Quantum Machine Intelligence (2024) 6:33
https://doi.org/10.1007/s42484-024-00169-w

RESEARCH ARTICLE

Quantum optimization for training quantum neural networks

Yidong Liao[1,2] · Min-Hsiu Hsieh[3] · Chris Ferrie[1]

Received: 17 October 2023 / Accepted: 9 May 2024 / Published online: 1 June 2024

Abstract
Training quantum neural networks (QNNs) using gradient-based or gradient-free classical optimization approaches is severely impacted by the presence of barren plateaus in the cost landscapes. In this paper, we devise a framework for leveraging quantum optimization algorithms to find optimal parameters of QNNs for certain tasks. To cast the optimization problem of training QNN into the context of quantum optimization, the parameters in QNN are quantized—moved from being classical to being stored in quantum registers which are in addition to those upon which the QNN is performing its computation. We then coherently encode the cost function of QNNs onto relative phases of a superposition state in the Hilbert space of the QNN parameters. The parameters are tuned with an iterative quantum optimization structure using adaptively selected Hamiltonians. The quantum mechanism of this framework exploits hidden structure in the QNN optimization problem and hence is expected to provide beyond-Grover speed up, mitigating the barren plateau issue.

Keywords Quantum neural networks · Quantum optimization · Quantum machine learning · Machine learning

1 Introduction

1.1 Quantum neural networks

Quantum neural networks (QNNs) are considered to be a leading candidate to achieve a quantum advantage in noisy intermediate-scale quantum (NISQ) devices. A QNN consists of a set of parameterized quantum gates within a predefined circuit ansatz. The design of the ansatz together with the value of the gate parameters determine the outcome of the QNN. In order to successfully perform certain tasks, QNNs must be trained to find optimal parameters for generating desired outcomes. In the majority of QNN research, the training is carried out by employing variational hybrid quantum-classical algorithms (McClean et al. 2016), in which the parameters are optimized by a classical optimizer using gradient-based or gradient-free approaches. In this paper, we achieve a scalable, maximally quantum pipeline of the applications of QNNs by replacing the classical optimizer by quantum optimizer. In short, we employ quantum optimization methods for training QNNs.

There are two main avenues for the application of QNNs. The first uses QNNs to generate quantum states that minimize the expectation value of a given Hamiltonian, such as the case in variational quantum eigensolvers (VQE) (Peruzzo et al. 2014) for chemistry problems or quantum approximate optimization algorithms (QAOA) (Farhi et al. 2014) for combinatorial optimization problems. The second path uses QNNs as data-driven machine learning models to perform discriminative (Farhi and Neven 2018; Schuld et al. 2020; Du et al. 2018b) and generative (Benedetti et al. 2019; Hamilton et al. 2019; Huang et al. 2020; Mitarai et al. 2018; Zeng et al. 2019) tasks for which QNNs could have more expressive power than their classical counterparts (Du et al. 2018a). Though an ever increasing amount of effort is being put into QNN research, there is evidence that they will be difficult to train due to flat optimization landscapes called barren plateaus (McClean et al. 2018).

The barren plateau issue has spawned several studies on the strategies to avoid them, including layerwise training

✉ Yidong Liao
yidong.liao@student.uts.edu.au

Min-Hsiu Hsieh
minhsiuh@gmail.com

Chris Ferrie
christopher.ferrie@uts.edu.au

[1] Centre for Quantum Software and Information, University of Technology Sydney, Sydney, NSW, Australia

[2] Sydney Quantum Academy, Sydney, NSW, Australia

[3] Hon Hai Quantum Computing Research Center, Taipei, Taiwan

(Skolik et al. 2020), using local cost functions (Cerezo et al. 2020), correlating parameters (Volkoff and Coles 2020), and pre-training (Verdon et al. 2019b), among others (Du et al. 2020a, b; Zhang et al. 2020). Such strategies give hope that the variational quantum-classical algorithms may avoid the exponential scaling due to the barren plateau issue. However, it has been shown that these strategies do not avoid another type of barren plateaus induced by hardware noise (Wang et al. 2020), and some strategies may lack theoretical grounding (Campos et al. 2020). In addition to noise, there is also other sources of barren plateaus due to entanglement growth (Marrero et al. 2020). Moreover, it has been shown that gradient-free approaches are also adversely affected by barren plateaus (Arrasmith et al 2020).

Our work presents a new alternative to training QNNs with a maximally coherent (i.e., quantum) protocol (Fig. 1).

1.2 Prior work

The above-noted results indicate that training QNNs using classical optimization methods have unprecedented challenges as the system scales up. Therefore, one seeks to leverage alternative optimization methods for training QNNs. Indeed, preliminary attempts have been made in this direction. Verdon et al. proposed a QAOA-like training protocol for QNNs (Verdon et al. 2018), and Gilyén et al. developed a quantum algorithm for calculating gradients faster than classical methods (Gilyén et al. 2019a). In these two works, to cast the optimization problem of training QNNs into the context of quantum optimization, the network parameters in the QNN are quantized—moved from being classical to being stored in quantum registers, which are in addition to those upon which the QNN is performing its computation. The quantized parameters are used as control registers of the parameterized gates on the QNN registers. The parameters can now be in superposition, which one hopes would allow for a *quantum parallelism*-type computation of the QNN with multiple parameter configurations.

In Ref. (Verdon et al. 2018), the quantum training process can be described as the state evolution in the joint Hilbert space of the parameter register and the QNN register. Their quantum training protocol consists of two alternating operations in a QAOA fashion—the first operation acts on both the parameter register and QNN register to encode the cost function of QNN onto a relative phase of the parameter state. The second operation acts only on the parameter register, and it is a variant of the original QAOA Mixers, tailored for the case that the parameters in the QNN are continuous

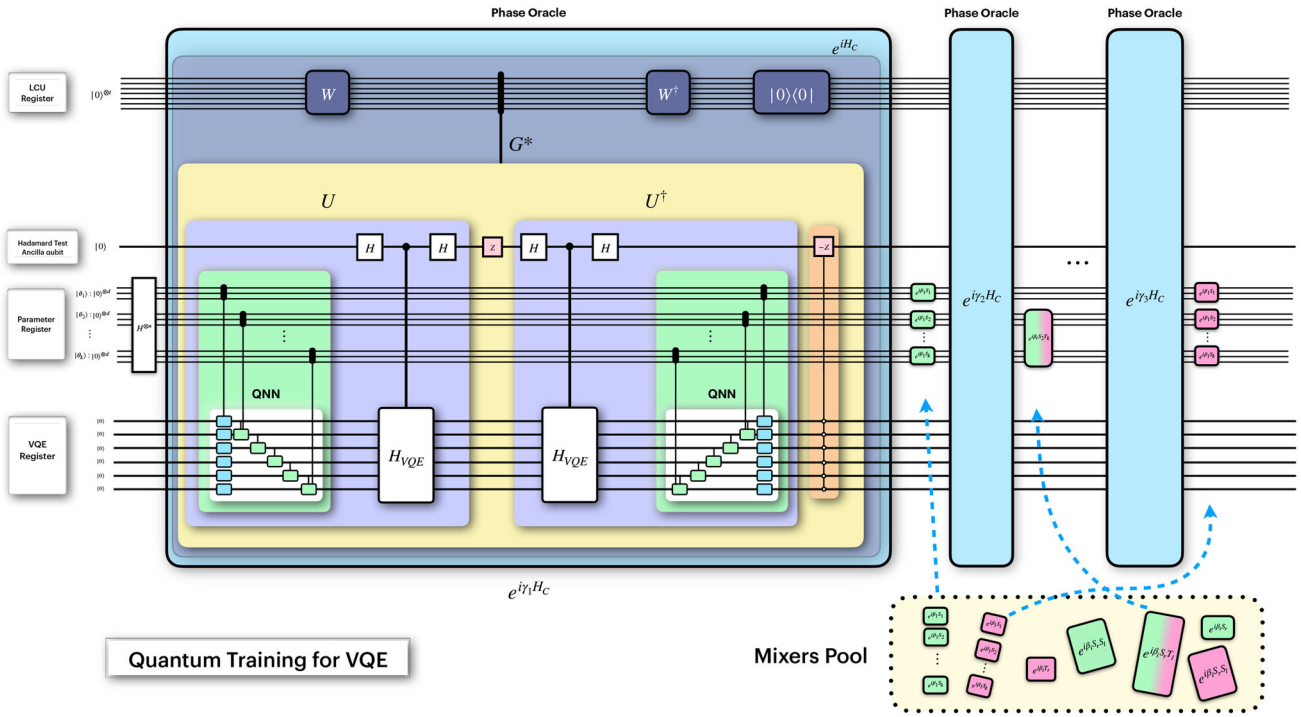


Fig. 1 Schematic of our quantum training algorithm for VQE. Here, we use the training of VQE as an example to present the schematic circuit construction of our quantum training algorithm for QNN. A video animation of the circuit construction is available at https://youtu.be/RVWkJZY6GNY. (This is vector image and best view with the zoom feature in standard PDF viewers.) Note: *1.* In all figures of this paper, we omit the minus signs in all time-evolution-like terms (i.e., exponential of a Hamiltonian e^{-iHt}) for sake of brevity and space. *2.* Some quantum registers are not depicted in this figure due to the limitation of space

variables. These two operation can be mathematically expressed as $e^{-i\gamma_i C(\boldsymbol{\theta})}$ and $e^{-i\beta_i H_M}$, where $\boldsymbol{\theta}$ are the parameters of QNN, $C(\boldsymbol{\theta})$ is the cost function of the QNN, γ_i and β_i are tunable hyperparameters, H_M is the mixer Hamiltonian. By heuristically tuning the hyperparameters, the quantum training is expected to home in on the optimal parameters of the QNN after several iterations of the QAOA alternating operations. We illustrate the alternating operations of their quantum training in Fig. 2.

Despite being the pioneering application of the QAOA method for training QNNs, the protocol in Ref. (Verdon et al. 2018) has some limitations. In the phase encoding operation, the parameter register and the QNN register are generally always entangled. This will have the effect of causing phase decoherence in the parameter eigenbasis. To minimize the effect of this decoherence, the tuneable hyperparameter γ_i must be sufficiently small—in other words, the phase encoding is coherent only in the first order of γ_i. To overcome this limitation—to enact phase encoding operation with arbitrary hyperparameters—the phase encoding operation with a small hyperparameter $\Delta\gamma$ should be repeated an excessive amount of times. This simulates the phase encoding operation with a large hyperparameter γ via $e^{-i\gamma C(\boldsymbol{\theta})} = e^{-i\Delta\gamma C(\boldsymbol{\theta})} e^{-i\Delta\gamma C(\boldsymbol{\theta})} e^{-i\Delta\gamma C(\boldsymbol{\theta})} \ldots$ These repetitions will yield large overhead in the complexity of the algorithm. In Ref. (Gilyén et al. 2019a), a *phase oracle* is designed for the phase encoding and can achieve it coherently and efficiently. (Note that throughout this paper, the term *phase oracle* has different meaning than the one in Ref. (Gilyén et al. 2019a); our *phase oracle* stands for the term *fractional phase oracle* in Ref. (Gilyén et al. 2019a).)

Nevertheless, they did not utilize the phase encoding as a component of QAOA routine to accomplish a fully quantum training algorithm for QNNs. Instead, they use the phase oracle as a component of quantum evaluation of the gradient of a QNN, which serves for gradient-based classical training of QNNs. However, this improvement will not be practically useful due to the barren plateau issue of QNNs.

In this paper, we devise an improved framework for training QNNs, taking advantage of the well-established parts in Refs. (Verdon et al. 2018) and Gilyén et al. (2019a), while eliminating the shortcomings. A schematic of our quantum training framework for QNNs is depicted in Fig. 3. More specifically, we achieve the following:

1. We replace the phase encoding operations in QAOA-like protocol of Ref. (Verdon et al. 2018) by the *phase oracle* in Ref. (Gilyén et al. 2019a). This achieves coherent encoding of the cost function onto a relative phase of parameter state, while avoiding the limitations of the hyper-parameters in the phase encoding.
2. For the mixers in the QAOA-like routine, we adopt a similar approach to Ref. (Zhu et al. 2020) by making the mixers adaptive—that is, we allow different mixers for each layer (particularly, to allow entangling mixers that act across different parameters). This potentially leads to a dramatic shortening of the depth of QAOA layers while significantly improving the quality of the solution (the optimal QNN parameters found by the QAOA routine).

By making the mixers flexible and adaptive to specific optimization problems, it is demanding to find an efficient

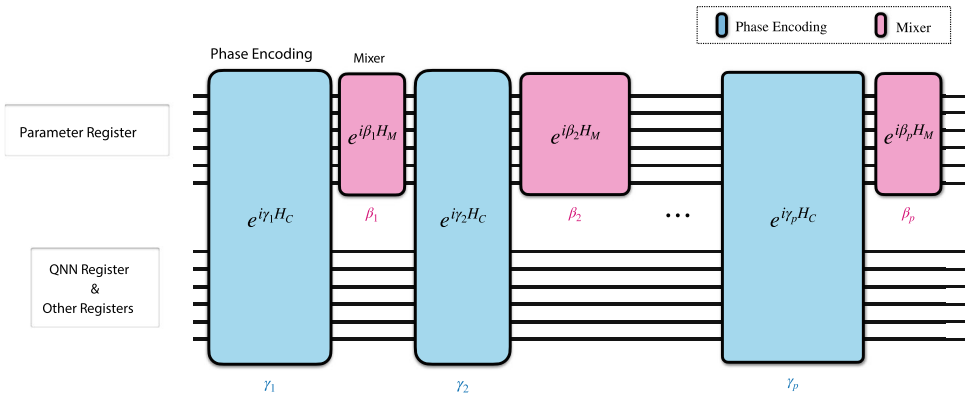


Fig. 2 QAOA-like training protocol for QNN, proposed in Ref. (Verdon et al. 2018). The quantum training protocol consists of two alternating operations in a QAOA fashion—the first operation acts on both the parameter register and QNN register to encode the cost function of QNN onto a relative phase of the parameter state. This operation is represented by the blue blocks in the figure. The second operation acts only on the parameter register, and it is a variant of the original QAOA Mixers, tailored for the case that the parameters in the QNN are continuous variables. This operation is represented by the pink blocks in the figure. These two operations can be mathematically expressed as $e^{-i\gamma_i C(\boldsymbol{\theta})}$ and $e^{-i\beta_i H_M}$, where $\boldsymbol{\theta}$ are the parameters of QNN, $C(\boldsymbol{\theta})$ is the cost function of the QNN, γ_i and β_i are tunable hyperparameters, and H_M is the mixer Hamiltonian. The width of each block represents the hyperparameters γ_i and β_i—the wider the block, the larger the value of the hyperparameters. The phase encoding operation $e^{-i\gamma_i H_C}$ acts as $e^{-i\gamma_i C(\boldsymbol{\theta})}$

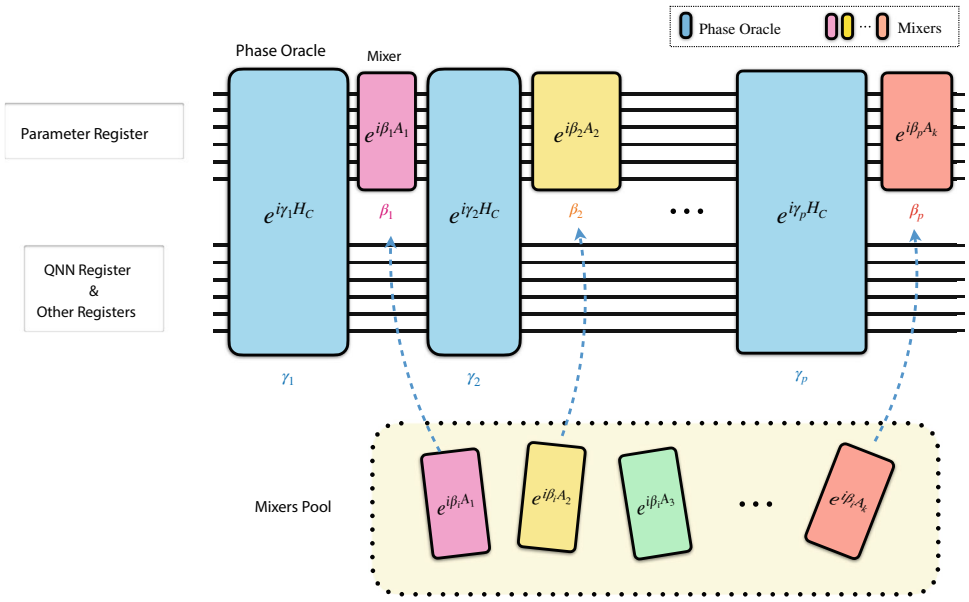


Fig. 3 Schematic of our framework for quantum training of QNN. Our quantum training for QNN taking advantage of the well-established parts in Refs. (Verdon et al. 2018) and Gilyén et al. (2019a), while eliminating their shortcomings. We replace the phase encoding operations in QAOA-like protocol of Ref. (Verdon et al. 2018) (as depicted in Fig 2) by the *phase oracle* in Ref. (Gilyén et al. 2019a). For the mixers in the QAOA-like routine, we allow different mixers for each layer, similar to Ref. (Zhu et al. 2020). In this figure, the color of each block represents the nature of the corresponding Hamiltonian: different color corresponds to different Hamiltonian (One can see that the cost Hamiltonian is the same throughout the training, whereas the mixer varies from layer to layer). The mixers pool contains the proper mixers tailored to our QNN training problem. These rules also apply to the other circuit schematic in this paper

way of determining the best sequence of mixers and the optimized hyperparameters. To address these, we adopt machine learning approaches (in particular, recurrent neural networks and reinforcement learning) as proposed in Refs. (Verdon et al. 2019b; Wauters et al. 2020; Warren et al 2020; Yao et al. 2020b). The quantum mechanism of this framework is the best candidate to exploit hidden structure in the QNN optimization problem, which would provide beyond-Grover speed up and mitigate the barren plateau issues for training QNNs.

1.3 Paper outline

The remainder of this paper is organized as follows: in Section 2, we review some essential preliminaries—particularly on the details of QAOA and its variants, from which we designed a new variant of QAOA tailored for our QNN training problem. Section 2.3 introduces a way of quantising parameters of a QNN—that is, we show how to create superposition of a QNN with multiple parameter configurations. In Section 3, we present quantum training by Grover adaptive search as a baseline prior to our quantum training framework using QAOA. In Section 4, we present the details of our framework including how to implement the phase oracle that can achieve coherent phase encoding of the cost function of a QNN, and which mixers to use for the QAOA routine, as well as the strategy to determine the mixers sequence and optimize their hyper-parameters. Section 5 presents the deployment potential of our quantum training to a variety of application including training VQE, learning a pure state, and training a quantum classifier. The final section summarizes our work and provides outlook for future work.

2 Preliminaries

2.1 Quantum optimization algorithms

2.1.1 Zoo of quantum optimization algorithms

For completeness and context, we list some typical quantum optimization algorithms in Table 1, including the primitive ones (adiabatic, quantum walks, QAOA, Grover adaptive search), their hybridizations (Jiang et al. 2017; Marsh and Wang 2018; Mbeng et al. 2019; Morley et al. 2019; Wang 2017), and their variants (Bulger 2005; Guéry-Odelin et al. 2019; Hegade et al. 2020; Hadfield et al. 2019a; Verdon et al. 2018; Whitfield et al. 2010; Yao et al. 2020b; Zhu et al. 2020). In this paper, for the training of QNNs, we focus on utilizing QAOA and its variants as well as Grover adaptive search, which we will review in the following subsections.

