cervera-Lierta, Thi Ha Kyaw, Tobias Haug, Sumner Alperin-Lea, Abhinav Anand, Matthias Degroote, Hermanni Heimonen, Jakob S Kottmann, Tim Menke et al. 'Noisy intermediate-scale quantum algorithms'. In: *Reviews of Modern Physics* 94.1 (2022), p. 015004.

[16] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe and Seth Lloyd. 'Quantum machine learning'. In: *Nature* 549.7671 (2017), pp. 195–202.

[17] Carsten Blank, Daniel K Park, June-Koo Kevin Rhee and Francesco Petruccione. 'Quantum classifier with tailored quantum kernel'. In: *npj Quantum Information* 6.1 (2020), p. 41.

[18] David M Blei, Alp Kucukelbir and Jon D McAuliffe. ‘Variational inference: A review for statisticians’. In: *Journal of the American statistical Association* 112.518 (2017), pp. 859–877.

[19] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne et al. ‘JAX: Composable Transformations of Python+ NumPy Programs (v0. 2.5)’. In: *Software available from https://github. com/google/jax* (2018).

[20] Sergey Bravyi, David Gosset and Robert König. ‘Quantum advantage with shallow circuits’. In: *Science* 362.6412 (2018), pp. 308–311.

[21] Martin Dietrich Buhmann. ‘Radial basis functions’. In: *Acta numerica* 9 (2000), pp. 1–38.

[22] Evgeni Burovski, Nikolay Prokof’ev, Boris Svistunov and Matthias Troyer. ‘The Fermi–Hubbard model at unitarity’. In: *New Journal of Physics* 8.8 (2006), p. 153.

[23] Zhenyu Cai, Ryan Babbush, Simon C Benjamin, Suguru Endo, William J Huggins, Ying Li, Jarrod R McClean and Thomas E O’Brien. ‘Quantum error mitigation’. In: *Reviews of Modern Physics* 95.4 (2023), p. 045005.

[24] Jorge E Camargo and Fabio A González. ‘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, November 15-18, 2009. Proceedings 14*. Springer. 2009, pp. 545–552.

[25] Abdulkadir Canatar, Evan Peters, Cengiz Pehlevan, Stefan M Wild and Ruslan Shaydulin. ‘Bandwidth Enables Generalization in Quantum Kernel Models’. In: *Transactions on Machine Learning Research* (2023).

[26] Dong-Sheng Cao, Yi-Zeng Liang, Qing-Song Xu, Qian-Nan Hu, Liang-Xiao Zhang and Guang-Hui Fu. ‘Exploring nonlinear relationships in chemical data using kernel-based methods’. In: *Chemometrics and Intelligent Laboratory Systems* 107.1 (2011), pp. 106–115.

[27] Rich Caruana, Steve Lawrence and C Giles. 'Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping'. In: *Advances in neural information processing systems* 13 (2000).

[28] Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio et al. 'Variational quantum algorithms'. In: *Nature Reviews Physics* 3.9 (2021), pp. 625–644.

[29] Marco Cerezo, Akira Sone, Tyler Volkoff, Lukasz Cincio and Patrick J Coles. 'Cost function dependent barren plateaus in shallow parametrized quantum circuits'. In: *Nature communications* 12.1 (2021), p. 1791.

[30] Marco Cerezo, Guillaume Verdon, Hsin-Yuan Huang, Lukasz Cincio and Patrick J Coles. 'Challenges and opportunities in quantum machine learning'. In: *Nature computational science* 2.9 (2022), pp. 567–576.

[31] Alba Cervera-Lierta, Jakob S Kottmann and Alán Aspuru-Guzik. 'Meta-variational quantum eigensolver: Learning energy profiles of parameterized hamiltonians for quantum simulation'. In: *PRX Quantum* 2.2 (2021), p. 020329.

[32] Olivier Chapelle, Patrick Haffner and Vladimir N Vapnik. 'Support vector machines for histogram-based image classification'. In: *IEEE transactions on Neural Networks* 10.5 (1999), pp. 1055–1064.

[33] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk and Yoshua Bengio. 'Learning phrase representations using RNN encoder-decoder for statistical machine translation'. In: *arXiv preprint arXiv:1406.1078* (2014).

[34] Alessia Ciacco, Francesca Guerriero and Giusy Macrina. 'Review of quantum algorithms for medicine, finance and logistics'. In: *Soft Computing* 29.4 (2025), pp. 2129–2170.

[35] Juan Ignacio Cirac, Maciej Lewenstein, Klaus Mølmer and Peter Zoller. 'Quantum superposition states of Bose-Einstein condensates'. In: *Physical Review A* 57.2 (1998), p. 1208.

[36] Benjamin A Cordier, Nicolas PD Sawaya, Gian Giacomo Guerreschi and Shannon K McWeeney. 'Biology and medicine in the landscape of quantum advantages'. In: *Journal of the Royal Society Interface* 19.196 (2022), p. 20220541.