Table 1 Zoo of quantum optimization algorithms

Primitives	Adiabatic	Quantum walk	QAOA	Grover adaptive search
Hybridization of primitives	Hybrid adiabatic-quantum-walk algorithms, others			
Variants of primitives	Shortcut to adiabaticity	Quantum stochastic walk	Adaptive QAOAs, Others	Quantum basin hopping

The first row contains the four primitive quantum optimization algorithms by adiabatic quantum evolution, quantum walks, QAOA, and Grover adaptive search. The second row contains the hybridization among these four primitives, e.g., hybrid adiabatic-quantum-walk algorithms (Morley et al. 2019). The third row contains the variants of the primitives, e.g., variants of QAOA include adaptive QAOAs (Yao et al. 2020b; Zhu et al. 2020), and others Hadfield et al. (2019a); Verdon et al. (2018)

Before that, however, some remarks on the fundamental differences of the adiabatic and QAOA protocols are in order. QAOA can be seen as a "trotterized" version of adiabatic evolution: the mixer Hamiltonians being the initial Hamiltonian in the analogous adiabatic algorithm, and the cost Hamiltonians being the final Hamiltonian. However, short-depth QAOA is not really the digitized version of the adiabatic problem, but rather an ad hoc ansatz. In Ref. (Streif and Leib 2019), it is shown that QAOA is able to deterministically find the solution of specially constructed optimization problems in cases where quantum annealing fails. We emphasize that QAOA is an interference-based algorithm such that non-target states interfere destructively while the target states interfere constructively. In Fig. 4, we depict this interference process of QAOA.

2.1.2 QAOA and its variants

In this section, we review the original quantum approximation optimization algorithm (QAOA) proposed in Ref. (Farhi et al. 2014) and its variants. Consider an unconstrained optimization problem on n-bit strings $\mathbf{z} = (z_1, z_2, z_3, ...z_n)$ where $z_i \in \{-1, 1\}$. We seek the optimal bit string $\mathbf{z}$ that maximizes (or minimizes) a cost function $C(\mathbf{z})$. Given the cost function $C(\mathbf{z})$ of a problem instance, the algorithm is characterized by two Hamiltonians: the *cost Hamiltonian* H_C and the *mixing Hamiltonian* H_{M}. The cost Hamiltonian H_C encodes the cost function $C(\mathbf{z})$ to be optimized and acts on n-qubit computational basis states as

$$H_C|\mathbf{z}\rangle = C(\mathbf{z})|\mathbf{z}\rangle.$$

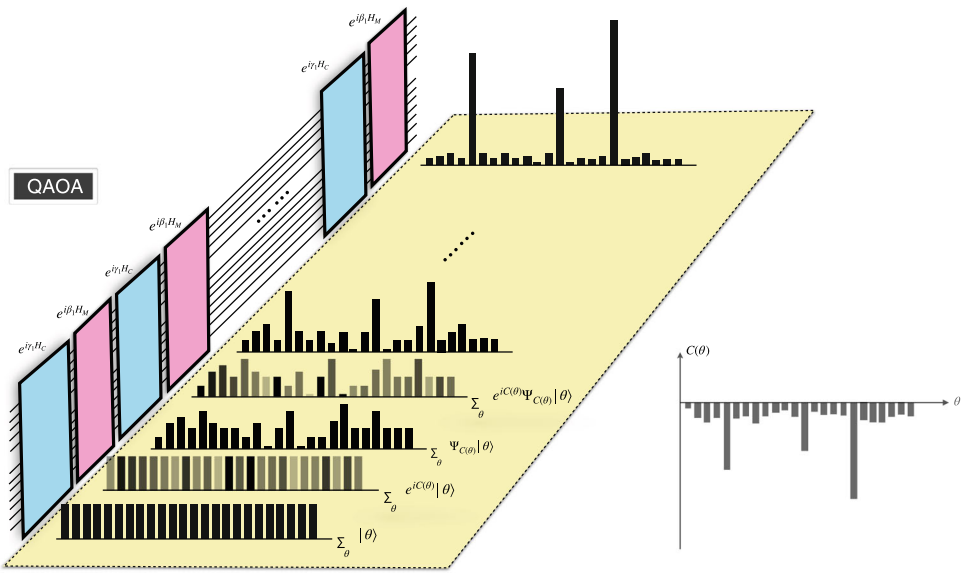


Fig. 4 Interference process of QAOA. QAOA is an interference-based algorithm such that non-target states interfere destructively, while the target states interfere constructively. Here, we illustrate this interference process by presenting the evolution of the quantum state of the parameters (black bar graphs on the yellow plane) alongside with the QAOA operations (blue and pink boxes on circuit lines, representing the phase encoding and mixers, respectively). The starting state $\sum_\theta |\theta\rangle$ (omitting the normalization factor) is the even superposition state of all possible parameter configurations. After the first phase encoding operation, the state becomes $\sum_\theta e^{-iC(\boldsymbol{\theta})}|\theta\rangle$ for which we use opacity of the bars to indicate the value of the phase; the magnitudes of the amplitudes in the state remain unchanged. After the first mixer, the state becomes $\sum_\theta \Psi_{C(\boldsymbol{\theta})}|\theta\rangle$ in which the magnitudes of the amplitudes in the state have changed. Similar process happens to the following operations, until the amplitudes of the optimal parameter configurations are amplified significantly (the furthest bar graph). The grey bar graph in the right corner is the cost function being optimized by QAOA

The mixing Hamiltonian H_{M} is chosen as to be

$$H_{\mathrm{M}} = \sum_{j=1}^{n} X_j,$$

where X_j is the Pauli X operator acting on the jth qubit. The initial state is the even superposition state of all possible solutions: $|s\rangle = \frac{1}{\sqrt{2^n}} \sum_{\mathbf{z}} |\mathbf{z}\rangle$. The QAOA algorithm consists of alternating time evolution under the two Hamiltonians H_C and H_{M} for p rounds, where the duration in round j is specified by the parameters γ_j and β_j, respectively. After all p rounds, the state becomes

$$|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle = e^{-i\beta_p H_{\mathrm{M}}} e^{-i\gamma_p H_C} \ldots e^{-i\beta_2 H_{\mathrm{M}}} e^{-i\gamma_2 H_C} e^{-i\beta_1 H_{\mathrm{M}}} e^{-i\gamma_1 H_C} |s\rangle.$$

The alternating operations can be illustrated as in Fig. 5. Finally, a measurement in the computational basis is performed on the state. Repeating the above state preparation and measurement, the expected value of the cost function,

$$\langle C\rangle = \langle \boldsymbol{\beta}, \boldsymbol{\gamma}| H_C |\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle,$$

can be estimated from the samples produced from the measurements.

The above steps are then repeated altogether, with updated sets of time parameters $\gamma_1, \ldots, \gamma_p, \beta_1, \ldots, \beta_p$. Typically, a classical optimization loop (such as gradient descent) is used to find the optimal parameters that maximize (or minimize) the expected value of the cost function $\langle C\rangle$. Then, measuring the resulting state of the optimal parameters provides an approximate solution to the optimization problem.

There has been a lot of progress on QAOA recently on both the experimental and theoretical fronts. There is evidence suggesting that QAOA may provide a significant quantum advantage over classical algorithms (Barkoutsos et al. 2020; Niu et al. 2019) and that it is computationally universal (Lloyd 2018; Morales et al. 2019). Despite these advances, there are limitations of QAOA. The performance improves with circuit depth, but circuit depth is still limited in near-term quantum processors. Moreover, deeper circuits translate into more variational parameters, which introduces challenges for the classical optimizer in minimizing the objective function. Ref. (Bravyi et al. 2019) show that the locality and symmetry of QAOA can limit its performance. These issues can be attributed to the form of the QAOA ansatz. A short-depth ansatz that is further tailored to a given combinatorial problem could therefore address the issues with the standard QAOA ansatz. However, identifying such an alternative is a highly non-trivial problem given the vast space of possible ansatzes. Farhi et al. (2017) allowed the mixer to rotate each qubit by a different angle about the x-axis and modified the cost Hamiltonian based on hardware connectivity. This modification was made primarily out of hardware capability concerns with the hope that superior-than-classical performance can be experimentally verified.

LH-QAOA In Ref. (Hadfield et al. 2019b), Hadfield et al. considered alternative mixers including entangling ones on two qubits. The selection of mixers is based on the criteria of preserving the relevant subspace for the given combinatorial problem, for which they entitled it Local Hamiltonian-QAOA (LH-QAOA). Here, we depict the quantum circuit schematic of LH-QAOA in Fig. 6.

QDD In Refs. (Verdon et al. 2018, 2019a), Verdon et al. adjusted the mixers for continuous optimization problem in which the parameters to be optimized are continuous variables. In the original QAOA ansatz, the mixer is chosen to be single-qubit X rotations applied on all qubits. These constitute an uncoupled sum of generators of shifts in the computational basis. Similarly, the appropriate mixers in the continuous case should shift the value for each digitized continuous variables stored in independent registers. They

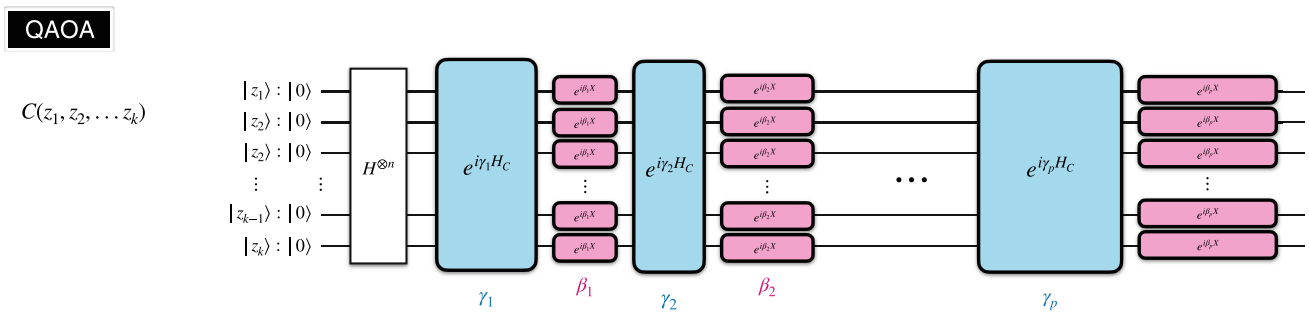


Fig. 5 Quantum circuit schematic of the operations in the original QAOA. The state is initialized by applying Hadamard gates on each qubit, represented as $H^{\otimes n}$. This results in the equal superposition state of all possible solutions. QAOA consists of alternating time evolution under the two Hamiltonians H_C and H_{M} for p rounds, where the duration in round j is specified by the parameters γ_j and β_j, respectively. In the original QAOA, the mixing Hamiltonian H_{M} is chosen as to be $H_{\mathrm{M}} = \sum_{j=1}^{n} X_j$. After all p rounds, the state becomes $|\boldsymbol{\beta}, \boldsymbol{\gamma}\rangle = e^{-i\beta_p H_{\mathrm{M}}} e^{-i\gamma_p H_C} \ldots e^{-i\beta_2 H_{\mathrm{M}}} e^{-i\gamma_2 H_C} e^{-i\beta_1 H_{\mathrm{M}}} e^{-i\gamma_1 H_C} |s\rangle$

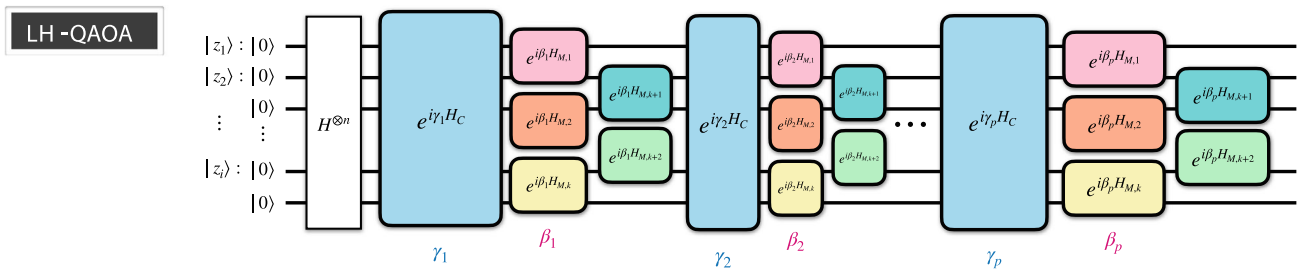


Fig. 6 Quantum circuit schematic of the operations in LH-QAOA. The overall process of LH-QAOA is similar to that of the original QAOA in Fig. 5, where the difference is that the mixer of LH-QAOA contains entangling an mixer Hamiltonian on two qubits. These are represented by the $H_{M,i}$ blocks with various colors in the figure. Note that in order to avoid an excessive amount of hyper-parameters, Hadfield et al. (2019b) choose the β_j for each $H_{M,i}$ to be the same in every layer

entitled it Quantum Dynamical Descent (QDD). Here, we depict the quantum circuit schematic of QDD in Fig. 7.

ADAPT-QAOA . LH-QAOA and QDD showcase the potential of problem-tailored mixers, but do not provide a general strategy for choosing mixers for different optimization problems. In Ref. (Zhu et al. 2020), Zhu et al. replaced the fixed mixer H_M by a set of different mixers A_k that change from layer to layer. They entitled this variation of QAOA as ADAPT-QAOA. This adaptive approach would dramatically shorten the depth of QAOA layers while significantly improving the quality of the solution. Here, we depict the quantum circuit schematic of ADAPT-QAOA in Fig. 8.

Compared to the original QAOA, allowing Y mixers and entangling mixers enables ADAPT-QAOA to dramatically improve algorithmic performance while achieving rapid convergence for problems with complex structures. This effect of the adaptive mechanism can be illustrated in Fig. 9.

The advantage of this adaptive ansatz may come from the counter-diabatic (CD) driving mechanism. Numerical evidence shows that the adaptive mixer sequence chosen by the algorithm coincides with that of "shortcut to adiabaticity" by CD driving (Zhu et al. 2020). Inspired by the CD driving procedure, another variant of QAOA, CD-QAOA (Yao et al. 2020b), also uses an adaptive ansatz to achieve similar advantages. CD-QAOA is designed for preparing the ground state of quantum-chaotic many-body spin chains. By using terms occurring in the adiabatic gauge potential as additional control unitaries, CD-QAOA can achieve fast high-fidelity many-body control.

Inspired by above variants of QAOA, we design a new variant of QAOA tailored for our QNN training problem. In our case, for QNN training, the parameters we are optimizing (the angles of rotation gates) are continuous (real) values. Therefore, the choice of mixer Hamiltonian has to be adapted (as in QDD). We also want take advantage of including alternative mixers and allowing adaptive mixers for different layers (as in ADAPT-QAOA). Thus, the proper QAOA ansatz for our QNN training problem should be an adaptive continuous version of QAOA, which we call *AC-QAOA*. Here, we depict the quantum circuit schematic of AC-QAOA in Fig. 10.

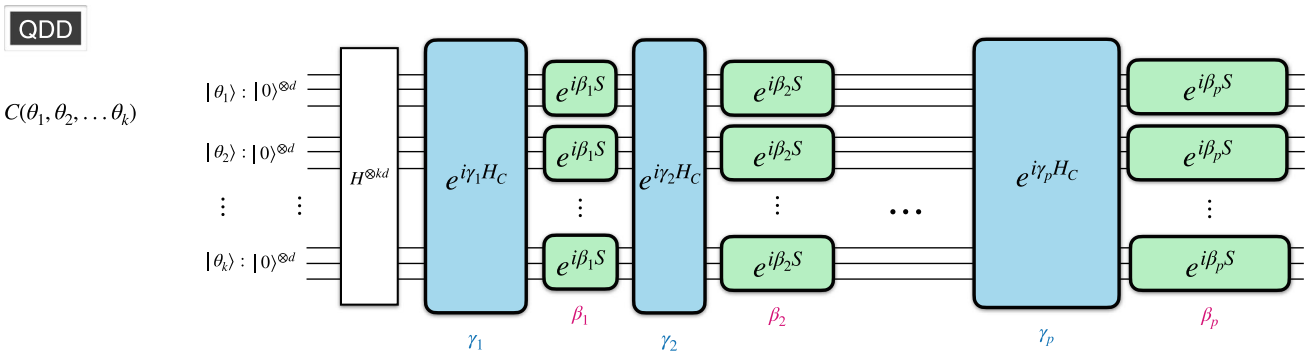


Fig. 7 Quantum circuit schematic of QDD. QDD solves optimization problems of continuous variable. In this figure, θ_i are the continuous variables to be optimized in the training, where each θ_i is digitized into binary form and stored in an independent register. The overall process of QDD is similar to that of the original QAOA, where the difference is that the mixer of QDD with Hamiltonian S is acting on the registers of θ_i (rather than single qubits as in the original QAOA). The effect of the mixer in QDD is to shift the value for each θ_i

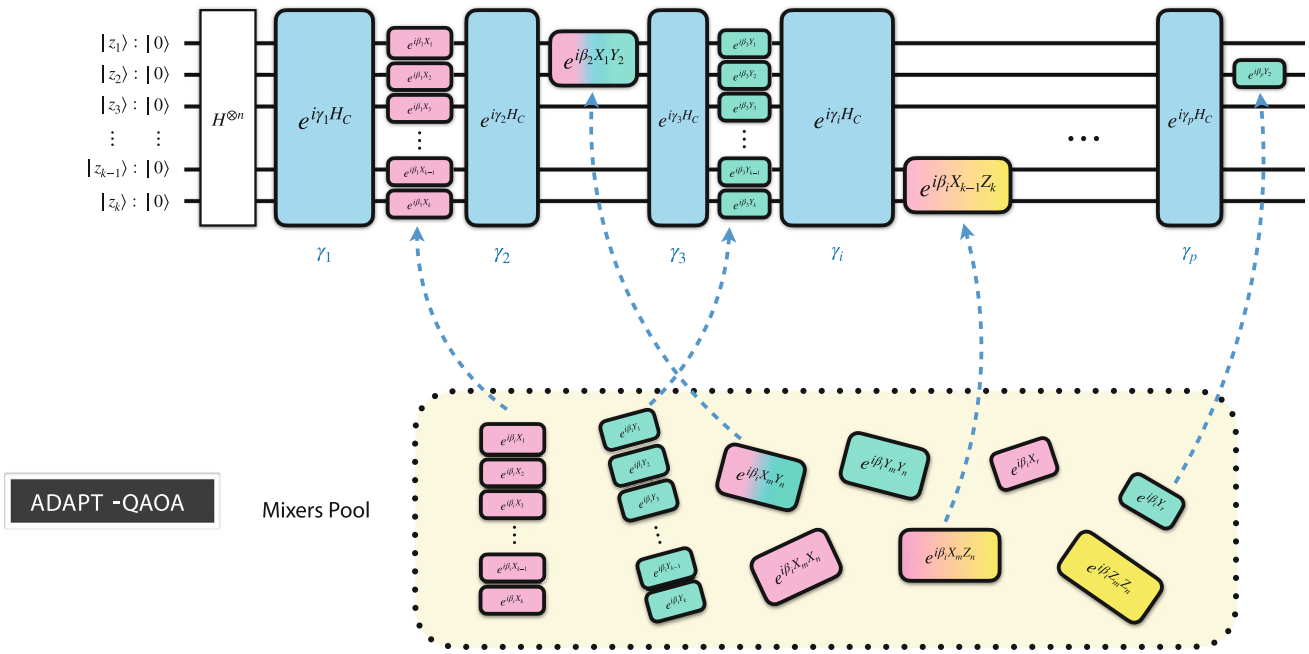


Fig. 8 Quantum circuit schematic of ADAPT-QAOA. The overall process of LH-QAOA is similar to that of the original QAOA in Fig. 5, where the difference is that the mixer of LH-QAOA contains variable mixers taken from a *mixers pool*. Define Q to be the set of qubits. The mixer pool of ADAPT-QAOA is $P_{\text{ADAPT-QAOA}} = \cup_{i\in Q}\left\{X_i, Y_i, \sum_{i\in Q} X_i, \sum_{i\in Q} Y_i\right\} \cup_{i,j\in Q\times Q} \{B_i C_j | B_i, C_j \in \{X, Y, Z\}\}$

2.1.3 Grover adaptive search

Grover's algorithm is generally used as a search method to find a set of desired solutions from a set of possible solutions. Dürr and Høyer presented an algorithm based on Grover's method that finds an element of minimum value inside an array of N elements using on the order of $O(\sqrt{N})$ queries to the oracle (Durr and Hoyer 1996). Baritompa et al. (2005) applied Grover's algorithm for global optimization, which they call Grover Adaptative Search (GAS). GAS has been applied in training classical neural networks (Liao et al. 2020) and polynomial binary optimization (Gilliam et al. 2020). In the following, we outline GAS.