[37] Paul Covington, Jay Adams and Emre Sargin. 'Deep neural networks for youtube recommendations'. In: *Proceedings of the 10th ACM conference on recommender systems*. 2016, pp. 191–198.

[38] N. Cristianini, J. Shawe-Taylor, A. Elisseeff and J. S. Kandola. 'On Kernel-Target Alignment'. In: *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*. 2001.

[39] Shunsuke Daimon and Yu-ichiro Matsushita. 'Quantum circuit generation for amplitude encoding using a transformer decoder'. In: *Physical Review Applied* 22.4 (2024), p. L041001.

[40] Andrea Dal Pozzolo, Giacomo Boracchi, Olivier Caelen, Cesare Alippi and Gianluca Bontempi. 'Credit card fraud detection: a realistic modeling and a novel learning strategy'. In: *IEEE transactions on neural networks and learning systems* 29.8 (2017), pp. 3784–3797.

[41] Pierre-Luc Dallaire-Demers, Jonathan Romero, Libor Veis, Sukin Sim and Alán Aspuru-Guzik. 'Low-depth circuit ansatz for preparing correlated fermionic states on a quantum computer'. In: *Quantum Science and Technology* 4.4 (2019), p. 045005.

[42] Luca Dellantonio. 'Towards simulating 2D effects in lattice gauge theories on a quantum computer'. In: *APS March Meeting Abstracts*. Vol. 2021. 2021, pp. M31–009.

[43] Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova. 'Bert: Pre-training of deep bidirectional transformers for language understanding'. In: *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*. 2019, pp. 4171–4186.

[44] Alberto Di Meglio, Karl Jansen, Ivano Tavernelli, Constantia Alexandrou, Srinivasan Arunachalam, Christian W Bauer, Kerstin Borras, Stefano Carrazza, Arianna Crippa,

Vincent Croft et al. ‘Quantum computing for high-energy physics: State of the art and challenges’. In: *PRx quantum* 5.3 (2024), p. 037001.

[45] NL Diaz, Diego García-Martín, Sujay Kazi, Martin Larocca and M Cerezo. ‘Showcasing a barren plateau theory beyond the dynamical lie algebra’. In: *arXiv preprint arXiv:2310.11505* (2023).

[46] Jean C Digitale, Jeffrey N Martin and Medellena Maria Glymour. ‘Tutorial on directed acyclic graphs’. In: *Journal of Clinical Epidemiology* 142 (2022), pp. 264–267.

[47] Yuxuan Du, Min-Hsiu Hsieh, Tongliang Liu, Shan You and Dacheng Tao. ‘Learnability of Quantum Neural Networks’. In: *PRX Quantum* 2.4 (2021), p. 040337.

[48] Yuxuan Du, Min-Hsiu Hsieh, Tongliang Liu, Shan You and Dacheng Tao. ‘Quantum differentially private sparse regression learning’. In: *IEEE Transactions on Information Theory* 68.8 (2022), pp. 5217–5233.

[49] Yuxuan Du, Tao Huang, Shan You, Min-Hsiu Hsieh and Dacheng Tao. ‘Quantum circuit architecture search for variational quantum algorithms’. In: *npj Quantum Information* 8.1 (2022), p. 62.

[50] Yuxuan Du, Tongliang Liu, Yinan Li, Runyao Duan and Dacheng Tao. ‘Quantum divide-and-conquer anchoring for separable non-negative matrix factorization’. In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 2018, pp. 2093–2099.

[51] Yuxuan Du, Yibo Yang, Dacheng Tao and Min-Hsiu Hsieh. ‘Demystify Problem-Dependent Power of Quantum Neural Networks on Multi-Class Classification’. In: *arXiv preprint arXiv:2301.01597* (2022).

[52] Trong Duong, Sang T Truong, Minh Tam, Bao Bach, Ju-Young Ryu and June-Koo Kevin Rhee. ‘Quantum neural architecture search with quantum circuits metric and bayesian optimization’. In: *arXiv preprint arXiv:2206.14115* (2022).

[53] Daniel J Egger, Claudio Gambella, Jakub Marecek, Scott McFaddin, Martin Mevissen, Rudy Raymond, Andrea Simonetto, Stefan Woerner and Elena Yndurain. ‘Quantum computing for finance: State-of-the-art and future prospects’. In: *IEEE Transactions on Quantum Engineering* 1 (2020), pp. 1–24.

[54] Suguru Endo, Iori Kurata and Yuya O Nakagawa. 'Calculation of the Green's function on near-term quantum computers'. In: *Physical Review Research* 2.3 (2020), p. 033281.

[55] Edward Farhi, Jeffrey Goldstone and Sam Gutmann. 'A quantum approximate optimization algorithm'. In: *arXiv preprint arXiv:1411.4028* (2014).

[56] Richard P Feynman. 'Simulating physics with computers'. In: *Feynman and computation*. CRC Press, 2018, pp. 133–153.

[57] Lucas Friedrich and Jonas Maziero. 'Quantum neural network cost function concentration dependency on the parametrization expressivity'. In: *Scientific Reports* 13.1 (2023), p. 9978.

[58] Ya Gao and Shiliang Sun. 'An empirical evaluation of linear and nonlinear kernels for text classification using support vector machines'. In: *2010 seventh international conference on fuzzy systems and knowledge discovery*. Vol. 4. IEEE. 2010, pp. 1502–1505.

[59] Izrail Moiseevitch Gelfand, Richard A Silverman et al. *Calculus of variations*. Courier Corporation, 2000.

[60] Michelle Gelman. 'A survey of methods for mitigating barren plateaus for parameterized quantum circuits'. In: *arXiv preprint arXiv:2406.14285* (2024).

[61] Gian Gentinetta, David Sutter, Christa Zoufal, Bryce Fuller and Stefan Woerner. 'Quantum kernel alignment with stochastic gradient descent'. In: *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Vol. 1. IEEE. 2023, pp. 256–262.

[62] András Gilyén, Yuan Su, Guang Hao Low and Nathan Wiebe. 'Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics'. In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. 2019, pp. 193–204.

[63] Jennifer Glick. 'Covariant quantum kernels for data with group structure'. In: *Bulletin of the American Physical Society* 67 (2022).

[64] Ian Goodfellow, Yoshua Bengio and Aaron Courville. *Deep learning*. MIT press, 2016.

[65] Sydney Henry Gould. *Variational methods for eigenvalue problems: an introduction to the methods of Rayleigh, Ritz, Weinstein, and Aronszajn*. Courier Corporation, 2012.

[66] Élie Gouzien and Nicolas Sangouard. 'Factoring 2048-bit rsa integers in 177 days with 13 436 qubits and a multimode memory'. In: *Physical review letters* 127.14 (2021), p. 140503.

[67] Harper R Grimsley, Sophia E Economou, Edwin Barnes and Nicholas J Mayhall. 'An adaptive variational algorithm for exact molecular simulations on a quantum computer'. In: *Nature communications* 10.1 (2019), p. 3007.

[68] Stuart Hadfield, Zhihui Wang, Bryan O'gorman, Eleanor G Rieffel, Davide Venturelli and Rupak Biswas. 'From the quantum approximate optimization algorithm to a quantum alternating operator ansatz'. In: *Algorithms* 12.2 (2019), p. 34.

[69] George Hadley and Murray C Kemp. *Variational methods in economics*. Vol. 1. Elsevier, 2014.

[70] Aram W Harrow, Avinatan Hassidim and Seth Lloyd. 'Quantum algorithm for linear systems of equations'. In: *Physical review letters* 103.15 (2009), p. 150502.

[71] Aram W Harrow and Ashley Montanaro. 'Quantum computational supremacy'. In: *Nature* 549.7671 (2017), pp. 203–209.

[72] Tobias Haug, Chris N Self and Myungshik Kim. 'Quantum machine learning of large datasets using randomized measurements'. In: *Machine Learning: Science and Technology* (2021).

[73] Vojtěch Havlíček, Antonio D Córcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow and Jay M Gambetta. 'Supervised learning with quantum-enhanced feature spaces'. In: *Nature* 567.7747 (2019), pp. 209–212.

[74] Xiaofei He and Partha Niyogi. 'Locality preserving projections'. In: *Advances in neural information processing systems* 16 (2003).

[75] Zhimin He, Maijie Deng, Shenggen Zheng, Lvzhou Li and Haozhen Situ. 'Gsqas: graph self-supervised quantum architecture search'. In: *Physica A: Statistical Mechanics and its Applications* 630 (2023), p. 129286.

[76] Zhimin He, Xuefen Zhang, Chuangtao Chen, Zhiming Huang, Yan Zhou and Haozhen Situ. 'A GNN-based predictor for quantum architecture search'. In: *Quantum Information Processing* 22.2 (2023), p. 128.

[77] Thomas Hofmann, Bernhard Schölkopf and Alexander J Smola. 'Kernel methods in machine learning'. In: (2008).

[78] Zoë Holmes, Kunal Sharma, Marco Cerezo and Patrick J Coles. 'Connecting ansatz expressibility to gradient magnitudes and barren plateaus'. In: *PRX quantum* 3.1 (2022), p. 010313.

[79] Ryszard Horodecki, Paweł Horodecki, Michał Horodecki and Karol Horodecki. 'Quantum entanglement'. In: *Reviews of modern physics* 81.2 (2009), pp. 865–942.