Consider a function $f : X \to \mathbb{R}$, where for ease of presentation assume $X = \{0, 1\}^n$. We are interested in solving $\min_{x\in X} f(x)$. The main idea of GAS is to construct an "adaptive" oracle for a given threshold y such that it flags all states $x \in X$ satisfying $f(x) < y$, namely the oracle marks a solution x if and only if another boolean function g_y satisfies $g_y(x) = 1$, where

$$g_y(x) = \begin{cases} 1 & \text{if } f(x) < y \\ 0 & \text{otherwise} \end{cases}, \tag{1}$$

The oracle $\mathcal{O}_{\text{Grover}}$ then acts as

$$\mathcal{O}_{\text{Grover}}|x\rangle = (-1)^{g_y(x)}|x\rangle. \tag{2}$$

We use Grover search to find a solution $\tilde{x}$ with a function value better than y. Then, we set $y = f(\tilde{x})$ and repeat until some formal termination criteria is met—for example, based on the number of iterations, time, or progress in y.

2.2 Swap test, Hadamard test, and the Grover operator

This section introduces the swap test, Hadamard test, and their corresponding Grover operators, which will be used in the phase encoding of the cost function of QNNs.

2.2.1 Swap test and its Grover operator

Let $|p_j\rangle, |t\rangle$ be the resulting quantum states of unitary operators P_j and T, respectively—that is, $|p_j\rangle = P_j|0\rangle^{\otimes n}$ and $|t\rangle = T|0\rangle^{\otimes n}$. The swap test is a technique that can be used to estimate $|\langle p_j|t\rangle|^2$ (Buhrman et al. 2001). The circuit of swap test is shown in Fig. 11.

We denote the unitary of the swap test circuit (dotted green box in Fig. 11) as U_j, which can be written as

$$U_j := [H \otimes I \otimes I] \cdot [|0\rangle\langle 0| \otimes P_j \otimes T + |1\rangle\langle 1| \otimes T \otimes P_j] \cdot [H \otimes I \otimes I]. \tag{3}$$

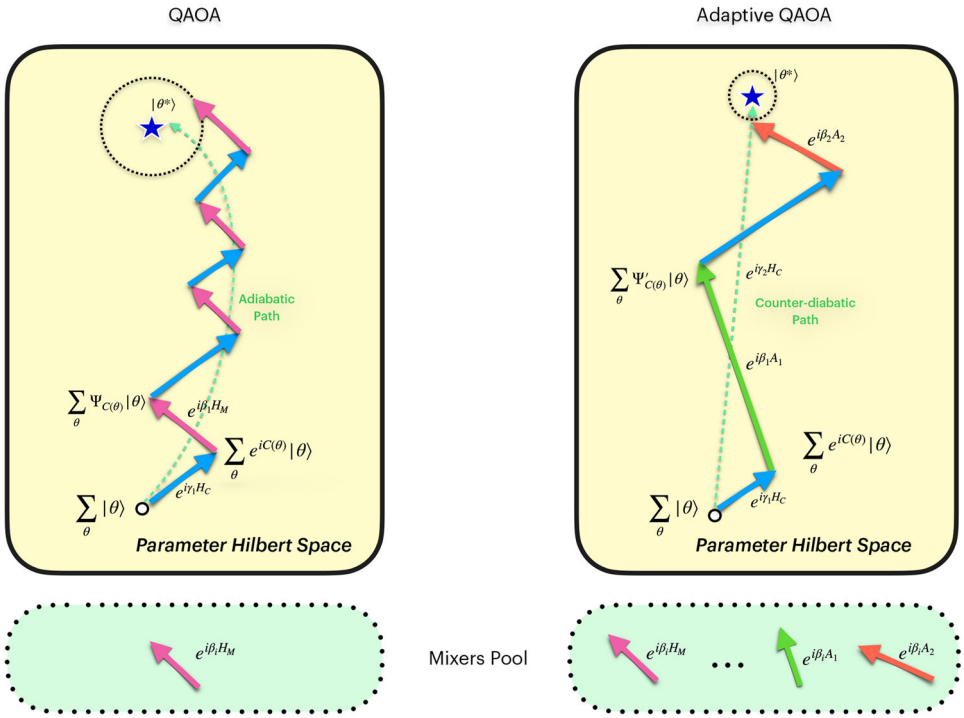


Fig. 9 Comparison of original QAOA and ADAPT-QAOA. In the left and right panels of this figure, we depict the state change in the Hilbert space of the parameters to be optimized, for the original QAOA and ADAPT-QAOA, respectively. The starting state $\sum_\theta|\theta\rangle$ (omitting the normalization factor), represented by the rounded dot at the bottom of each space, is the even superposition state of all possible solutions. The arrows represent the state evolution generated by the cost Hamiltonian and mixer Hamiltonian, and the color and direction of the arrows indicate the nature of the evolution. Blue arrows represent the evolution by the cost Hamiltonian. Arrows of other colors represent the evolution by different mixer Hamiltonians. In the original QAOA, there is only one mixer (shown in pink) available. Whereas in ADAPT-QAOA, there are more alternative mixers to chose from the mixers pool. The two algorithms try to reach the target state $|\theta^*\rangle$ (represented by the blue star) by stacking these arrows, which represent the alternating operations of two QAOAs. For reference, we sketched the relevant paths—adiabatic path for the original QAOA and counter-diabatic path for ADAPT-QAOA—along the state evolution of the two QAOAs. As can be seen, the ADAPT-QAOA takes much fewer iterations to reach a closer point to the target state. This illustrates that compared to the original QAOA, allowing alternative mixers enables ADAPT-QAOA to dramatically improve algorithmic performance while achieving rapid convergence

The output state from U_j is denoted as $|\phi_j\rangle$:

$$|\phi_j\rangle = \frac{1}{\sqrt{2}}(|+\rangle|p_j\rangle|t\rangle + |-\rangle|t\rangle|p_j\rangle). \tag{4}$$

Rearranging the terms, we have

$$|\phi_j\rangle = \frac{1}{2}\left(|0\rangle\otimes(|p_j\rangle|t\rangle + |t\rangle|p_j\rangle) + |1\rangle\otimes(|p_j\rangle|t\rangle - |t\rangle|p_j\rangle)\right). \tag{5}$$

Denote $|u_j\rangle$ and $|v_j\rangle$ as the normalized states of $|p_j\rangle|t\rangle + |t\rangle|p_j\rangle$ and $|p_j\rangle|t\rangle - |t\rangle|p_j\rangle$, respectively. Then, there is a real number $\theta_j \in [\pi/4, \pi/2]$ such that

$$|\phi_j\rangle = \sin\theta_j|0\rangle|u_j\rangle + \cos\theta_j|1\rangle|v_j\rangle. \tag{6}$$

θ_j satisfies $\cos\theta_j = \sqrt{1-|\langle p_j|t\rangle|^2}/\sqrt{2}$, $\sin\theta_j = \sqrt{1+|\langle p_j|t\rangle|^2}/\sqrt{2}$; therefore, we have

$$|\langle p_j|t\rangle|^2 = -\cos 2\theta_j. \tag{7}$$

From Eqs. 7 and 6, we can see that the value of $|\langle p_j|t\rangle|^2$ is encoded in the amplitude of the output state $|\phi_j\rangle$ of swap test. This will be used in the amplitude encoding of QNN cost function which is a crucial component of the quantum training.

Applying the Schmidt decomposition method to state $|\phi_j\rangle$, we arrive at

$$|\phi_j\rangle = \frac{-i}{\sqrt{2}}(e^{i\theta_j}|w_+\rangle_j - e^{-i\theta_j}|w_-\rangle_j), \tag{8}$$

where $|w_\pm\rangle_j = \frac{1}{\sqrt{2}}(|0\rangle|u_j\rangle \pm i|1\rangle|v_j\rangle)$.

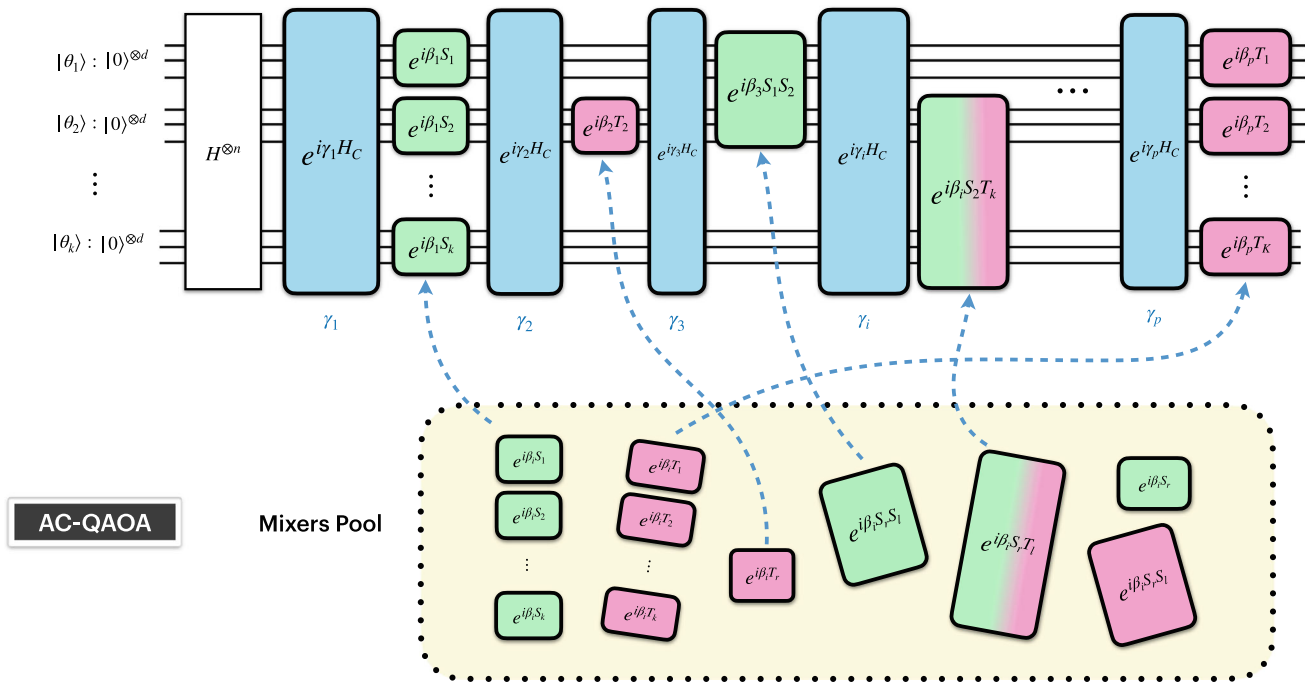


Fig. 10 Quantum circuit schematic of AC-QAOA. AC-QAOA is a variant of QAOA we designed for solving optimization of continuous variables with the short-depth advantage of QAOA layers. In this figure, θ_i are the continuous variables to be optimized in the training. Each θ_i is digitized into binary form and stored in an independent register. The overall process of AC-QAOA is similar to that of the original QAOA, with the difference being as follows. *1.* The mixers of AC-QAOA with Hamiltonians S_i and T_i are acting on the registers of θ_i (rather than single qubits as in the original QAOA). *2.* The mixers of AC-QAOA contain alternative mixers taken from a *mixers pool* and can vary from layer to layer

One can construct a Grover operator using U_j as follows:

$$G_j := (I^{\otimes(2n+1)} - 2|\phi_j\rangle\langle\phi_j|)(Z \otimes I^{\otimes 2n}), \tag{9}$$

$$= U_j C U_j^\dagger (Z \otimes I^{\otimes 2n}), \tag{10}$$

where $Z = |0\rangle\langle 0| - |1\rangle\langle 1|$ is the Pauli-Z operator; $C = I^{\otimes(2n+1)} - 2|0\rangle^{\otimes(2n+1)}\langle 0|^{\otimes(2n+1)}$ can be implemented as the circuit shown in Fig. 12. The circuit representation of G_j is shown in Fig. 13.

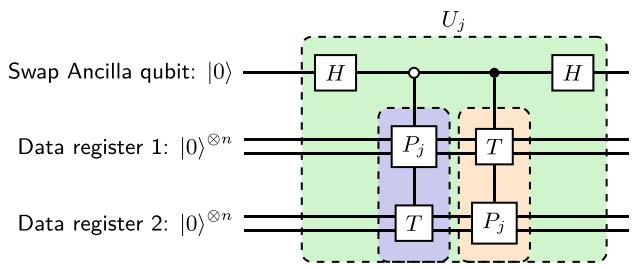


Fig. 11 Circuit diagram of swap test. Here, we present an alternative form of swap test: instead of applying the swap operation on two quantum states, the circuit in this figure simulates the "swap" effect by applying two unitaries P_j, T on two registers in different order controlled by an ancilla qubit. The "anti-control" symbol is defined as: when the control qubit is in state $|0\rangle$, the unitary being controlled is executed; when the control qubit is in state $|1\rangle$, the unitary being controlled is not executed

It is easy to check that $|w_\pm\rangle_j$ are the eigenstates of G_j—that is,

$$G_j|w_\pm\rangle = e^{\pm i2\theta_j}|w_\pm\rangle_j. \tag{11}$$

Recall from Eq. 7 the value of $|\langle p_j|t\rangle|^2$ is encoded in the phase of the eigenvalue of G_j. This will be used in the phase encoding of QNN cost function which is a crucial component of the quantum training.

2.2.2 Hadamard test and its "Grover operator"

Similar to the swap test, the Hadamard test is a technique that can be used to estimate $\langle 0|P_j^\dagger T P_j|0\rangle$, for two unitary operators P_j and T (assuming T is Hermitian). The circuit of Hadamard test is shown in Fig. 14.

We denote the unitary of the Hadamard test circuit (the dotted green box in Fig. 14) as U_j' and the output state from U_j' as

$$|\phi_j'\rangle = \frac{1}{\sqrt{2}}(|+\rangle P_j|0\rangle + |-\rangle T P_j|0\rangle). \tag{12}$$

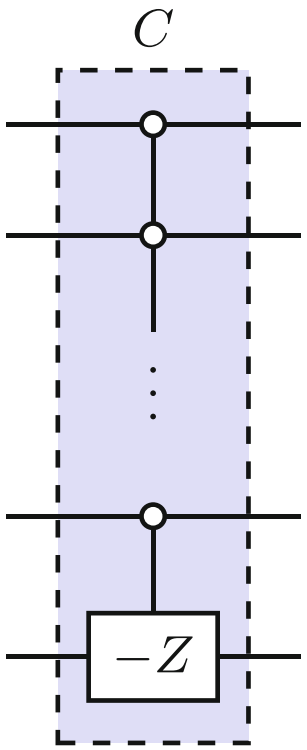


Fig. 12 Quantum circuit to implement unitary $C = I^{\otimes(2n+1)} - 2|0\rangle^{\otimes(2n+1)}\langle 0|^{\otimes(2n+1)}$

Rearranging the terms, we have

$$|\phi_j'\rangle = \frac{1}{2}(|0\rangle \otimes (P_j|0\rangle + TP_j|0\rangle)) + |1\rangle \otimes (P_j|0\rangle - TP_j|0\rangle)). \tag{13}$$

Denote $|u_j'\rangle$ and $|v_j'\rangle$ as the normalized states of $P_j|0\rangle + TP_j|0\rangle$ and $P_j|0\rangle - TP_j|0\rangle$, respectively. Then, there is a real number $\theta_j' \in [0, \pi/2]$ such that

$$|\phi_j'\rangle = \sin\theta_j'|0\rangle|u_j'\rangle + \cos\theta_j'|1\rangle|v_j'\rangle. \tag{14}$$

Fig. 13 Quantum circuit to implement G_j. G_j is defined as $G_j := U_j C U_j^\dagger (Z \otimes I^{\otimes 2n})$

θ_j' satisfies $\cos\theta_j' = \sqrt{1 - \langle 0|P_j^\dagger T P_j|0\rangle}/\sqrt{2}$, $\sin\theta_j' = \sqrt{1 + \langle 0|P_j^\dagger T P_j|0\rangle}/\sqrt{2}$. Therefore, we have

$$\langle 0|P_j^\dagger T P_j|0\rangle = -\cos 2\theta_j'. \tag{15}$$

We can define the Grover operator G_j' from U_j' in the same way as in last subsection for the swap test and obtain similar eigen-relation. The value of $\langle 0|P_j^\dagger T P_j|0\rangle$ is encoded in the phase of the eigenvalue of G_j'. This will be used in the phase encoding of QNN cost function which is a crucial component of the quantum training.

2.3 Creating superpositions of QNNs

As an essential building block for our quantum training protocol, we present a way to create superpositions of QNNs entangled with corresponding parameters. That is, we construct a controlled unitary $\mathcal{P}$ such that

$$P|\boldsymbol{\theta}\rangle|0\rangle \to |\boldsymbol{\theta}\rangle \otimes U(\boldsymbol{\theta})|0\rangle \quad \text{for every } \boldsymbol{\theta} \tag{16}$$

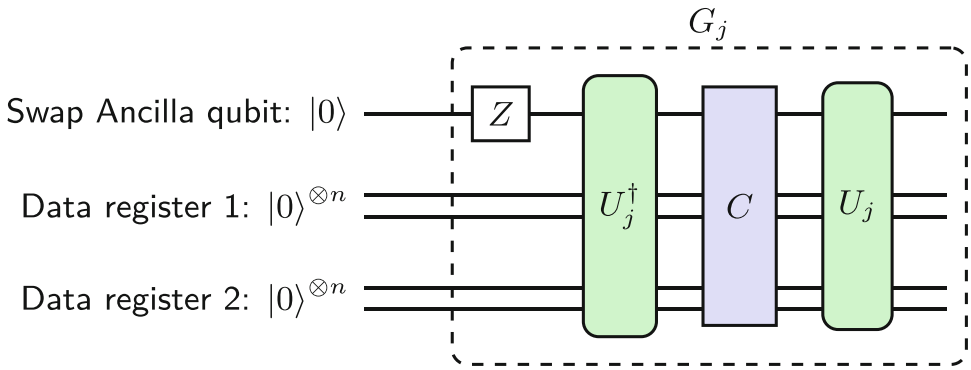
in which $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_M)$ is the set of trainable parameters in the QNN and $U(\boldsymbol{\theta})$ is the unitary of the QNN with corresponding parameters. When P acts on a superposition state of parameters $\sum_{\boldsymbol{\theta}} \omega_{\boldsymbol{\theta}}|\boldsymbol{\theta}\rangle$, we have

$$P\sum_{\boldsymbol{\theta}} \omega_{\boldsymbol{\theta}}|\boldsymbol{\theta}\rangle|0\rangle \to \sum_{\boldsymbol{\theta}} \omega_{\boldsymbol{\theta}}|\boldsymbol{\theta}\rangle \otimes U(\boldsymbol{\theta})|0\rangle. \tag{17}$$

The action of the controlled unitary P is depicted in Fig. 15.