[80] Hsin-Yuan Huang, Michael Broughton, Masoud Mohseni, Ryan Babbush, Sergio Boixo, Hartmut Neven and Jarrod R McClean. 'Power of data in quantum machine learning'. In: *Nature communications* 12.1 (2021), p. 2631.

[81] Yiming Huang, Xiao Yuan, Huiyuan Wang and Yuxuan Du. 'Coreset selection can accelerate quantum machine learning models with provable generalization'. In: *Physical Review Applied* 22.1 (2024), p. 014074.

[82] Zhuo Huang, Xiaobo Xia, Li Shen, Bo Han, Mingming Gong, Chen Gong and Tongliang Liu. 'Harnessing out-of-distribution examples via augmenting content and style'. In: *ICLR*. 2023.

[83] Thomas Hubregtsen, David Wierichs, Elies Gil-Fuster, Peter-Jan HS Derks, Paul K Faehrmann and Johannes Jakob Meyer. 'Training quantum embedding kernels on near-term quantum computers'. In: *Physical Review A* 106.4 (2022), p. 042431.

[84] Jonas Jäger and Roman V Krems. 'Universal expressiveness of variational quantum classifiers and quantum kernels for support vector machines'. In: *Nature Communications* 14.1 (2023), p. 576.

[85] Navdeep Jaitly and Geoffrey E Hinton. 'Vocal tract length perturbation (VTLP) improves speech recognition'. In: *Proc. ICML Workshop on Deep Learning for Audio, Speech and Language*. Vol. 117. 2013, p. 21.

[86] Hao Jiang, Wai-Ki Ching, Ka Fai Cedric Yiu and Yushan Qiu. 'Stationary Mahalanobis kernel SVM for credit risk evaluation'. In: *Applied Soft Computing* 71 (2018), pp. 407–417.

[87] Tyler Jones, Kaiah Steven, Xavier Poncini, Matthew Rose and Arkady Fedorov. 'Approximations in transmon simulation'. In: *Physical Review Applied* 16.5 (2021), p. 054039.

[88] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M Chow and Jay M Gambetta. 'Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets'. In: *nature* 549.7671 (2017), pp. 242–246.

[89] Diederik P Kingma and Max Welling. 'Auto-encoding variational bayes'. In: *arXiv preprint arXiv:1312.6114* (2013).

[90] Mario Köppen. 'The curse of dimensionality'. In: *5th online world conference on soft computing in industrial applications (WSC5)*. Vol. 1. 2000, pp. 4–8.

[91] Zoran Krunic, Frederik F Flöther, George Seegan, Nathan D Earnest-Noble and Omar Shehab. 'Quantum kernels for real-world predictions based on electronic health records'. In: *IEEE Transactions on Quantum Engineering* 3 (2022), pp. 1–11.

[92] Jonas Kübler, Simon Buchholz and Bernhard Schölkopf. 'The inductive bias of quantum kernels'. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 12661–12673.

[93] Ankit Kulshrestha and Ilya Safro. 'Beinit: Avoiding barren plateaus in variational quantum algorithms'. In: *2022 IEEE international conference on quantum computing and engineering (QCE)*. IEEE. 2022, pp. 197–203.

[94] Martin Larocca, Nathan Ju, Diego García-Martín, Patrick J Coles and Marco Cerezo. 'Theory of overparametrization in quantum neural networks'. In: *Nature Computational Science* 3.6 (2023), pp. 542–551.

[95] Martin Larocca, Supanut Thanasilp, Samson Wang, Kunal Sharma, Jacob Biamonte, Patrick J Coles, Lukasz Cincio, Jarrod R McClean, Zoë Holmes and Marco Cerezo. 'Barren plateaus in variational quantum computing'. In: *Nature Reviews Physics* (2025), pp. 1–16.

[96] Y. Lecun and L. Bottou. 'Gradient-based learning applied to document recognition'. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[97] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard and Lawrence D Jackel. 'Backpropagation applied to handwritten zip code recognition'. In: *Neural computation* 1.4 (1989), pp. 541–551.

[98] Joonho Lee, William J Huggins, Martin Head-Gordon and K Birgitta Whaley. 'Generalized unitary coupled cluster wave functions for quantum computation'. In: *Journal of chemical theory and computation* 15.1 (2018), pp. 311–324.

[99] Cong Lei, Yuxuan Du, Peng Mi, Jun Yu and Tongliang Liu. 'Neural auto-designer for enhanced quantum kernels'. In: *arXiv preprint arXiv:2401.11098* (2024).

[100] Cong Lei and Xiaofeng Zhu. 'Unsupervised feature selection via local structure learning and sparse learning'. In: *Multimedia Tools and Applications* 77 (2018), pp. 29605–29622.

[101] Lorenzo Leone, Salvatore FE Oliviero, Lukasz Cincio and Marco Cerezo. 'On the practical usefulness of the hardware efficient ansatz'. In: *Quantum* 8 (2024), p. 1395.

[102] Junde Li, Rasit O Topaloglu and Swaroop Ghosh. 'Quantum generative models for small molecule drug discovery'. In: *IEEE transactions on quantum engineering* 2 (2021), pp. 1–8.

[103] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P Trevino, Jiliang Tang and Huan Liu. 'Feature selection: A data perspective'. In: *ACM computing surveys (CSUR)* 50.6 (2017), pp. 1–45.

[104] Yangding Li, Cong Lei, Yue Fang, Rongyao Hu, Yonggang Li and Shichao Zhang. 'Unsupervised feature selection by combining subspace learning with feature self-representation'. In: *Pattern Recognition Letters* 109 (2018), pp. 35–43.

[105] Ying Li and Simon C Benjamin. 'Efficient variational quantum simulator incorporating active error minimization'. In: *Physical Review X* 7.2 (2017), p. 021050.

[106] Yujia Li, Daniel Tarlow, Marc Brockschmidt and Richard Zemel. 'Gated graph sequence neural networks'. In: *arXiv preprint arXiv:1511.05493* (2015).

[107] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng and Jun Zhou. 'A survey of convolutional neural networks: analysis, applications, and prospects'. In: *IEEE transactions on neural networks and learning systems* 33.12 (2021), pp. 6999–7019.

[108] Kehuan Linghu, Yang Qian, Ruixia Wang, Meng-Jun Hu, Zhiyuan Li, Xuegang Li, Huikai Xu, Jingning Zhang, Teng Ma, Peng Zhao et al. 'Quantum circuit architecture search on a superconducting processor'. In: *Entropy* 26.12 (2024), p. 1025.

[109] Huan-Yu Liu, Tai-Ping Sun, Yu-Chun Wu, Yong-Jian Han and Guo-Ping Guo. 'Mitigating barren plateaus with transfer-learning-inspired parameter initializations'. In: *New Journal of Physics* 25.1 (2023), p. 013039.

[110] Minzhao Liu, Junyu Liu, Rui Liu, Henry Makhanov, Danylo Lykov, Anuj Apte and Yuri Alexeev. 'Embedding learning in hybrid quantum-classical neural networks'. In: *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE. 2022, pp. 79–86.

[111] Yunchao Liu, Srinivasan Arunachalam and Kristan Temme. 'A rigorous and robust quantum speed-up in supervised machine learning'. In: *Nature Physics* 17.9 (2021), pp. 1013–1017.

[112] Seth Lloyd, Masoud Mohseni and Patrick Rebentrost. 'Quantum principal component analysis'. In: *Nature Physics* 10.9 (2014), pp. 631–633.

[113] Seth Lloyd, Maria Schuld, Aroosa Ijaz, Josh Izaac and Nathan Killoran. 'Quantum embeddings for machine learning'. In: *arXiv preprint arXiv:2001.03622* (2020).

[114] Xudong Lu, Kaisen Pan, Ge Yan, Jiaming Shan, Wenjie Wu and Junchi Yan. 'QAS-bench: rethinking quantum architecture search and a benchmark'. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 22880–22898.

[115] Salimeh Mahdavifar, Saeed Mahdavifar and R Jafari. 'Magnetic quantum correlations in the one-dimensional transverse-field XXZ model'. In: *Physical Review A* 96.5 (2017), p. 052303.

[116] Anil Kumar Mandle, Pranita Jain and Shailendra Kumar Shrivastava. 'Protein structure prediction using support vector machine'. In: *International Journal on Soft Computing* 3.1 (2012), p. 67.

[117] Sam McArdle, Suguru Endo, Alán Aspuru-Guzik, Simon C Benjamin and Xiao Yuan. 'Quantum computational chemistry'. In: *Reviews of Modern Physics* 92.1 (2020), p. 015003.

[118] Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush and Hartmut Neven. 'Barren plateaus in quantum neural network training landscapes'. In: *Nature communications* 9.1 (2018), p. 4812.

[119] Larry R Medsker, Lakhmi Jain et al. 'Recurrent neural networks'. In: *Design and applications* 5.64-67 (2001), p. 2.

[120] Antonio A Mele, Glen B Mbeng, Giuseppe E Santoro, Mario Collura and Pietro Torta. 'Avoiding barren plateaus via transferability of smooth solutions in a Hamiltonian variational ansatz'. In: *Physical Review A* 106.6 (2022), p. L060401.

[121] Chinmay Mishra, Shane Thompson, Raphael Pooser and George Siopsis. 'Quantum computation of an interacting fermionic model'. In: *Quantum Science and Technology* 5.3 (2020), p. 035010.

[122] Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa and Keisuke Fujii. 'Quantum circuit learning'. In: *Physical Review A* 98.3 (2018), p. 032309.