This controlled unitary can be realized by dividing each rotation gate in QNN into a sequence of binary segments, followed by applying controlled operations on them. A simple example of one rotation gate, for example $U(\boldsymbol{\theta}) = R_z(\theta)$, is illustrated in Fig. 16.

Each bit string of the parameter register can be seen as a binary representation of the rotation angle, and the associated basis state of the register is entangled with the rotation gate of the corresponding angle. For instance, in the example above,



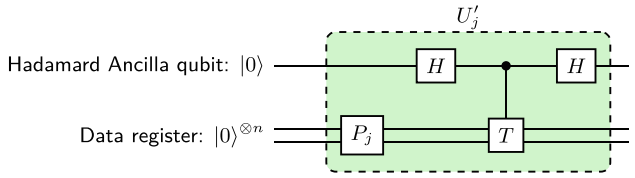


Fig. 14 Circuit diagram of Hadamard test. The circuit is used to estimate $\langle 0|P_j^\dagger T P_j|0\rangle$, for two unitary P_j and T. The Hadamard test will be used for the phase encoding of QNN cost function which is a crucial component of the quantum training

the bit string 111 corresponds to the angle $7\bar{\theta}/8$ and $|111\rangle$ is associated with $R_z(7\bar{\theta}/8)$, where $\bar{\theta}$ is the maximum value that angle θ can take. This relation can be fully illustrated in Fig. 17, in which we take $\bar{\theta} = \pi$.

The unitary operator of P can be written as

$$P = \sum_j |j\rangle\langle j| \otimes P_j, \tag{18}$$

in which P_j is a specific configuration of the QNN defined by its control bit string j. This representation does not only apply to a single rotation gate, but also to the case where there are multiple parameterized rotation gates in the QNN. An example of two rotation gates is depicted in Fig. 18.

In order to achieve precision ϵ_0 for each rotation angle, the number of control qubits needed is $d = \lceil \log_2(1/\epsilon_0) \rceil$. Let r be the number of rotation gates in a QNN, then the total number of control qubits needed is dr.

3 QNN training by Grover adaptive search

In this section, we discuss using Grover adaptive search to perform global optimization of QNNs. As presented in Section 2.1.3, the core of the Grover adaptive search is the adaptive oracle defined in Eq. 2. Next, we detail how to construct such oracle for QNN training.

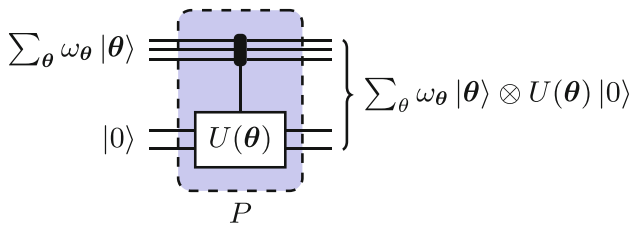


Fig. 15 Action of the controlled unitary P. In this figure, the upper register is parameter register and the lower register is the QNN register. $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_M)$ is the set of trainable parameters in the QNN, and $U(\boldsymbol{\theta})$ is the unitary of the QNN with corresponding parameters. The qubits in the parameter register act as control qubits on the rotation gates in the QNN. The controlled operations (in the dotted blue box) are denoted as P. When P acts on a superposition state of parameters $\sum_{\boldsymbol{\theta}} \omega_{\boldsymbol{\theta}}|\boldsymbol{\theta}\rangle$, the output state is $\sum_{\boldsymbol{\theta}} \omega_{\boldsymbol{\theta}}|\boldsymbol{\theta}\rangle \otimes U(\boldsymbol{\theta})|0\rangle$ in which the parameter register and QNN register are entangled

3.1 Construction of the Grover oracle

The adaptive Grover oracle $\mathcal{O}_{\text{Grover}}$ in the context of QNN training acts as

$$\mathcal{O}_{\text{Grover}}|\boldsymbol{\theta}\rangle \otimes |\mathbf{0}\rangle_{\text{QNN+ancillas}} = (-1)^{g(C(\boldsymbol{\theta})-C^*)}|\boldsymbol{\theta}\rangle \otimes |\mathbf{0}\rangle_{\text{QNN+ancillas}}, \quad \text{for every } \boldsymbol{\theta}, \tag{19}$$

in which C^* is the adaptive threshold for the cost function and the function g is defined as

$$g(x) = \begin{cases} 1 & x < 0 \\ 0 & \text{otherwise} \end{cases}. \tag{20}$$

When $\mathcal{O}_{\text{Grover}}$ is acting on a superposition state of parameters $\sum_{\boldsymbol{\theta}} \omega_{\boldsymbol{\theta}}|\boldsymbol{\theta}\rangle$, we have

$$\mathcal{O}_{\text{Grover}} \sum_{\boldsymbol{\theta}} \omega_{\boldsymbol{\theta}}|\boldsymbol{\theta}\rangle \otimes |\mathbf{0}\rangle_{\text{QNN+ancillas}} = \sum_{\boldsymbol{\theta}} (-1)^{g(C(\boldsymbol{\theta})-C^*)} \omega_{\boldsymbol{\theta}}|\boldsymbol{\theta}\rangle \otimes |\mathbf{0}\rangle_{\text{QNN+ancillas}}. \tag{21}$$

The QNN Grover oracle $\mathcal{O}_{\text{Grover}}$ can be constructed by the following steps.

3.1.1 Amplitude encoding

The first step is to encode the cost function of QNN into amplitude. Depending on the form of the cost function of the QNN, the amplitude encoding can be achieved by the swap test or Hadamard test. The correspondences are summarized in Table 2.

Amplitude encoding by swap test For the task of learning a pure state $|\psi\rangle = T|0\rangle$ (T is a given unitary), the cost function is the fidelity between the generated state from the QNN and the state $|\psi\rangle = T|0\rangle$. In this case, the amplitude encoding can be achieved by swap test, as shown in the circuit in Fig. 19.

We denote the unitary for the swap test circuit (in dotted green box) as U and the input and output state of U as $|\Psi_0\rangle$ and $|\Psi_1\rangle$, respectively. The input to U, $|\Psi_0\rangle$, can be written as (note here and throughout the paper, we omit the normalization factor):

$$|\Psi_0\rangle = |0\rangle \otimes (\sum_j |j\rangle) \otimes |0\rangle^n_{\text{QNN1}} |0\rangle^n_{\text{QNN2}} \tag{22}$$

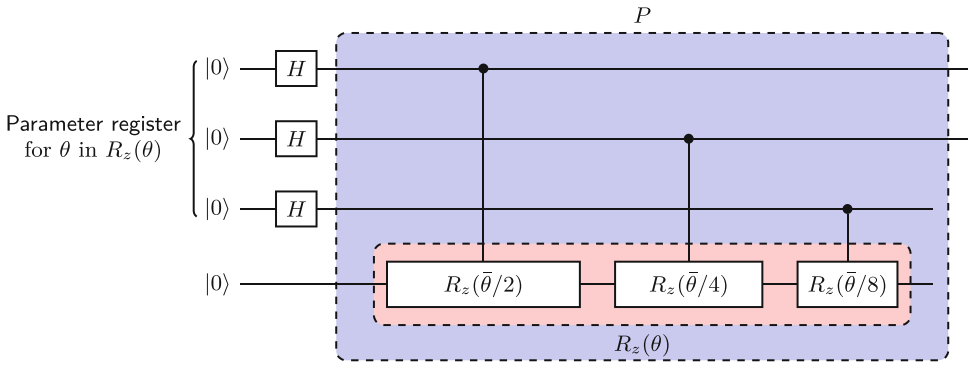


Fig. 16 An example of the construction of P for one rotation gate $R_z(\theta)$. In this example, the parameter register consists of three qubits, each qubit controls a "partial" rotation on the fourth qubit. The "partial" rotations are the binary segments $R_z(\bar{\theta}/2)$,$R_z(\bar{\theta}/4)$,$R_z(\bar{\theta}/8)$ in which $\bar{\theta}$ is the maximum value that angle θ can take

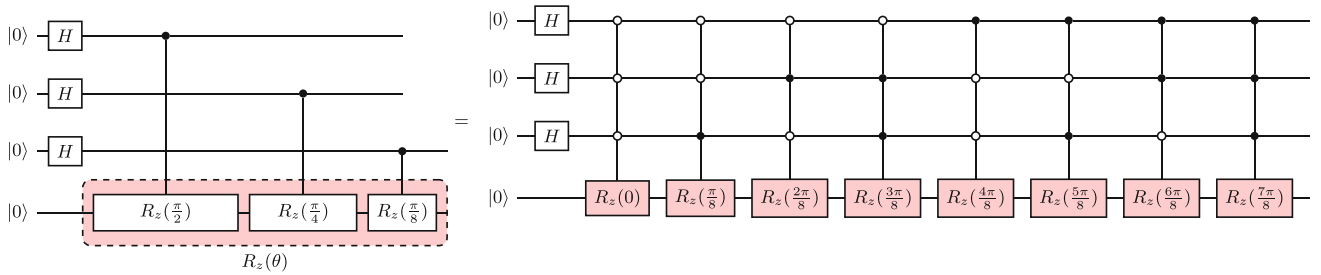


Fig. 17 An example of the effect of P defined in Fig. 16. Each bit string of the parameter register can be seen as a binary representation of the rotation angle, and the associated basis state of the register is entangled with the rotation gate of the corresponding angle. For instance, in the example above, the bit string 111 corresponds to the angle $7\bar{\theta}/8$ and $|111\rangle$ is associated with $R_z(7\bar{\theta}/8)$

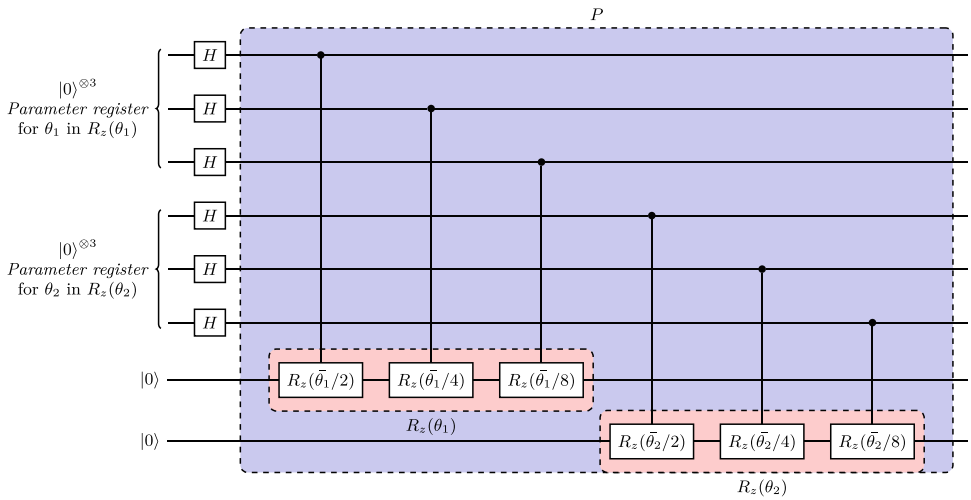


Fig. 18 Example of the construction of P for QNN consisting of two rotation gates. In this example, the QNN consists of two rotation gates $R_z(\theta_1)$, $R_z(\theta_2)$ on the lower two qubits. The upper 6 qubits are divided into two parameter registers for the two rotation angles θ_1, θ_2 respectively. Each qubit controls a "partial" rotation. For instance, the "partial" rotations of $R_z(\theta_1)$ are the binary segments $R_z(\bar{\theta_1}/2)$, $R_z(\bar{\theta_1}/4)$, $R_z(\bar{\theta_1}/8)$ in which $\bar{\theta_1}$ is the maximum value that angle θ_1 can take

Table 2 QNN cost functions for two type of tasks

Task	Cost function		Amplitude encoding method
Generating ground state of T	Expectation value	$C(\boldsymbol{\theta}) = \langle 0\|U^\dagger(\boldsymbol{\theta})TU(\boldsymbol{\theta})\|0\rangle$	***Hadamard test***
Generating a pure state $\|\psi\rangle = T\|0\rangle$ (T is a given unitary)	Fidelity	$C(\boldsymbol{\theta}) = \|\langle 0\|U^\dagger(\boldsymbol{\theta})T\|0\rangle\|^2$	***Swap test***

Here, we present the cost functions for two tasks, respectively: For the task of generating ground state of some given Hamiltonian T (we use T instead of H here and assume T is Hermitian), the cost function is chosen to be the expectation value of T. For the task of generating a pure state $|\psi\rangle = T|0\rangle$ (T is a given unitary), the cost function is chosen to be the fidelity between the generated state from QNN and the state $|\psi\rangle = T|0\rangle$

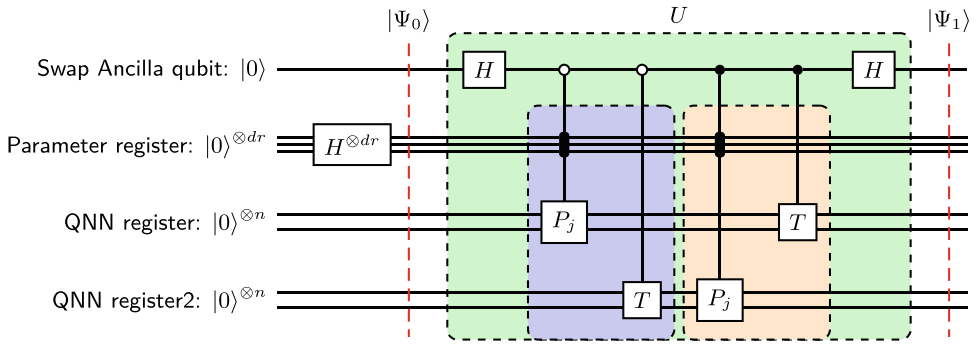


Fig. 19 Amplitude encoding by swap test. This circuit can perform the swap test depicted in Fig. 11 in parallel for multiple P_j. Here, P_j represents QNN with specific (the "jth") parameter configuration. To achieve swap test in parallel, we add an extra register—the parameter register—as the control of P_j: each computational basis j of the parameter register corresponds to a specific parameter configuration in P_j. As illustrated in Fig. 15, once the parameter register is in superposition state (by the Hadamard gates $H^{\otimes dr}$), the corresponding P_j are in superposition. We refer the control operation on QNN as "controlled-QNN." Comparing with the normal swap test depicted in Fig. 11, the difference here is that the swap ancilla qubit is anti-controlling/controlling the "controlled-QNN" together with the unitary T (as gathered together in the dotted blue/orange box). It can be proven that the entire circuit in dotted the green box (denoted as U) can be expressed as $U = \sum_j |j\rangle\langle j| \otimes U_j$ where U_j is the swap test unitary for P_j defined in Fig. 11. This indicates that U effectively performs the swap test in parallel for multiple P_j. Recall the fact that the normal swap test U_j encodes $|\langle p_j|t\rangle|^2$ in the amplitude of the output state (Eq. 7 and Eq. 6); here, the "parallel swap test" U encodes the QNN cost function $|\langle p_j|t\rangle|^2$ in the amplitude of a superposition of P_j(QNN) with different parameters

Then, U can be written explicitly as

$$U := [H \otimes I \otimes I \otimes I] \cdot [|0\rangle\langle 0| \otimes (\sum_j |j\rangle\langle j| \otimes P_j \otimes T) + |1\rangle\langle 1| \otimes (\sum_j |j\rangle\langle j| \otimes T \otimes P_j)] \cdot [H \otimes I \otimes I \otimes I], \quad (23)$$

Here, P_j represents QNN with specific parameter configuration defined by its control bit string j, as defined in Eq. 18. It can be proven (see Appendix A) that U can be rewritten as

$$U = \sum_j |j\rangle\langle j| \otimes U_j, \quad (24)$$

where U_j is the *individual swap test unitary* on unitary P_j and target unitary T, defined as in Eq. 3:

$$U_j := [H \otimes I \otimes I] \cdot [|0\rangle\langle 0| \otimes P_j \otimes T + |1\rangle\langle 1| \otimes T \otimes P_j] \cdot [H \otimes I \otimes I]. \quad (25)$$

As in Eq. 6, the resulting state of U_j acting on $|\Psi_0\rangle$ is $|\phi_j\rangle := U_j|0\rangle\,|0\rangle^n_{\text{QNN1}}\,|0\rangle^n_{\text{QNN2}}$ and has the following form:

$$|\phi_j\rangle = \sin\theta_j|u_j\rangle|0\rangle + \cos\theta_j|v_j\rangle|1\rangle. \quad (26)$$

The final output state from U, $|\Psi_1\rangle = U\,|\Psi_0\rangle$, is therefore

$$|\Psi_1\rangle = \sum_j |j\rangle \underbrace{(\sin\theta_j\,|u_j\rangle|0\rangle + \cos\theta_j\,|v_j\rangle|1\rangle)}_{|\phi_j\rangle} = \sum_j |j\rangle\,|\phi_j\rangle \quad (27)$$

From Eqs. 27 and 7, we can see that the cost function (fidelity $|\langle p_j|t\rangle|^2$) for different parameters has been encoded into the amplitudes of the state $|\Psi_1\rangle$.

Amplitude encoding by Hadamard test For the task of generating ground states of given Hamiltonian T, the cost function is the expectation value of T with respect to the generated state from the QNN. In this case, the amplitude encoding can be achieved by the Hadamard test, as shown in the circuit in Fig. 20.

Since the analysis for the case of the Hadamard test is very similar to that of the swap test, we omit the details here. For the same reason, we only present the case using the swap test also in the next section when discussing phase encoding.

3.1.2 Amplitude estimation

The second step following the amplitude encoding is to use amplitude estimation (Brassard et al. 2000) to extract and store the cost function into an additional register which we call the "amplitude register." In the following, we present the details of amplitude estimation.