[123] Giuseppe Montalbano and Leonardo Banchi. 'Quantum adversarial learning for kernel methods'. In: *Quantum Machine Intelligence* 7.1 (2025), p. 15.

[124] Kouhei Nakaji and Naoki Yamamoto. 'Expressibility of the alternating layered ansatz for quantum computation'. In: *Quantum* 5 (2021), p. 434.

[125] Michael A Nielsen and Isaac Chuang. *Quantum computation and quantum information*. 2002.

[126] Carlos Ortiz Marrero, Mária Kieferová and Nathan Wiebe. 'Entanglement-induced barren plateaus'. In: *PRX quantum* 2.4 (2021), p. 040316.

[127] Mateusz Ostaszewski, Lea M Trenkwalder, Wojciech Masarczyk, Eleanor Scerri and Vedran Dunjko. 'Reinforcement learning for optimization of variational quantum circuit architectures'. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 18182–18194.

[128] Vidvuds Ozoliņš, Rongjie Lai, Russel Caflisch and Stanley Osher. 'Compressed modes for variational problems in mathematics and physics'. In: *Proceedings of the National Academy of Sciences* 110.46 (2013), pp. 18368–18373.

[129] Jae-eun Park, Brian Quanz, Stephen Wood, Heather Higgins and Ray Harishankar. 'Practical application improvement to Quantum SVM: theory to practice'. In: *Annual Conference on Neural Information Processing Systems*. 2020.

[130] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga et al. 'Pytorch: An imperative style, high-performance deep learning library'. In: *Advances in neural information processing systems* 32 (2019).

[131] Arti Patle and Deepak Singh Chouhan. 'SVM kernel functions for classification'. In: *2013 International conference on advances in technology and engineering (ICATE)*. IEEE. 2013, pp. 1–9.

[132] Rowan Pellow-Jarman, Anban Pillay, Ilya Sinayskiy and Francesco Petruccione. 'Hybrid genetic optimization for quantum feature map design'. In: *Quantum Machine Intelligence* 6.2 (2024), p. 45.

[133] H. Peng, F. Long and C. Ding. 'Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy'. In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 27.8 (2005), pp. 1226–1238.

[134] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik and Jeremy L O'brien. 'A variational eigenvalue solver on a photonic quantum processor'. In: *Nature communications* 5.1 (2014), p. 4213.

[135] John Preskill. 'Quantum computing 40 years later'. In: *Feynman Lectures on Computation*. CRC Press, 2023, pp. 193–244.

[136] John Preskill. 'Quantum computing in the NISQ era and beyond'. In: *Quantum* 2 (2018), p. 79.

[137] Yang Qian, Xinbiao Wang, Yuxuan Du, Xingyao Wu and Dacheng Tao. 'The dilemma of quantum neural networks'. In: *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[138] Ali Rad, Alireza Seif and Norbert M Linke. 'Surviving the barren plateau in variational quantum circuits with Bayesian learning initialization'. In: *arXiv preprint arXiv:2203.02464* (2022).

[139] Michael Ragone, Bojko N Bakalov, Frédéric Sauvage, Alexander F Kemper, Carlos Ortiz Marrero, Martín Larocca and M Cerezo. 'A Lie algebraic theory of barren plateaus for deep parameterized quantum circuits'. In: *Nature Communications* 15.1 (2024), p. 7172.

[140] Patrick Rebentrost, Masoud Mohseni and Seth Lloyd. 'Quantum support vector machine for big data classification'. In: *Physical review letters* 113.13 (2014), p. 130503.

[141] Karel Rektorys. *Variational methods in mathematics, science and engineering*. Springer Science & Business Media, 2012.

[142] KJ Resch, JS Lundeen and AM Steinberg. 'Quantum state preparation and conditional coherence'. In: *Physical review letters* 88.11 (2002), p. 113601.

[143] Rosenblatt and F. 'The perceptron: a probabilistic model for information storage and organization in the brain.' In: *Psychological Review* 65 (1958), pp. 386–408.

[144] Vedika Saravanan and Samah M Saeed. 'Noise adaptive quantum circuit mapping using reinforcement learning and graph neural network'. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 43.5 (2023), pp. 1374–1386.

[145] Yuki Sato, Hiroshi C Watanabe, Rudy Raymond, Ruho Kondo, Kaito Wada, Katsuhiro Endo, Michihiko Sugawara and Naoki Yamamoto. 'Variational quantum algorithm for generalized eigenvalue problems and its application to the finite-element method'. In: *Physical Review A* 108.2 (2023), p. 022429.

[146] Frederic Sauvage, Sukin Sim, Alexander A Kunitsa, William A Simon, Marta Mauri and Alejandro Perdomo-Ortiz. 'FLIP: A flexible initializer for arbitrarily-sized parametrized quantum circuits'. In: *arXiv preprint arXiv:2103.08572* (2021).

[147] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner and Gabriele Monfardini. 'The graph neural network model'. In: *IEEE transactions on neural networks* 20.1 (2008), pp. 61–80.

[148] Bernhard Schölkopf, Alexander Smola and Klaus-Robert Müller. 'Kernel principal component analysis'. In: *International conference on artificial neural networks*. Springer. 1997, pp. 583–588.

[149] Maria Schuld and Nathan Killoran. 'Quantum machine learning in feature hilbert spaces'. In: *Physical review letters* 122.4 (2019), p. 040504.

[150] Maria Schuld, Ilya Sinayskiy and Francesco Petruccione. 'An introduction to quantum machine learning'. In: *Contemporary Physics* 56.2 (2015), pp. 172–185.

[151] Abhay Shastry, Abhijith Jayakumar, Apoorva Patel and Chiranjib Bhattacharyya. 'Shot-frugal and Robust quantum kernel classifiers'. In: *arXiv preprint arXiv:2210.06971* (2022).

[152] Ruslan Shaydulin and Stefan M Wild. 'Importance of kernel bandwidth in quantum machine learning'. In: *Physical Review A* 106.4 (2022), p. 042407.

[153] Peter W Shor. 'Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer'. In: *SIAM review* 41.2 (1999), pp. 303–332.

[154] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever and Ruslan Salakhutdinov. 'Dropout: a simple way to prevent neural networks from overfitting'. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[155] Shengyang Sun, Guodong Zhang, Chaoqi Wang, Wenyuan Zeng, Jiaman Li and Roger Grosse. 'Differentiable compositional kernel learning for Gaussian processes'. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 4828–4837.

[156] Leo Sünkel, Darya Martyniuk, Denny Mattern, Johannes Jung and Adrian Paschke. 'GA4QCO: Genetic Algorithm for Quantum Circuit Optimization'. In: *arXiv preprint arXiv:2302.01303* (2023).

[157] Ilya Sutskever, Oriol Vinyals and Quoc V Le. 'Sequence to sequence learning with neural networks'. In: *Advances in neural information processing systems* 27 (2014).

[158] Hiroyuki Takeda, Sina Farsiu and Peyman Milanfar. 'Kernel regression for image processing and reconstruction'. In: *IEEE Transactions on image processing* 16.2 (2007), pp. 349–366.

[159] Supanut Thanasilp, Samson Wang, Marco Cerezo and Zoe Holmes. 'Exponential concentration and untrainability in quantum kernel methods'. In: *Bulletin of the American Physical Society* (2023).

[160] Supanut Thanasilp, Samson Wang, Nhat Anh Nghiem, Patrick Coles and Marco Cerezo. 'Subtleties in the trainability of quantum machine learning models'. In: *Quantum Machine Intelligence* 5.1 (2023), p. 21.

[161] Jinkai Tian, Xiaoyu Sun, Yuxuan Du, Shanshan Zhao, Qing Liu, Kaining Zhang, Wei Yi, Wanrong Huang, Chaoyue Wang, Xingyao Wu et al. 'Recent advances for quantum neural networks in generative learning'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).

[162] Elham Torabian and Roman V Krems. 'Compositional optimization of quantum circuits for quantum kernels of support vector machines'. In: *Physical Review Research* 5.1 (2023), p. 013211.

[163] Ivor W Tsang, James T Kwok, Pak-Ming Cheung and Nello Cristianini. 'Core vector machines: Fast SVM training on very large data sets.' In: *Journal of Machine Learning Research* 6.4 (2005).

[164] AV Uvarov and Jacob D Biamonte. 'On barren plateaus and cost function locality in variational quantum algorithms'. In: *Journal of Physics A: Mathematical and Theoretical* 54.24 (2021), p. 245301.

[165] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser and Illia Polosukhin. 'Attention is all you need'. In: *Advances in neural information processing systems* 30 (2017).

[166] Guillaume Verdon, Michael Broughton, Jarrod R McClean, Kevin J Sung, Ryan Babbush, Zhang Jiang, Hartmut Neven and Masoud Mohseni. 'Learning to learn with quantum neural networks via classical neural networks'. In: *arXiv preprint arXiv:1907.05415* (2019).

[167] Samson Wang, Enrico Fontana, Marco Cerezo, Kunal Sharma, Akira Sone, Lukasz Cincio and Patrick J Coles. 'Noise-induced barren plateaus in variational quantum algorithms'. In: *Nature communications* 12.1 (2021), p. 6961.

[168] Xinbiao Wang, Yuxuan Du, Yong Luo and Dacheng Tao. 'Towards understanding the power of quantum kernels in the NISQ era'. In: *Quantum* 5 (2021), p. 531.