After the amplitude encoding by the swap test, we introduce an extra register $|0\rangle^t_{\text{amplitude}}$, and the output state $|\Psi_1\rangle$ (using the same notation) becomes

$$|\Psi_1\rangle = \sum_j |j\rangle\,|\phi_j\rangle|0\rangle^t_{\text{amplitude}}, \quad (28)$$

where $|\phi_j\rangle$ can be decomposed as

$$|\phi_j\rangle = \frac{-i}{\sqrt{2}}\left(e^{i\theta_j}|\omega_+\rangle_j - e^{i(-\theta_j)}|\omega_-\rangle_j\right). \quad (29)$$

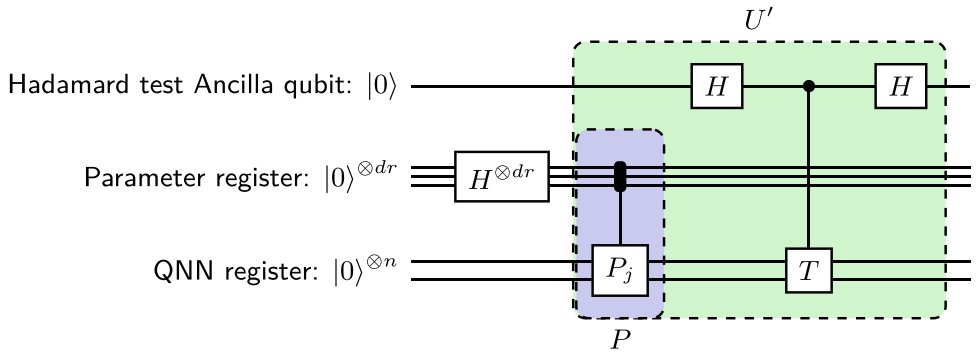


Fig. 20 Amplitude encoding by Hadamard test. This circuit can perform the Hadamard test depicted in Fig. 14 in parallel for multiple P_j. Here, P_j represents QNN with specific (the "jth") parameter configuration. To achieve Hadamard test in parallel, we add an extra register—the parameter register—as the control of P_j: each computational basis j of the parameter register corresponds to a specific parameter configuration in P_j. As illustrated in Fig. 15, once the parameter register is in superposition state (by the Hadamard gates $H^{\otimes dr}$), the corresponding P_j are in superposition. It can be proven that the entire circuit in dotted the green box (denoted as U) can be expressed as $U' = \sum_j |j\rangle\langle j| \otimes U'_j$ where U'_j is the Hadamard test unitary for P_j defined in Fig. 14. This indicates that U' effectively performs the swap test in parallel for multiple P_j. Recall the fact that the normal Hadamard test U'_j encodes $\langle 0|P_j^\dagger T P_j|0\rangle$ in the amplitude of the output state (Eq. 14 and Eq. 15); here, the "parallel Hadamard test" U' encodes the QNN cost function $\langle 0|P_j^\dagger T P_j|0\rangle$ in the amplitude of a superposition of P_j(QNN) with different parameters

Hence, we have

$$|\Psi_1\rangle = \sum_j \frac{-i}{\sqrt{2}} \left(e^{i\theta_j} |j\rangle |\omega_+\rangle_j - e^{i(-\theta_j)} |j\rangle |\omega_-\rangle_j \right) |0\rangle^t_{\text{amplitude}}. \tag{30}$$

The overall Grover operator G is defined as

$$G := UC_2U^{-1}C_1, \tag{31}$$

where C_1 is the Z gate on the swap ancilla qubit and C_2 is "flip zero state" unitary which is similar to that defined in Fig. 12. It can be shown (see Appendix A) that G can be expressed as

$$G = \sum_j |j\rangle\langle j| \otimes G_j, \tag{32}$$

where G_j is the individual Grover operator as defined in Eq. 10. The overall Grover operator G possesses the following eigen-relation:

$$G |j\rangle |\omega_\pm\rangle_j = e^{i(\pm 2\theta_j)} |j\rangle |\omega_\pm\rangle_j. \tag{33}$$

Next, we apply phase estimation of the overall Grover operator G on the input state $|\Psi_1\rangle$. The resulting state $|\Psi_2\rangle$ can be written as

$$|\Psi_2\rangle = \sum_j \frac{-i}{\sqrt{2}} \left(e^{i\theta_j} |j\rangle |\omega_+\rangle_j |2\theta_j\rangle - e^{i(-\theta_j)} |j\rangle |\omega_-\rangle_j |-2\theta_j\rangle \right). \tag{34}$$

Note here in Eq. 34, $|\pm 2\theta_j\rangle$ denotes the eigenvalues $\pm 2\theta_j$ being stored in the amplitude register with some finite precision.

3.1.3 Threshold oracle and uncomputations

Next, we apply a threshold oracle U_O on the amplitude register and an extra phase ancilla qubit, which acts as

$$U_O|\pm 2\theta_j\rangle = (-1)^{g(\theta_j - \theta^*)}|\pm 2\theta_j\rangle, \tag{35}$$

where θ^* is implicitly defined as

$$C^* = -\cos 2\theta^*, \quad \theta^* \in [\pi/4, \pi/2]. \tag{36}$$

Note that in Eq. 35, we omit the state of the phase ancilla qubit.

The state after the oracle $|\Psi_3\rangle$ can be written as

$$|\Psi_3\rangle = \sum_j \frac{-i}{\sqrt{2}} (-1)^{g(\theta_j - \theta^*)} \Big(e^{i\theta_j} |j\rangle |\omega_+\rangle_j |2\theta_j\rangle - e^{i(-\theta_j)} |j\rangle |\omega_-\rangle_j |-2\theta_j\rangle \Big), \tag{37}$$

The procedure thus far can be illustrated in a circuit as in Fig. 21.

After we perform the uncomputation of phase estimation, the resulting state is

$$|\Psi_4\rangle = \sum_j \frac{-i}{\sqrt{2}} (-1)^{g(\theta_j - \theta^*)} \left(e^{i\theta_j} |j\rangle |\omega_+\rangle_j |0\rangle^t_{\text{amplitude}} \right.$$

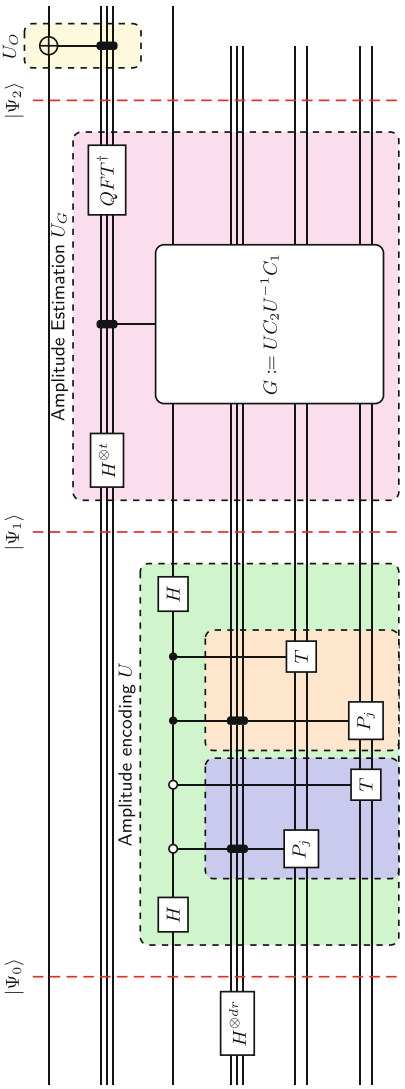


Fig. 21 *Major steps in the construction of the Grover oracle.* Step 0: We initialize the system by applying Hadamard gates on the parameter register, leading to the state $|\Psi_0\rangle = |0\rangle_s \otimes (\sum_j |j\rangle) \otimes |0\rangle^n_{\text{QNN1}} |0\rangle^n_{\text{QNN2}}$. Step 1 (dotted green box): Amplitude encoding of the cost function, as illustrated in Fig. 19 (refer the caption of Fig. 19 for the meaning of each symbol), resulting in the state $|\Psi_1\rangle = \sum_j |j\rangle (\sin\theta_j |u_j\rangle|0\rangle + \cos\theta_j |v_j\rangle|1\rangle)$, in which θ_i contains the cost function. Step 2 (dotted pink box): Amplitude estimation to extract and store the cost function into an additional register which we call the "amplitude register," resulting in the state $|\Psi_2\rangle = \sum_j \frac{-i}{2} \left(e^{i\theta_j} |j\rangle |\omega_+\rangle_j |2\theta_j\rangle - e^{i(-\theta_j)} |j\rangle |\omega_-\rangle_j |-2\theta_j\rangle\right)$. Step 3 (dotted yellow box): Threshold oracle to encode the cost function into relative phase by using a phase ancilla qubit, resulting in the state $|\Psi_3\rangle = \sum_j \frac{-i}{2} (-1)^{g(\theta_j - \theta^*)} \left(e^{i\theta_j} |j\rangle |\omega_+\rangle_j |2\theta_j\rangle - e^{i(-\theta_j)} |j\rangle |\omega_-\rangle_j |-2\theta_j\rangle\right)$

$$-e^{i(-\theta_j)}\,|j\rangle\,|\omega_-\rangle_j|0\rangle^t_{\text{amplitude}}\Big), \tag{38}$$

$$=\sum_j(-1)^{g(\theta_j-\theta^*)}\,|j\rangle\,|\phi_j\rangle|0\rangle^t_{\text{amplitude}}. \tag{39}$$

Finally, we perform the uncomputation of the swap test, and the resulting state is

$$|\Psi_5\rangle=\sum_j(-1)^{g(\theta_j-\theta^*)}\,|j\rangle\,|0\rangle^n_{\text{QNN1}}\,|0\rangle^n_{\text{QNN2}}\,|0\rangle|0\rangle^t_{\text{amplitude}}. \tag{40}$$

As can be seen from Eqs. 40 and 7, the above steps implemented the Grover oracle $\mathcal{O}_{\text{Grover}}$ (defined in Eq. 19). After the above procedure, a relative phase, which depends on the cost function of the QNN $|\langle p_j|t\rangle|^2$ and the threshold, has been coherently added to the parameter state. Importantly, uncomputation allows the parameter register to be decoupled from the QNN and other registers.

3.2 Performance of the quantum training by Grover adaptive search

Taking training VQE as an example, in Table 3, we present the result for the number of "controlled-QNN" runs, the number of QNN runs, and the number of measurements needed in the quantum training by Grover adaptive search. The derivation is included in Appendix B.

3.3 Advantages and disadvantages of training by Grover adaptive search

In the presence of a noise-free barren plateau, the Grover adaptive search mechanism can find global optima without an exponential number of measurements. However, it has the following disadvantages:

- It can be seen from Table 3 that in quantum training by Grover adaptive search, the number of "controlled-QNN" runs is exponential in the number of parameters in QNN. Even in the case where the number of parameters scales only linearly with the number of qubits in a QNN, the quantum training by Grover takes excessive runtime. Moreover, it invokes very deep circuit.
- Training by Grover adaptive search does not circumvent the noise-induced barren plateau. When the entire cost landscape is flatten in the case of noise-induced barren plateau (Wang et al. 2020), it requires exponential precision of the amplitude estimation. That is, ϵ_2 should be exponentially small. According to Table 3, this adds another exponentially large factor to the number of "controlled-QNN" runs and QNN runs.

While these disadvantages most probably rule out Grover adaptive search for NISQ-era devices, it still represents a maximally quantum solution. For fault-tolerant devices, this method is the provably optimal approach for QNN cost function with no structure; it enjoys a quadratic speed-up which is a significant improvement compare to the exponential "slowdown" of the classical training methods due to the barren plateau issue.

4 QNN training by adaptive QAOA

As depicted in Fig. 3, our framework for quantum training of QNNs consists of two major components.

- **Phase oracle.** This coherently encodes the cost function of QNNs onto a relative phase of a superposition state in the Hilbert space of the parameters (Gilyén et al. 2019a).
- **Adaptive mixers.** These exploit hidden structure in QNN optimization problems, hence can achieve short-depth circuit (McClean et al. 2020).

Iterations of the phase oracle and the adaptive mixers constitute a QAOA routine which quantumly homing in on optimal network parameters of QNNs. This section presents the details of our framework.

4.1 Phase oracle

We aim to coherently achieve the phase encoding for the cost function of the QNN by a phase oracle O_{Phase}, which acts as

$$O_{\text{Phase}}|\boldsymbol{\theta}\rangle|\mathbf{0}\rangle_{\text{QNN+ancillas}}\rightarrow e^{-i\gamma C(\boldsymbol{\theta})}|\boldsymbol{\theta}\rangle|\mathbf{0}\rangle_{\text{QNN+ancillas}} \tag{41}$$

Table 3 Performance of the quantum training by Grover adaptive search

Number of "controlled-QNN" runs	Number of QNN runs	Number of measurements
$O\left(\frac{1}{\epsilon_2\epsilon_1}\left(\frac{1}{\epsilon_0}\right)^{r/2}s^{-0.5}\right)$	$O\left(\frac{1}{\epsilon_2}\left(r\log\left(\frac{1}{\epsilon_0}\right)\right)^{1.5}\right)$	$O\left(\left(r\log\left(\frac{1}{\epsilon_0}\right)\right)^{1.5}\right)$

Here, we present the result for the number of "controlled-QNN" runs, the number of QNN runs, and the number of measurements needed in the quantum training by Grover adaptive search. In this table, r is the number of parameters (rotation angles) in QNN, $1-\epsilon_1$ is the probability of success of the phase estimation, ϵ_2 is the precision we set up for the evaluation of the cost function using amplitude estimation, ϵ_0 is the precision of each angle value, s is the number of global optima of the QNN cost function

in which γ is a free parameter to be optimized. When O_{Phase} is acting on a superposition state of parameters $\sum_{\boldsymbol{\theta}} \omega_{\boldsymbol{\theta}} |\boldsymbol{\theta}\rangle$, we have

$$O_{\text{Phase}} \sum_{\boldsymbol{\theta}} \omega_{\boldsymbol{\theta}} |\boldsymbol{\theta}\rangle |\mathbf{0}\rangle_{\text{QNN+ancillas}} \rightarrow \sum_{\boldsymbol{\theta}} e^{-i\gamma C(\boldsymbol{\theta})} \omega_{\boldsymbol{\theta}} |\boldsymbol{\theta}\rangle |\mathbf{0}\rangle_{\text{QNN+ancillas}} \tag{42}$$

As detailed in Ref. (Gilyén et al. 2019a), this phase oracle can be constructed based on the amplitude encoding which we have implemented in Section 3.1.1. Next, we present the details of how to construct the phase oracle from the amplitude encoding by amplitude estimation or linear combination of unitaries (LCU) (Berry et al. 2015).

Phase oracle by amplitude estimation The procedure to achieve O_{Phase} by amplitude estimation is very similar to that of $\mathcal{O}_{\text{Grover}}$; the only difference is that the threshold U_O (defined in Eq. 35) needs to be replaced by U'_O which acts as

$$U'_O |\pm 2\theta_j\rangle = e^{-i\gamma C(\boldsymbol{\theta})} |\pm 2\theta_j\rangle. \tag{43}$$

Recall Eqs. 7 and 15 and the form of the cost function in Table 2, the cost function $C(\boldsymbol{\theta})$ is encoded in θ_j as $C(\boldsymbol{\theta}) = -\cos 2\theta_j$; therefore, U'_O acts as

$$U'_O |\pm 2\theta_j\rangle = e^{i\gamma \cos 2\theta_j} |\pm 2\theta_j\rangle. \tag{44}$$

Once we have chosen the specific value of γ, U'_O can be constructed according to Eq. 44.

Phase oracle by LCU For this approach, we start with constructing an operator G^* defined similar as in Eq. 31[1]:

$$G^* := C_2 U^{-1} C_1 U, \tag{45}$$

It has been shown in Ref. (Gilyén et al. 2019a) that

$$e^{-i\frac{1}{2}(C(\boldsymbol{\theta})-1)} \cdot I \approx \sum_{m=-M}^{M} \beta_m G^{*m}, \tag{46}$$

where $\beta_m = \sum_{k=|m|}^{M} \binom{2k}{k-m} \frac{(-1)^m i^k}{k! 2^{2k}}$, $M \in \mathbb{N}_+$.

Define a new cost function $C'(\boldsymbol{\theta}) := \frac{1}{2}(C(\boldsymbol{\theta}) - 1)$ (optimizing $C'(\boldsymbol{\theta})$ is equivalent to optimizing $C(\boldsymbol{\theta})$), we have

$$e^{-iC'(\boldsymbol{\theta})} \cdot I \approx \sum_{m=-M}^{M} \beta_m G^{*m}. \tag{47}$$

[1] Note that here, our definition of G^* is slightly different from the G_U in Ref. (Gilyén et al. 2019a): C_1 and C_2 being negative to their counterpart in the definition of G_U. However, the two negative signs cancel, therefore we have $G^* = G_U$.

This series of G^* can be implemented using the LCU technique (together with the subsequent "oblivious amplitude amplification") (Berry et al. 2015) in which the number of calls to G^* needed is only logarithmic of the inverse of the desired precision (Gilyén et al. 2019a). Using the techniques in Ref. (Gilyén et al. 2019b), we can convert phase oracle with $e^{-iC'(\boldsymbol{\theta})}$ into phase oracle with $e^{-i\gamma C'(\boldsymbol{\theta})}$ for arbitrary γ bounded from $[-1, 1]$), by only logarithmic (of the inverse of the desired precision) number of queries of phase oracle with $e^{-iC'(\boldsymbol{\theta})}$.

In Fig. 22, we summarize the two approaches for the phase encoding of the cost function.

4.2 Adaptive mixers

As in Section 2.1.2, we designed a new variant of QAOA—"Adaptive-Continuous (AC-QAOA)"—to be the ansatz of our quantum training for QNN. We summarize the reason of this choice as follows:

1. **[Why "continuous"]** In our optimization problem of QNN training, the parameters we are optimizing (the angles of rotation gates) are continuous variables (real values); hence, the choice of mixer Hamiltonian has to be designed for continuous variables. For example, the mixer Hamiltonian of the original QAOA (X rotations) generate shifts in the computational basis; here in continuous case, the corresponding mixer should shift the value for each digitized continuous variables stored in independent registers.
2. **[Why "adaptive"]** The cost function of QNNs is complicated and task-specific (given by the learning objectives). Hence, it is hard to analytically determine good mixers for our optimization problem of QNN training. Therefore, we would want to take advantage of including alternative mixers and allowing adaptive mixers for different layer (as in ADAPT-QAOA).

Adopting "AC-QAOA" could exploit hidden structure in QNN optimization problem and dramatically shorten the depth of QAOA layers while significantly improving the quality of the solution (McClean et al. 2020).

Generally, the mixer pool of AC-QAOA should include two types of mixer Hamiltonians for continuous variables:

1. Quadratic functions of the position operator and the momentum operator for single continuous variables, e.g., the squeezing operator (Weedbrook et al. 2012).
2. Entangling mixers that acts on two continuous variables, e.g., the two-mode squeezing operator (Weedbrook et al. 2012).

These operators could be carried out in continuous variable quantum systems. However, we will focus on the circuit

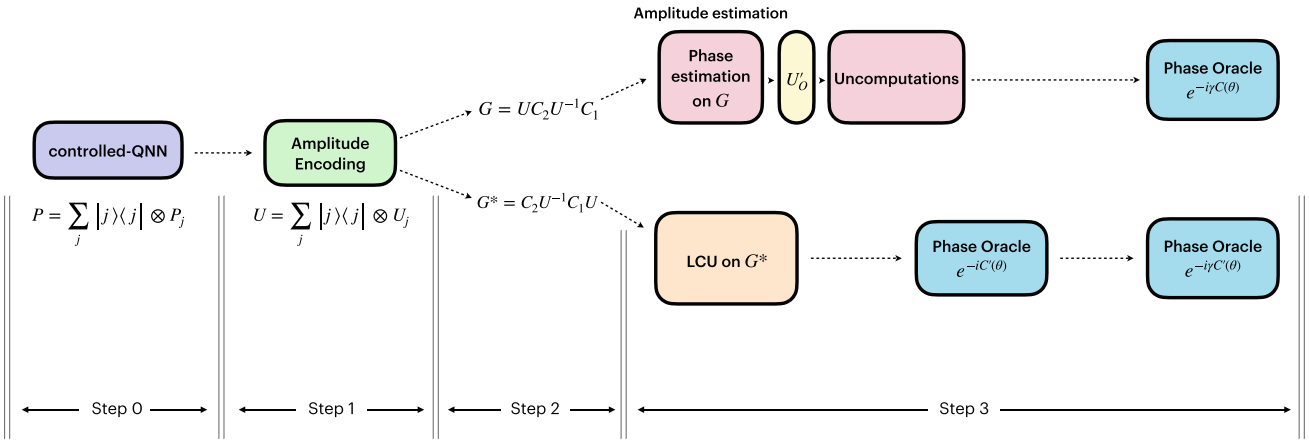


Fig. 22 Pipeline of the construction of the phase oracle. Here, we summarize the two approaches by amplitude estimation and by LCU for the phase encoding of the cost function. Step 0: Creating superposition for QNN with different parameters, which is implemented by "controlled-QNN" (see Fig. 11), denoted by $P = \sum_j |j\rangle\langle j| \otimes P_j$. Step 1: Amplitude encoding of the cost function, by the unitary operation $U = \sum_j |j\rangle\langle j| \otimes U_j$. Step 2: Constructing the "Grover operator" upon the amplitude encoding unitary. In the approach using amplitude estimation, the Grover operator G is constructed as $G = UC_2U^{-1}C_1$. In the approach using LCU, the Grover operator G^* is constructed as $G^* = C_2U^{-1}C_1U$. Step 3: Phase encoding of the cost function, by amplitude estimation (upper path) or by LCU (lower path). In the upper path, the phase oracle is achieved by phase estimation on G, threshold oracle U'_O, and uncomputation. In the lower path, LCU on G^* (together with the subsequent "oblivious amplitude amplification") (Berry et al. 2015) realizes $e^{-iC'(\boldsymbol{\theta})}$ which is then converted to the phase oracle with arbitrary γ —$e^{i\gamma C'(\boldsymbol{\theta})}$ using the method in Ref. (Gilyén et al. 2019b). $C'(\boldsymbol{\theta}) := \frac{1}{2}(C(\boldsymbol{\theta}) - 1)$ is a new cost function; optimizing $C'(\boldsymbol{\theta})$ is equivalent to optimizing $C(\boldsymbol{\theta})$

implementation of these mixers when using a collection of qubits to approximate the behavior of continuous variables.

When using a qudit of dimension d to digitally simulate a continuous variable, the position operator can be written as

$$J_d := \sum_{j=0}^{d-1} j|j\rangle\langle j|, \tag{48}$$

in which j is the digitized value of the continuous variable.

We can use N qubits to simulate the qudit and construct J_d for $d = 2^N$ as Verdon et al. (2018),

$$J_{2^N} = \sum_{n=1}^{N} 2^{n-2}(I^{(n)} - Z^{(n)}), \tag{49}$$

where $I^{(n)}$ and $Z^{(n)}$ are the identity and the Pauli-Z operator (respectively) for the n^{th} qubit.

The momentum operator, which acts as generator of shifts in the value of a continuous variable (denote as S), can be written as the discrete Fourier transform of J_d (Verdon et al. 2018),

$$S := F_d^\dagger J_d F_d, \tag{50}$$

in which the discrete Fourier transform F_d is defined by

$$F_d|j\rangle := \frac{1}{\sqrt{d}} \sum_{k=0}^{d-1} \omega_d^{-jk}|k\rangle, \tag{51}$$

where $\omega_d := e^{2\pi i/d}$.

As mentioned above, a general mixer Hamiltonian is the quadratic functions of the position operator J_d and the momentum operator S; therefore, using Eqs. 49 and 50 (set $d = 2^N$), we can rewrite a mixer Hamiltonian as a summation of simple unitaries. Hence, utilizing the Hamiltonian simulation technique in Berry et al. (2015), the mixer operator can be efficiently implemented. For instance, the digitized version of the generator of squeezing operator (denote as T) is defined as

$$T := J_d S + S J_d. \tag{52}$$

Plugging Eq. 50 into Eq. 52, together with Eq. 49 (set $d = 2^N$), we can see that $T = J_d F_d^\dagger J_d F_d + F_d^\dagger J_d F_d J_d$ can be expressed as the summation of simple unitaries. Therefore, the corresponding mixer with Hamiltonian T can be efficiently implemented using the Hamiltonian simulation technique in Berry et al. (2015). Similarly, the entangling mixers on two continuous variables with Hamiltonian $S_i S_j$, $S_i T_j$, $T_i T_j$ (the subscript i, j indicates that they are for specific variables) can be implemented in the same manner. In Fig. 23, we depict the schematic diagram of applying AC-QAOA to QNN training.

Due to the fact that the non-Gaussian operators are costly to implement, we only consider up-to-second-order polynomial functions of the position operator J_d and the momentum operator S for the mixer Hamiltonian. The mixer pool can generally include mixers with Hamiltonians: J_d, S, $J_d S$, $S J_d$,

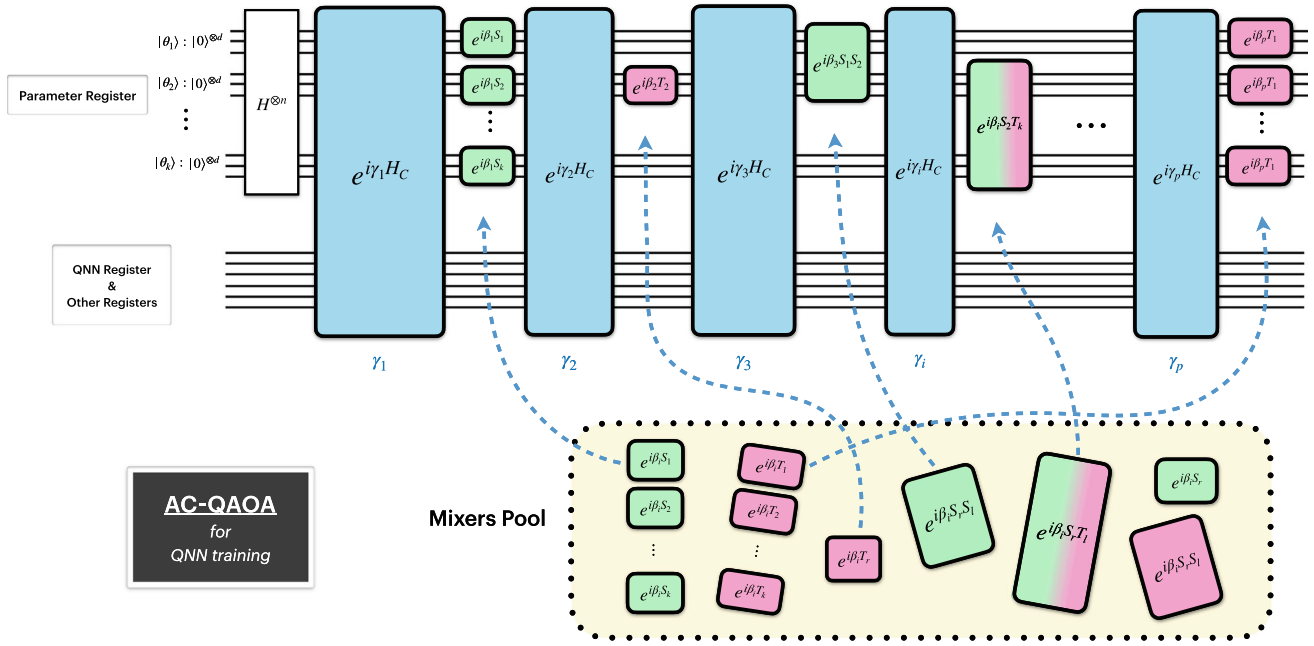


Fig. 23 Schematic diagram of applying AC-QAOA to QNN training. AC-QAOA is a variant of QAOA we designed for solving optimization of continuous variables with the short-depth advantage of QAOA layers, see Fig. 10. This figure illustrates applying AC-QAOA to QNN training, following the scheme in Fig. 3. The quantum training protocol consists of alternating operations in a QAOA fashion—the first operation acts on both the parameter register and QNN register to encode the cost function of QNN onto a relative phase of the parameter state. This operation is represented by the blue blocks in the figure. The other operations are the mixers (green and pink boxes) which act only on the parameter register. In the parameter register, θ_i are the continuous variables to be optimized in the training; each θ_i is digitized into binary form and stored in an independent register. The overall process of AC-QAOA is similar to that of the original QAOA, with the difference being as follows. *1*. The mixers of AC-QAOA with Hamiltonians S_i and T_i are acting on the registers of θ_i (rather than single qubits as in the original QAOA). *2*. The mixers of AC-QAOA contain alternative mixers taken from a *mixers pool* and can vary from layer to layer

$J_d{}^2$, S^2, $J_d{}^2 + S^2$ for one continuous variable and the entangling mixers for two continuous variables. Comparing to the mixer pool of ADAPT-QAOA for discrete variable, we can have the following the analogy:

1. The momentum operator S is the (digitized) continuous version of X mixers that shift the value for each digitized continuous variables stored in independent registers.
2. $J_d S$ is the (digitized) continuous version of Y mixers which "unlock" geodesics in parameter space, allowing the QAOA iterations reaching the target state faster Yao et al. (2020b).

We note that quadratic Hamiltonians are efficiently simulatable (classically), but only when the initial state is from a special class of Gaussian states (e.g., the vacuum state) (Bartlett et al. 2002). Here, the initial state in the qubit encoding is far from Gaussian, and a continuous variable analog of our technique would use an equivalent encoding.

By making the mixers flexible and adaptive to specific optimization problems, it is demanding to find an efficient way of determining the mixers sequence and optimizing the hyper-parameters. There are several research works on using machine learning approaches (recurrent neural networks (RNN) and reinforcement learning (RL)) to determine the mixers sequence and optimize the hyper-parameters. These works achieved significantly less measurements than conventional approach(e.g., gradient based methods). We list the papers in the following table:

	RNN	RL
Determining mixers sequence	Ref. (Warren et al 2020)	Ref. (Yao et al. 2020b)
Optimizing hyper-parameter	Ref. (Verdon et al. 2019b)	Refs. (Khairy et al. 2019; Wauters et al. 2020; Yao et al. 2020a, b)

We adopt the approaches developed in these works to our quantum training of QNNs for efficiently determining the mixers sequence and optimizing the hyper-parameters.

4.3 Advantages of training by QAOA

As we have discussed in Section 3.3, due to the global-search nature of Grover's algorithm, the quantum training using Grover adaptive search can circumvent the noise-free barren plateau; however, it has certain limitations and disadvantages such as: (1) cannot handle the noise-induced barren plateau; (2) requires an exponential number of calls to the "controlled-QNN" with excessive lengths of circuit and run time.

In contrast, our quantum training using adaptive continuous QAOA could eliminate the limitations of that using Grover adaptive search and the advantages come in the following two folds:

1. The phase oracle by LCU approach does not explicitly evaluate/store the value of the cost function at any stage of the algorithm and the number of calls to the "controlled-QNN" scales only logarithmic with respect to the inverse of the desired precision (Gilyén et al. 2019a). Therefore, the phase encoding is not affected by the noise-induced barren plateau for which the precision required is exponentially small. This is better than the case using Grover adaptive search.
2. The adaptive mixers can dramatically reduce the number of QAOA iterations while significantly increasing the quality of the output solution. This will enable our quantum training to achieve high performance within relatively shallow circuit and short run time. Thanks to the phase encoding faithfully conserving all the information and structure in the cost function, our adaptive QAOA protocol can exploit hidden structures in the QNN training problem. (Whereas the Grover oracle "cuts off" the cost function with the threshold effectively losing some information and structure in the cost function.) Therefore, adaptive QAOA can offer beyond-Grover speed-up. Moreover, numerical experiments in Yao et al. (2020b) show that when using the adaptive approach, the depth of the QAOA steps can be independent of the problem size (number of qubits); this would yield even more advantage when system size scales up.

5 Applications

In this section, we discuss several applications of QNN which our quantum training algorithm can apply to. For each application, we first briefly illustrate the usage of QNN and the corresponding cost function for the task, then we present the way of amplitude encoding tailored for this application. Based on the amplitude encoding, the construction of the full quantum training algorithm is similar for every application.

5.1 Training VQE

Variational quantum eigensolvers (VQEs) utilize QNN to estimate the eigenvalue corresponding to some eigenstate of a Hamiltonian. The most common instance is ground state estimation in which the QNN (a parameterized circuit ansatz) is applied to an initial state (e.g., the zero state) over multiple qubits to generate the ground state. The parameters in the QNN are optimized so that the generated state of the QNN possesses the lowest expectation value of the given Hamiltonian. A schematic of VQE for ground state estimation is presented in Fig. 24.

Consider a Hamiltonian

$$H = \sum_i a_i U_i$$

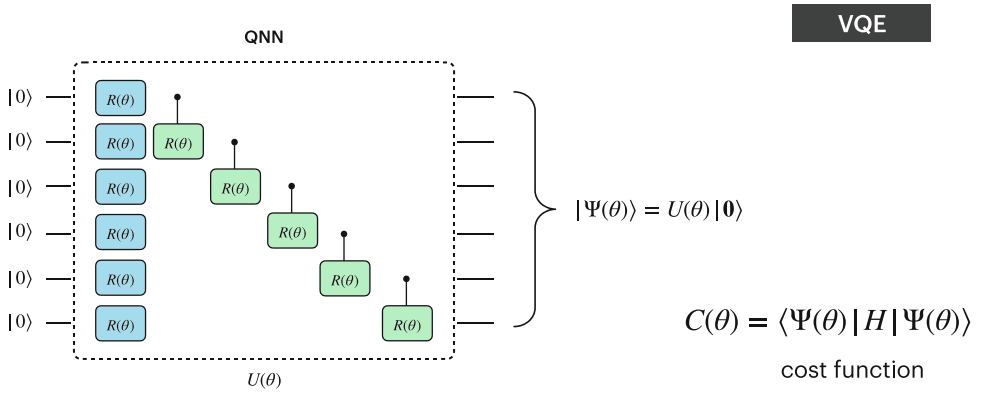


Fig. 24 Schematic of VQE for ground state estimation. The QNN (a parameterized circuit ansatz) is applied to an initial state (e.g., the zero state) over multiple qubits to generate the ground state of a given Hamiltonian H. The parameters in the QNN, i.e., the rotation angles of the parametrized gates (here, for simplicity, we use the same symbol θ for all the angles of different gates), are optimized so that the generated state of the QNN possesses the lowest expectation value of the given Hamiltonian

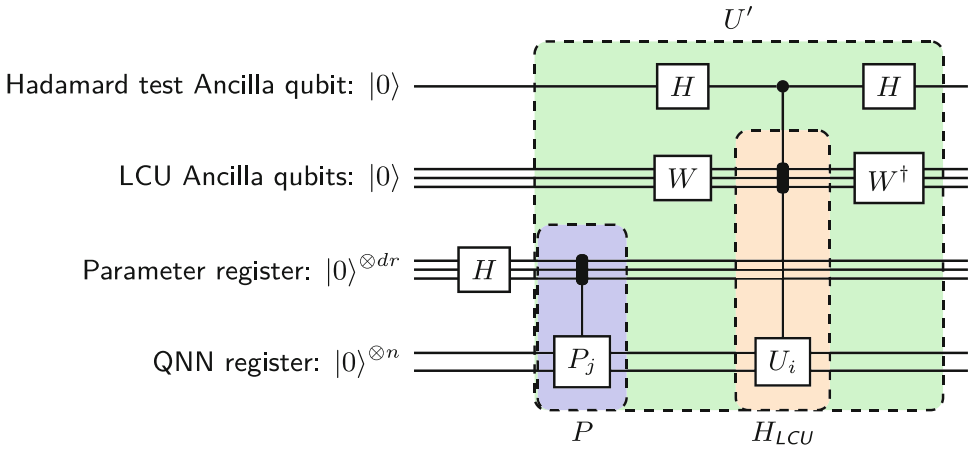


Fig. 25 Circuit for the amplitude encoding of the cost function for VQE. Here, we use the Hadamard test circuit for the amplitude encoding of the cost function, as detailed in 3 3.1 3.1.1 b. We use a technique "linear combinations of unitaries" (LCU) (Childs and Wiebe 2012) to implement the given Hamiltonian $H = \sum_i a_i U_i$. The unitary oracles W, H_{LCU} are defined as $W|0\rangle = \sum_i \sqrt{a_i}|i\rangle$, $H_{\mathsf{LCU}} = \sum_i |i\rangle\langle i| \otimes U_i$

where U_i is a unitary, $a_i > 0$ and $\sum_j a_i = 1$. (This assumption can be made without loss of generality by renormalizing the Hamiltonian and absorbing signs into the unitary matrix.) Let the state $|\psi(\boldsymbol{\theta})\rangle$ for $\boldsymbol{\theta} \in \mathbb{R}^m$ be the variational state prepared by the QNN. (m is the number of parameters in the QNN.) The cost function of the QNN is

$$C(\boldsymbol{\theta}) = \langle\psi(\boldsymbol{\theta})| \sum_i a_i U_i |\psi(\boldsymbol{\theta})\rangle. \tag{53}$$

Our goal is then to estimate

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left(\langle\psi(\boldsymbol{\theta})| \sum_i a_i U_i |\psi(\boldsymbol{\theta})\rangle \right). \tag{54}$$

Here, we use a technique "linear combinations of unitaries" (LCU) (Childs and Wiebe 2012) to implement the Hamiltonian. Define new unitary oracles W, H_{LCU} such that

$$W|0\rangle = \sum_i \sqrt{a_i}|i\rangle, \tag{55}$$

$$H_{\mathsf{LCU}} = \sum_i |i\rangle\langle i| \otimes U_i. \tag{56}$$

The amplitude encoding of the cost function of the QNN can be implemented using the following circuit in Fig. 25 (Gilyén et al. 2019a):

5.2 Learning to generate a pure state

Another application of our quantum training is when QNN is served as a generative model to learn a pure state. In our scenario, the target state is generated by a given unitary (e.g., a given sequence of gates), and the QNN serves as another generator circuit for the target state. The parameters in QNN are optimized such that the generated state of QNN matches the target state. This approach can be used to transform a given sequence of gates to a different/simpler sequence (e.g., translating circuits from superconducting gate sets to ion trap gate sets). A schematic of this application is presented in Fig. 26.

The amplitude encoding for this application has been given in Section 3 3.1 3.1.1 a.

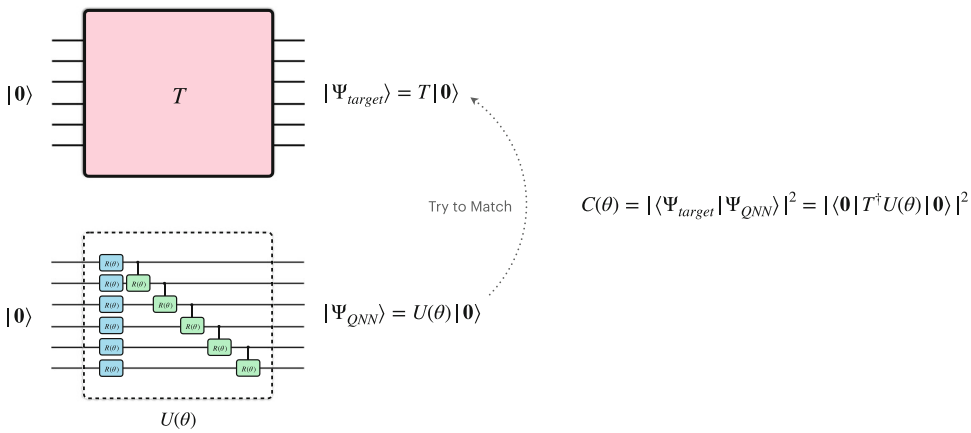


Fig. 26 Schematic of using QNN to generate a pure state. In our scenario, the target state is generated by a given unitary T, i.e., $|\Psi_{target}\rangle = T|0\rangle$); the QNN (denoted as $U(\theta)$) serves as another generator circuit for the target state. The parameters in QNN are optimized such that the generated state of QNN $|\Psi_{QNN}\rangle$ matches the target state. The cost function is the fidelity between the target state and the generated state by QNN

5.3 Training a quantum classifier

Finally, we discuss the application of QNN as a quantum classifier that performs supervised learning which is a standard problem in machine learning.