[169] Yabo Wang, Bo Qi, Chris Ferrie and Daoyi Dong. 'Trainability enhancement of parameterized quantum circuits via reduced-domain parameter initialization'. In: *Physical Review Applied* 22.5 (2024), p. 054005.

[170] Dave Wecker, Matthew B Hastings and Matthias Troyer. 'Progress towards practical quantum variational algorithms'. In: *Physical Review A* 92.4 (2015), p. 042303.

[171] Alexander Weiße, Gerhard Wellein, Andreas Alvermann and Holger Fehske. 'The kernel polynomial method'. In: *Reviews of modern physics* 78.1 (2006), pp. 275–306.

[172] Roeland Wiersema, Cunlu Zhou, Yvette de Sereville, Juan Felipe Carrasquilla, Yong Baek Kim and Henry Yuen. 'Exploring entanglement and optimization within the hamiltonian variational ansatz'. In: *PRX quantum* 1.2 (2020), p. 020319.

[173] Gang Wu and Edward Y Chang. 'KBA: Kernel boundary alignment considering imbalanced data distribution'. In: *IEEE Transactions on knowledge and data engineering* 17.6 (2005), pp. 786–795.

[174] Sau Lan Wu, Shaojun Sun, Wen Guan, Chen Zhou, Jay Chan, Chi Lung Cheng, Tuan Pham, Yan Qian, Alex Zeng Wang, Rui Zhang et al. 'Application of quantum machine learning using the quantum kernel algorithm on high energy physics analysis at the LHC'. In: *Physical Review Research* 3.3 (2021), p. 033221.

[175] Wenjie Wu, Ge Yan, Xudong Lu, Kaisen Pan and Junchi Yan. 'QuantumDARTS: differentiable quantum architecture search for variational quantum algorithms'. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 37745–37764.

[176] Yusen Wu, Bujiao Wu, Jingbo Wang and Xiao Yuan. 'Quantum Phase Recognition via Quantum Kernel Methods'. In: *Quantum* 7 (2023), p. 981.

[177] Xuchen You, Shouvanik Chakrabarti and Xiaodi Wu. 'A convergence theory for over-parameterized variational quantum eigensolvers'. In: *arXiv preprint arXiv:2205.12481* (2022).

[178] Changan Yuan, Zhi Zhong, Cong Lei, Xiaofeng Zhu and Rongyao Hu. 'Adaptive reverse graph learning for robust subspace learning'. In: *Information Processing & Management* 58.6 (2021), p. 102733.

[179] Xiao Yuan, Suguru Endo, Qi Zhao, Ying Li and Simon C Benjamin. 'Theory of variational quantum simulation'. In: *Quantum* 3 (2019), p. 191.

[180] Anqi Zhang and Shengmei Zhao. 'Evolutionary-based quantum architecture search'. In: *arXiv preprint arXiv:2212.00421* (2022).

[181] Chris Zhang, Mengye Ren and Raquel Urtasun. 'Graph hypernetworks for neural architecture search'. In: *arXiv preprint arXiv:1810.05749* (2018).

[182] Kaining Zhang, Liu Liu, Min-Hsiu Hsieh and Dacheng Tao. 'Escaping from the barren plateau via gaussian initializations in deep variational quantum circuits'. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 18612–18627.

[183] Shi-Xin Zhang, Jonathan Allcock, Zhou-Quan Wan, Shuo Liu, Jiace Sun, Hao Yu, Xing-Han Yang, Jiezhong Qiu, Zhaofeng Ye, Yu-Qin Chen et al. 'Tensorcircuit: a quantum software framework for the nisq era'. In: *Quantum* 7 (2023), p. 912.

[184] Shi-Xin Zhang, Chang-Yu Hsieh, Shengyu Zhang and Hong Yao. 'Differentiable quantum architecture search'. In: *Quantum Science and Technology* 7.4 (2022), p. 045023.

[185] Shi-Xin Zhang, Chang-Yu Hsieh, Shengyu Zhang and Hong Yao. 'Neural predictor based quantum architecture search'. In: *Machine Learning: Science and Technology* 2.4 (2021), p. 045027.

[186] Bin Zhao, James T Kwok and Changshui Zhang. 'Multiple kernel clustering'. In: *Proceedings of the 2009 SIAM international conference on data mining*. SIAM. 2009, pp. 638–649.

[187] Xiaofeng Zhu, Rongyao Hu, Cong Lei, Kim Han Thung, Wei Zheng and Can Wang. 'Low-rank hypergraph feature selection for multi-output regression'. In: *World Wide Web* 22 (2019), pp. 517–531.

[188] Xiaofeng Zhu, Cong Lei, Hao Yu, Yonggang Li, Jiangzhang Gan and Shichao Zhang. 'Robust Graph Dimensionality Reduction.' In: *IJCAI*. 2018, pp. 3257–3263.