To formalize the learning task, let $\mathcal{X}$ be a set of inputs and $\mathcal{Y}$ a set of outputs. Given a dataset $\mathcal{D} = \{(x_1, y_1), ..., (x_M, y_M)\}$ of pairs of so called *training inputs* $x_m \in \mathcal{X}$ and *target outputs* $y_m \in \mathcal{Y}$ for $m = 1, ..., M$, the task of the model is to predict the output $y \in \mathcal{Y}$ of a new input $x \in \mathcal{X}$. For simplicity, we will assume in the following that $\mathcal{X} = \mathbb{R}^N$ and $\mathcal{Y} = \{0, 1\}$, which is a binary classification task on a N-dimensional real input space. In summary, the quantum classifier aims to learn an effective labeling function $\ell : \mathcal{X} \to \{0, 1\}$.

Given an input x_i and a set of parameters $\boldsymbol{\theta}$, the quantum classifier first embeds x_i into the state of a n-qubit quantum system via a state preparation circuit S_{x_i} such that $S_{x_i}|0\rangle = |\varphi(x_i)\rangle$ and subsequently uses a learnable quantum circuit $U(\boldsymbol{\theta})$ (QNN) as a predictive model to make inference. The predicted class label $y^{(i)} = f(x_i, \boldsymbol{\theta})$ is retrieved by measuring a designated qubit in the state $U(\boldsymbol{\theta})|\varphi(x)\rangle$. A schematic of the quantum classifier is presented in Fig. 27. Note that although the variational quantum classifier could be operated as a multiclass classifier, here we limit ourselves to the case of the binary classification discussed above and cast the multi-label tasks as a set of binary discrimination subtasks.

Denote $p(\lambda)$ as the probability of the measurement result on the designated qubit being λ ($\lambda \in \{0, 1\}$). The cost function of each training data point $L_i(\boldsymbol{\theta})$, as a function of y_i and $y^{(i)}$ and hence a function of $y_i, x_i, \boldsymbol{\theta}$ which we denote as $L(x_i, y_i, \boldsymbol{\theta})$, is chosen to be the probability of the measurement result on the designated qubit being identical to the given label (Schuld et al. 2020), namely

$$L_i(\boldsymbol{\theta}) = L(x_i, y_i, \boldsymbol{\theta}) := p(y_i). \tag{57}$$

Note here that the larger the probability is, the more correct the prediction is, so we want to maximize the cost (in this paper, in order to be coherent with the former narrative, we use "cost" instead of commonly used "likelihood" of inferring the correct label for a data sample.)

On the other hand, the quantum state of the system after the state preparation and QNN inference can be written as

$$|\Psi_i(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|\varphi(x_i)\rangle = \sqrt{p(0)}|0\rangle|u_{\boldsymbol{\theta}}\rangle + \sqrt{1-p(0)}|1\rangle|v_{\boldsymbol{\theta}}\rangle \tag{58}$$

$$= \begin{cases} \sqrt{p(y_i)}|0\rangle|u_{\boldsymbol{\theta}}\rangle + \sqrt{1-p(y_i)}|1\rangle|v_{\boldsymbol{\theta}}\rangle, & y_i = 0 \\ \sqrt{p(y_i)}|1\rangle|u_{\boldsymbol{\theta}}\rangle + \sqrt{1-p(y_i)}|0\rangle|v_{\boldsymbol{\theta}}\rangle, & y_i = 1 \end{cases} \tag{59}$$

in which $|u_{\boldsymbol{\theta}}\rangle, |v_{\boldsymbol{\theta}}\rangle$ are some normalized state that depend on $\boldsymbol{\theta}$.

From Eqs. 57 and 59, we can see that the cost of each data sample $L(x_i, y_i, \boldsymbol{\theta})$ is naturally encoded in the amplitude of the output state of QNN $|\Psi_i(\boldsymbol{\theta})\rangle$. We illustrate the amplitude encoding of the cost function for quantum classifier in Fig. 28. Constructing the "controlled-QNN" will achieve the amplitude encoding for all the parameter configurations in parallel. Based on this amplitude encoding, we can construct $e^{-i\gamma L(x_i, y_i, \boldsymbol{\theta})}$ using the methods discussed in Section 4.1.[2]

The total cost function of the whole training set can be defined as (for simplicity, we omit $\frac{1}{M}$ here)

$$C(\boldsymbol{\theta}) = \sum_i L(x_i, y_i, \boldsymbol{\theta}). \tag{60}$$

It follows immediately

$$e^{-i\gamma C(\boldsymbol{\theta})} = \Pi_i e^{-i\gamma L(x_i, y_i, \boldsymbol{\theta})}. \tag{61}$$

Therefore, the phase encoding of the total cost function can be implemented by accumulating individual phase encoding for each training sample; this process can be illustrated in Fig. 29.

Armed with the phase encoding of the total cost function, we can now construct the full quantum training protocol as in Fig. 30.

6 Discussion

In this paper, we proposed a framework leveraging quantum optimization routines for training QNNs. We have designed a variant of QAOA (AC-QAOA) tailored for QNN training problems. Our framework of using AC-QAOA to train QNNs consists of two major components: (1) a *phase oracle* that can achieve coherent phase encoding of the cost function of QNN and (2) *adaptive mixers* that can dramatically shorten the depth of QAOA layers while significantly improving the quality of the solution. We adopt RNN and RL to determine mixers sequence and optimize hyper-parameters. Various applications which our quantum algorithm can apply to were presented.

QAOA itself and all of its variants are, by construction, heuristics, and therefore their advantages are ultimately determined by testing performance on concrete problems. Heuristically, AC-QAOA is expected to process the advantages of its ancestors, i.e., ADAPT-QAOA and QDD. We leave as future work for demonstrating the advantages by numerical experiments. The estimation of the number of qubits needed is given in Appendix C. For a small toy example with 5 qubits and 10 rotation gates in the QNN, our

[2] Note that for the training data point with $y_i = 0$, C_1 in the Grover operator G and G^* has to be adjusted to $-Z$

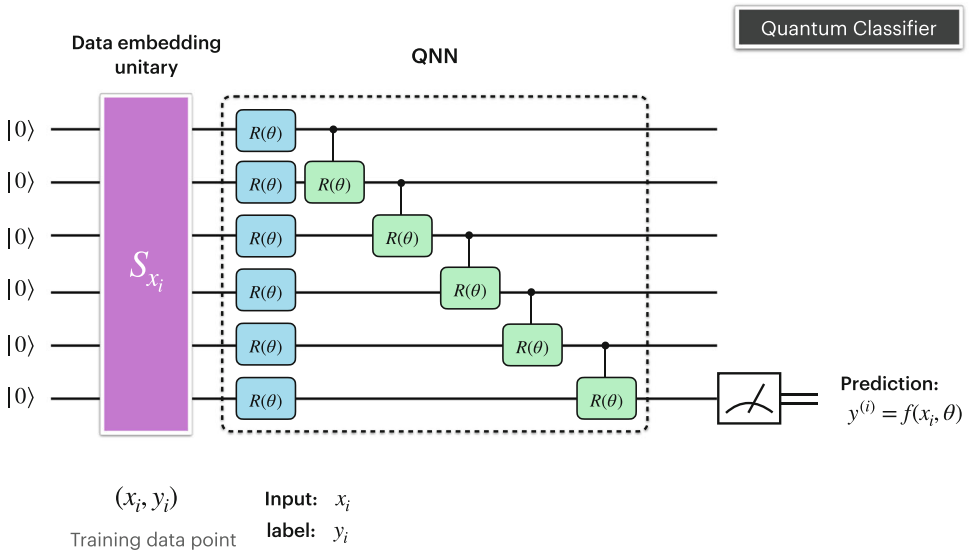


Fig. 27 Schematic of a quantum classifier. For a training data point (x_i, y_i), the quantum classifier first embeds x_i into the state of a n-qubit quantum system via a data embedding circuit S_{x_i} (purple box) such that $S_{x_i}|0\rangle = |\varphi(x_i)\rangle$ and subsequently uses a learnable quantum circuit $U(\boldsymbol{\theta})$ (QNN) as a predictive model to make inference (here, for simplicity, we use the same symbol θ for all the angles of different gates). The predicted class label $y^{(i)} = f(x_i, \boldsymbol{\theta})$ is retrieved by measuring a designated qubit in the state $U(\boldsymbol{\theta})|\varphi(x)\rangle$

protocol requires roughly 60 qubits to implement. Thus, we expect to demonstrate our protocol on near-term devices.

In this paper, we have only discussed optimizing the rotation parameters in QNNs (which belongs to a continuous optimization problem). However, our framework can also be used in *learning circuit structure*—i.e., to find better circuit ansatz (which belongs to discrete optimization problem)—or even learning the structure and parameters simultaneously. We leave these extensions to future work. Furthermore, we would like to explore the possibility of applying other quantum optimization algorithms (such as adiabatic quantum evolution, quantum walks) to QNN training. We hope this work will provide a useful framework for quantum training of future quantum devices.

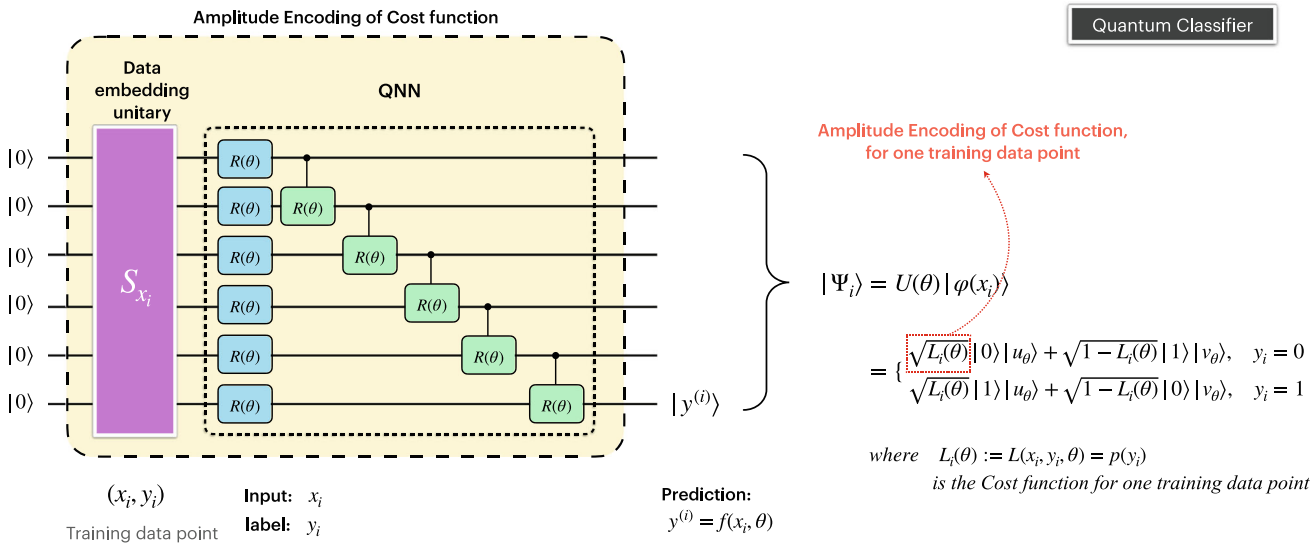


Fig. 28 Amplitude encoding of the cost function for quantum classifier. For a training data point (x_i, y_i), the quantum classifier first embeds x_i into the state of a n-qubit quantum system via a data embedding circuit S_{x_i} (purple box) such that $S_{x_i}|0\rangle = |\varphi(x_i)\rangle$ and subsequently uses a learnable quantum circuit $U(\boldsymbol{\theta})$ (QNN) as a predictive model to make inference (here, for simplicity, we use the same symbol θ for all the angles of different gates). The predicted class label $y^{(i)} = f(x_i, \boldsymbol{\theta})$ is retrieved by measuring a designated qubit in the state $U(\boldsymbol{\theta})|\varphi(x)\rangle$. Denote $p(\lambda)$ as the probability of the measurement result on the designated qubit being λ ($\lambda \in \{0, 1\}$). The cost function of each training data point $L_i(\boldsymbol{\theta})$, as a function of y_i and $y^{(i)}$ and hence a function of $y_i, x_i, \boldsymbol{\theta}$ which we denote as $L(x_i, y_i, \boldsymbol{\theta})$, is chosen to be the the probability of the measurement result on the designated qubit being identical to the given label, namely $L_i(\boldsymbol{\theta}) = L(x_i, y_i, \boldsymbol{\theta}) := p(y_i)$. We can see that the cost of each data sample is naturally encoded in the amplitude of the output state of QNN

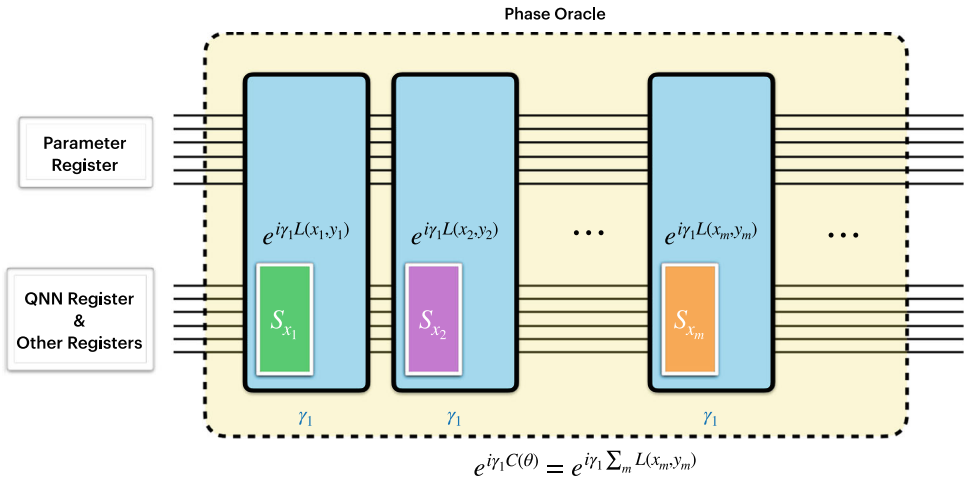


Fig. 29 Phase encoding of the total cost function of quantum classifier. The total cost function of the whole training set can be defined as $C(\boldsymbol{\theta}) = \sum_i L(x_i, y_i, \boldsymbol{\theta})$. It follows immediately $e^{-i\gamma C(\boldsymbol{\theta})} = \Pi_i e^{-i\gamma L(x_i,y_i,\boldsymbol{\theta})}$. Therefore, the phase encoding of the total cost function (the overall yellow box) can be implemented by accumulating individual phase encoding for each training sample (blue boxes). In this figure, we omit $\boldsymbol{\theta}$ in $L(x_i, y_i, \boldsymbol{\theta})$ for simplicity. The inner boxes in the blue boxes represent different data embedding unitary for the training data points

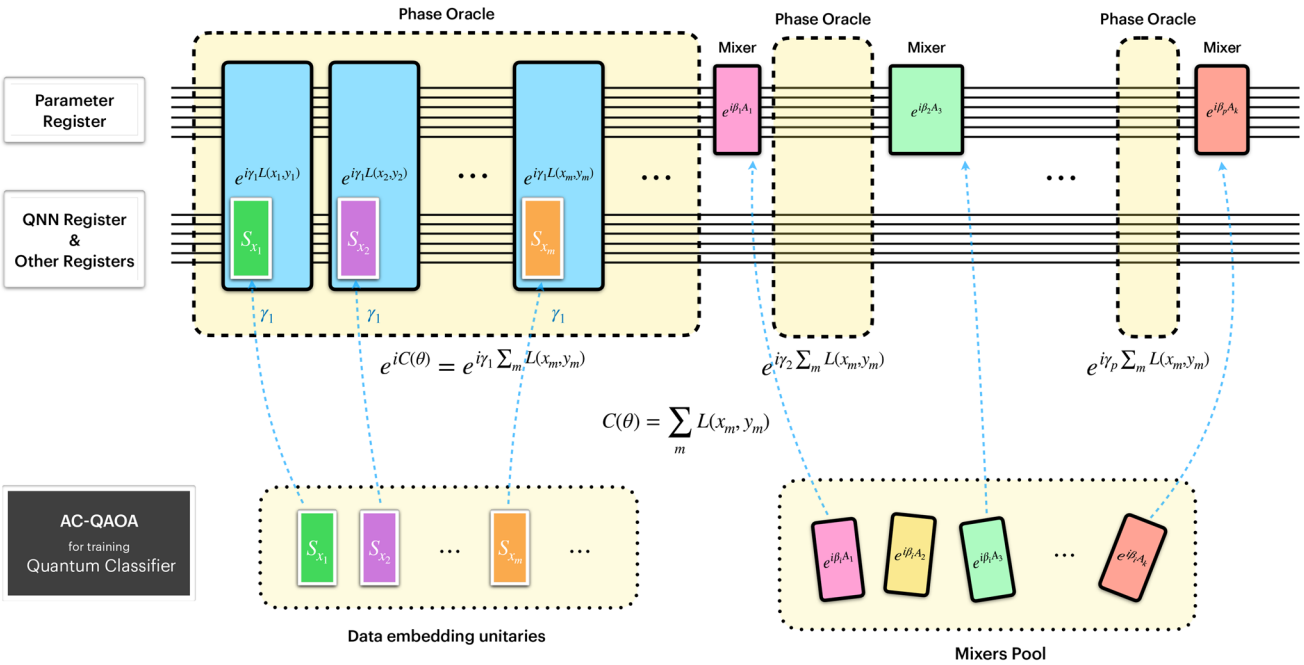


Fig. 30 Schematic of our quantum training protocol for quantum classifier. The full quantum training protocol consists of the alternation of the phase oracle that achieves coherent phase encoding of the cost function and the adaptive mixers chosen from a mixers pool. The phase encoding of the total cost function for the quantum classifier is detailed in Fig. 29. The total cost function of the whole training set can be defined as $C(\boldsymbol{\theta}) = \sum_i L(x_i, y_i, \boldsymbol{\theta})$. It follows that $e^{-i\gamma C(\boldsymbol{\theta})} = \Pi_i e^{-i\gamma L(x_i,y_i,\boldsymbol{\theta})}$. Therefore, the phase oracle for the total cost function (the yellow boxes in the upper part of this figure) can be implemented by accumulating individual phase encoding for each training sample (blue boxes). In this figure, we omit $\boldsymbol{\theta}$ in $L(x_i, y_i, \boldsymbol{\theta})$ for simplicity. The colorful boxes with white border represent different data embedding unitary for the training data points. The colorful boxes with black border (excluding the blue ones for the phase encoding) represent different mixers chosen from a mixers pool

A. Proofs

Here, we provide proofs for the claims mentioned in the main text.

$$U = \sum_j |j\rangle\langle j| \otimes U_j$$

Proof

$$\begin{aligned}
U &= [H \otimes I \otimes I \otimes I] \cdot [|0\rangle\langle 0| \otimes \Big(\sum_j |j\rangle\langle j| \otimes P_j \otimes T\Big) \\
&\quad + |1\rangle\langle 1| \otimes (\sum_j |j\rangle\langle j| \otimes T \otimes P_j)] \cdot [H \otimes I \otimes I \otimes I] \\
&= \sum_j [H \otimes I \otimes I \otimes I] \cdot [|0\rangle\langle 0| \otimes |j\rangle\langle j| \otimes P_j \otimes T \\
&\quad + |1\rangle\langle 1| \otimes |j\rangle\langle j| \otimes T \otimes P_j] \cdot [H \otimes I \otimes I \otimes I] \\
&= \sum_j [I \otimes H \otimes I \otimes I] \cdot [|j\rangle\langle j| \otimes |0\rangle\langle 0| \otimes P_j \otimes T \\
&\quad + |j\rangle\langle j| \otimes |1\rangle\langle 1| \otimes T \otimes P_j] \cdot [I \otimes H \otimes I \otimes I] \\
&= \sum_j [I \otimes H \otimes I \otimes I] \cdot [|j\rangle\langle j| \otimes (|0\rangle\langle 0| \otimes P_j \otimes T \\
&\quad + |1\rangle\langle 1| \otimes T \otimes P_j)] \cdot [I \otimes H \otimes I \otimes I] \\
&= \sum_j (I \cdot |j\rangle\langle j| \cdot I) \otimes \Big([H \otimes I \otimes I] \cdot [|0\rangle\langle 0| \otimes P_j \otimes T \\
&\quad + |1\rangle\langle 1| \otimes T \otimes P_j] \cdot [H \otimes I \otimes I]\Big) \\
&= \sum_j |j\rangle\langle j| \otimes U_j
\end{aligned}$$

□

$$G = \sum_j |j\rangle\langle j| \otimes G_j$$

Proof

$$G = C_1 U^{-1} C_2 U = \tag{A1}$$

$$C_1 \left(\sum_j |j\rangle\langle j| \otimes U_j^\dagger\right) C_2 \left(\sum_k |k\rangle\langle k| \otimes U_k\right) = \tag{A2}$$

$$\sum_j \sum_k C_1 \left(|j\rangle\langle j| \otimes U_j^\dagger\right) C_2 \left(|k\rangle\langle k| \otimes U_k\right) = \tag{A3}$$

$$\sum_j |j\rangle\langle j| \otimes C_1 U_j^\dagger C_2 U_j = \tag{A4}$$

$$\sum_j |j\rangle\langle j| \otimes G_j \tag{A5}$$

□

B. Performance of quantum training using Grover adaptive search

It has been shown in Ref. (Baritompa et al. 2005) that global optimization by Grover adaptive search takes $O(\sqrt{N/s})$ calls of Grover oracle in which N is the dimension of the search space, s is the number of global optima and assuming s is small compared to N. A unique optimum will be found after $O(\log N)$ improvements on the threshold, in expectation. The number of measurements invoked between the improvements in no larger than $O(\log \sqrt{N})$. Therefore, the total number of measurements for Grover adaptive search to find a global optimum is no larger than $O(\log N \log \sqrt{N})$.

Here, we apply the above results to our QNN training problem. Taking training VQE as an example, we evaluate the number of "controlled-QNN" runs, the number of QNN runs, and the number of measurements respectively as follows.

1. Number of "controlled-QNN" runs

For our quantum training, each Grover iteration consists of the steps of amplitude encoding (AE), phase estimation (PE), threshold oracle, uncomputation. Taking training VQE as an example, the number of the number of "controlled-QNN" runs of each step can be listed as

- *In amplitude encoding (AE):* $n_{AE} = 1$
- *In phase estimation (PE):* $n_{PE} = (2^t - 1)2 = 2^{t+1} - 2$
- *In threshold oracle:* 0
- *In uncomputation:* $n_{PE} + n_{AE}$

in which t is the number of qubits in the amplitude register for the phase estimation.

The number of "controlled-QNN" runs of each Grover iteration, which we denote as N_0, is the sum of the above numbers:

$$N_0 = 2(n_{AE} + n_{PE}) = 2(2^{t+1} - 1) \tag{B1}$$

To obtain phase accurate to n' bits with probability of success at least $1 - \epsilon_1$, t is chosen as

$$t = n' + \lceil \log(2 + \frac{1}{2\epsilon_1}) \rceil. \tag{B2}$$

Hence,

$$N_0 \approx (2^{n'+2}(2 + \frac{1}{2\epsilon_1}) - 2) \tag{B3}$$

For small ϵ_1, we have

$$N_0 \approx 2^{n'+2} \frac{1}{2\epsilon_1} \tag{B4}$$

n' determines the precision of the QNN cost function evaluated by phase estimation, which we denote as ϵ_2, and we have

$$2^{-n'} = \epsilon_2 \tag{B5}$$

therefore,

$$N_0 \approx \frac{2}{\epsilon_2 \epsilon_1} \tag{B6}$$

Therefore, the total number of "controlled-QNN" runs for $O(\sqrt{N/s})$ Grover iterations scales as

$$N_{\text{controlled-QNN}} \sim O(\frac{1}{\epsilon_2 \epsilon_1}\sqrt{N/s}) \tag{B7}$$

in which N is the dimension of the parameter space of QNN and s is the number of global optima of the QNN cost function.

Let r be the number of parameters (rotation angles) in QNN, and d be the number of control qubits for each rotation angle. Therefore,

$$N = 2^{dr} \tag{B8}$$

On the other hand,

$$2^{-d} = \epsilon_0, \tag{B9}$$

where ϵ_0 is the precision of each angle value. Hence,

$$N = (\frac{1}{\epsilon_0})^r \tag{B10}$$

Inserting Eq. B10 into Eq. B7, we get

$$N_{\text{controlled-QNN}} \sim O\left(\frac{1}{\epsilon_2 \epsilon_1}\left(\frac{1}{\epsilon_0}\right)^{r/2} s^{-\frac{1}{2}}\right) \tag{B11}$$

2. Number of measurements

As mentioned before, the total number of measurements for Grover adaptive search to find a global optimum scales as

$$N_{\text{Measurements}} \sim O(\log N \log \sqrt{N}) \tag{B12}$$

Inserting Eq. B8 into Eq. B12, we get

$$N_{\text{Measurements}} \sim O\left((dr)^{3/2}\right) \tag{B13}$$

From Eq. B9, we have $d \sim O\left(\log\left(\frac{1}{\epsilon_0}\right)\right)$; therefore,

$$N_{\text{Measurements}} \sim O\left(\left(r \log\left(\frac{1}{\epsilon_0}\right)\right)^{1.5}\right) \tag{B14}$$

3. Number of QNN runs

After each measurement on the parameter register, we obtain a specific parameter configuration of QNN. We then need to estimate the cost function for this particular parameter configuration. For VQE, the cost function is the expectation value of some Hamiltonian and the number of the estimation scale as $O(1/\epsilon^\alpha)$ for some small power α (Knill et al. 2007) (α is a small integer about 1 or 2), where ϵ is the desired accuracy of the expectation value. For our QNN training, we choose the accuracy ϵ to be ϵ_2 defined above. Taking $\alpha = 1$, the number of QNN runs after each measurements scale as $O(1/\epsilon_2)$ and the total number of QNN runs all the measurements during the quantum training scale as

$$N_{\text{QNN}} \sim O\left(\frac{1}{\epsilon_2}\right) N_{\text{Measurements}}. \tag{B15}$$

Inserting Eq. B14 into Eq. B15, we have

$$N_{\text{QNN}} \sim O\left(\frac{1}{\epsilon_2}\left(r \log\left(\frac{1}{\epsilon_0}\right)\right)^{1.5}\right) \tag{B16}$$

C. Number of qubits needed for quantum training by AC-QAOA

Taking training VQE as example, the number of qubits in each register can be listed as

- *For QNN register*: n qubits
- *For parameter register*: dr qubits (r is the number of parameters in QNN, and d is the number of control qubits for each rotation angle.)
- *For Hadamard test*: 1 ancilla qubit
- *For LCU register and other registers*: $O(\log\log(1/\epsilon))$ (Gilyén et al. 2019a) qubits (ϵ is the precision of implementing the phase oracle by LCU)

In total, the number of qubits needed for quantum training is

$$n_{total} \sim n + dr + O(\log\log(1/\epsilon)) \tag{C1}$$

For instance, when $n = 5$, $d = 5$, $r = 10$, $\epsilon = 10^{-8}$, $n_{total} \approx 60$.

Author Contributions Y.L. wrote the main manuscript. C.F. and M.H. supervised the research. All authors reviewed the manuscript.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions.

Availability of data and materials This declaration is not applicable.

Declarations

Conflict of interest The authors declare no competing interests.

References

Arrasmith A, Cerezo M, Czarnik P, Cincio L, Coles PJ (2020) Effect of barren plateaus on gradient-free optimization

Baritompa WP, Bulger DW, Wood GR (2005) Grovers quantum algorithm applied to global optimization. SIAM J Optim 15(4):1170–1184. https://doi.org/10.1137/040605072

Barkoutsos PKl, Nannicini G, Robert A, Tavernelli I, Woerner S (2020) Improving variational quantum optimization using CVaR. Quantum 4:256. ISSN 2521-327X. https://doi.org/10.22331/q-2020-04-20-256

Bartlett SD, Sanders BC, Braunstein SL, Nemoto K (2002) Efficient classical simulation of continuous variable quantum information processes. Phys Rev Lett 88:097904. https://doi.org/10.1103/PhysRevLett.88.097904. https://link.aps.org/doi/10.1103/PhysRevLett.88.097904

Benedetti M, Garcia-Pintos D, Perdomo O, Leyton-Ortega V, Nam Y, Perdomo-Ortiz A (2019) A generative modeling approach for benchmarking and training shallow quantum circuits. npj Quantum Inf 5(1). ISSN 2056-6387. https://doi.org/10.1038/s41534-019-0157-8

Berry Dominic W, Childs Andrew M, Cleve Richard, Kothari Robin, Somma Rolando D (2015) Simulating Hamiltonian dynamics with a truncated Taylor series. Phys Rev Lett 114:090502. https://doi.org/10.1103/PhysRevLett.114.090502. https://link.aps.org/doi/10.1103/PhysRevLett.114.090502

Brassard G, Hoyer P, Mosca M, Tapp A (2000) Quantum amplitude amplification and estimation. arXiv e-prints, art. quant-ph/0005055

Bravyi S, Kliesch A, Koenig R, Tang E (2019) Obstacles to state preparation and variational optimization from symmetry protection

Buhrman H, Cleve R, Watrous J, de Wolf R (2001) Quantum fingerprinting. Phys Rev Lett 87(16). https://doi.org/10.1103/PhysRevLett.87.167902. ISSN 1079-7114

Bulger D (2005) Quantum basin hopping with gradient-based local optimisation. arXiv e-prints, art. quant-ph/0507193

Campos E, Nasrallah A, Biamonte J (2020) Abrupt transitions in variational quantum circuit training

Cerezo M, Sone A, Volkoff T, Cincio L, Coles PJ (2020) Cost-function-dependent barren plateaus in shallow quantum neural networks. Preprint at arXiv:2001.00550. https://arxiv.org/abs/2001.00550

Childs AM, Wiebe N (2012) Hamiltonian simulation using linear combinations of unitary operations

Du Y, Hsieh M-H, Liu T, Tao D (2018a) The expressive power of parameterized quantum circuits. Preprint at arXiv:1810.11922

Du Y, Hsieh M-H, Liu T, Tao D (2018b) Implementable quantum classifier for nonlinear data

Du Y, Hsieh M-H, Liu T, You S, Tao D (2020a) On the learnability of quantum neural networks

Du Y, Huang T, You S, Hsieh M-H, Tao D (2020b) Quantum circuit architecture search: error mitigation and trainability enhancement for variational quantum solvers

Durr C, Hoyer P (1996) A quantum algorithm for finding the minimum. arXiv e-prints, art. quant-ph/9607014

Farhi E, Goldstone J, Gutmann S, Neven H (2017) Quantum algorithms for fixed qubit architectures

Farhi E, Neven H (2018) Classification with quantum neural networks on near term processors. Preprint at arXiv:1802.06002. https://arxiv.org/abs/1802.06002

Farhi E, Goldstone J, Gutmann S (2014) A quantum approximate optimization algorithm

Gilliam A, Woerner S, Gonciulea C (2020) Grover adaptive search for constrained polynomial binary optimization

Gilyén A, Arunachalam S, Wiebe N (2019a) Optimizing quantum optimization algorithms via faster quantum gradient computation. Proceedings of the thirtieth annual ACM-SIAM symposium on discrete algorithms, pp 1425–1444. https://doi.org/10.1137/1.9781611975482.87

Gilyén A, Su Y, Low GH, Wiebe N (2019b) Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics. Proceedings of the 51st Annual ACM SIGACT symposium on theory of computing - STOC 2019. https://doi.org/10.1145/3313276.3316366

Guéry-Odelin D, Ruschhaupt A, Kiely A, Torrontegui E, Martínez-Garaot S, Muga JG (2019) Shortcuts to adiabaticity: concepts, methods, and applications. Rev Mod Phys 91(4). https://doi.org/10.1103/revmodphys.91.045001. ISSN 1539-0756

Hadfield S, Wang Z, O'Gorman B, Rieffel EG, Venturelli D, Biswas R (2019a) From the quantum approximate optimization algorithm to a quantum alternating operator ansatz. Algorithms 12(2):34. https://doi.org/10.3390/a12020034. ISSN 1999-4893

Hadfield S, Wang Z, O'Gorman B, Rieffel EG, Venturelli D, Biswas R (2019b) From the quantum approximate optimization algorithm to a quantum alternating operator ansatz. Algorithms 12(2):34. https://www.mdpi.com/1999-4893/12/2/34

Hamilton KE, Dumitrescu EF, Pooser RC (2019) Generative model benchmarks for superconducting qubits. Phys Rev A 99(6). https://doi.org/10.1103/PhysRevA.99.062323. ISSN 2469-9934

Hegade NN, Paul K, Ding Y, Sanz M, Albarrán-Arriagada F, Solano E, Chen X (2020) Shortcuts to adiabaticity in digitized adiabatic quantum computing

Huang HL, Du Y, Gong M, Zhao Y, Wu Y, Wang C, Li S, Liang F, Lin J, Xu Y, Yang R, Liu T, Hsieh MH, Deng H, Rong H, Peng CZ, Lu CY, Chen YA, Tao D, Zhu X, Pan JW (2020) Experimental quantum generative adversarial networks for image generation,

Jiang Z, Rieffel EG, Wang Z(2017) Near-optimal quantum circuit for Grover's unstructured search using a transverse field. Phys Rev A

95(6). https://doi.org/10.1103/PhysRevA.95.062317. ISSN 2469-9934
Khairy S, Shaydulin R, Cincio L, Alexeev Y, Balaprakash P (2019) Reinforcement-learning-based variational quantum circuits optimization for combinatorial problems
Knill E, Ortiz G, Somma RD (2007) Optimal quantum measurements of expectation values of observables. Phys Rev A 75:012328. https://doi.org/10.1103/PhysRevA.75.012328. https://link.aps.org/doi/10.1103/PhysRevA.75.012328
Liao Y, Ebler D, Liu F, Dahlsten O (2020) Quantum speed-up in global optimization of binary neural nets. New J Phys. http://iopscience.iop.org/article/10.1088/1367-2630/abc9ef
Lloyd S (2018) Quantum approximate optimization is computationally universal
Marrero CO, Kieferová M, Wiebe N (2020) Entanglement induced barren plateaus
Marsh S, Wang J (2018) A quantum walk assisted approximate algorithm for bounded np optimisation problems
Mbeng GB, Fazio R, Santoro G (2019) Quantum annealing: a journey through digitalization, control, and hybrid quantum variational schemes
McClean JR, Romero J, Babbush R, Aspuru-Guzik A (2016) The theory of variational hybrid quantum-classical algorithms. New J Phy 18(2):023023. https://iopscience.iop.org/article/10.1088/1367-2630/18/2/023023/meta
McClean JR, Boixo S, Smelyanskiy VN, Babbush R, Neven H (2018) Barren plateaus in quantum neural network training landscapes. Nature communications, 9(1):4812. https://www.nature.com/articles/s41467-018-07090-4
McClean JR, Harrigan MP, Mohseni M, Rubin NC, Jiang Z, Boixo S, Smelyanskiy VN, Babbush R, Neven H (2020) Low depth mechanisms for quantum optimization. Preprint at arXiv:2008.08615
Mitarai K, Negoro M, Kitagawa M, Fujii K (2018) Quantum circuit learning. Phys Rev A 98(3):032309. https://doi.org/10.1103/PhysRevA.98.032309. https://journals.aps.org/pra/abstract/10.1103/PhysRevA.98.032309
Morales MES, Biamonte J, Zimborás Z (2019) On the universality of the quantum approximate optimization algorithm
Morley JG, Chancellor N, Bose S, Kendon V (2019) Quantum search with hybrid adiabatic-quantum-walk algorithms and realistic noise. Phys Rev A 99:022339. https://doi.org/10.1103/PhysRevA.99.022339. https://link.aps.org/doi/10.1103/PhysRevA.99.022339
Niu MY, Lu S, Chuang IL (2019) Optimizing QAOA: Success Probability and Runtime Dependence on Circuit Depth
Peruzzo A, McClean J, Shadbolt P, Yung M-H, Zhou X-Q, Love PJ, Aspuru-Guzik A, O'Brien JL (2014) A variational eigenvalue solver on a photonic quantum processor. Nat Commun 5:4213. https://doi.org/10.1038/ncomms5213
Schuld M, Bocharov A, Svore KM, Wiebe N (2020) Circuit-centric quantum classifiers. Phys Rev A 101(3). https://doi.org/10.1103/PhysRevA.101.032308. ISSN 2469-9934
Skolik A, McClean JR, Mohseni M, van der Smagt P, Leib M (2020) Layerwise learning for quantum neural networks. Preprint at arXiv:2006.14904. https://arxiv.org/abs/2006.14904
Streif M, Leib M(2019) Comparison of QAOA with quantum and simulated annealing
Verdon G, Pye J, Broughton M (2018) A universal training algorithm for quantum deep learning. Preprint at arXiv:1806.09729. https://arxiv.org/abs/1806.09729
Verdon G, Arrazola JM, Brádler K, Killoran N (2019a) A quantum approximate optimization algorithm for continuous problems
Verdon G, Broughton M, McClean JR, Sung KJ, Babbush R, Jiang Z, Neven H, Mohseni M (2019b) Learning to learn with quantum neural networks via classical neural networks
Volkoff T, Coles PJ (2020) Large gradients via correlation in random parameterized quantum circuits. Preprint at arXiv:2005.12200. https://arxiv.org/abs/2005.12200
Wang S, Fontana E, Cerezo M, Sharma K, Sone A, Cincio L, Coles PJ (2020) Noise-induced barren plateaus in variational quantum algorithms
Wang Y (2017) A quantum walk enhanced Grover search algorithm for global optimization
Warren A, Zhu L, Tang HL, Najafi K, Barnes E, Economou S (2020) RNN-VQE: a machine learning approach to generating variational ansatze. Bull Am Phys Soc
Wauters Matteo M, Panizon Emanuele, Mbeng Glen B, Santoro Giuseppe E (2020) Reinforcement-learning-assisted quantum optimization. Phys Rev Research 2:033446. https://doi.org/10.1103/PhysRevResearch.2.033446. https://link.aps.org/doi/10.1103/PhysRevResearch.2.033446
Weedbrook Christian, Pirandola Stefano, García-Patrón Raúl, Cerf Nicolas J, Ralph Timothy C, Shapiro Jeffrey H, Lloyd Seth (2012) Gaussian quantum information. Rev Mod Phys 84:621–669. https://doi.org/10.1103/RevModPhys.84.621. https://link.aps.org/doi/10.1103/RevModPhys.84.621
Whitfield JD, Rodríguez-Rosario CA, Aspuru-Guzik A (2010) Quantum stochastic walks: a generalization of classical random walks and quantum walks. Phys Rev A 81(2). https://doi.org/10.1103/PhysRevA.81.022323. ISSN 1094-1622
Yao J, Bukov M, Lin L (2020a) Policy gradient based quantum approximate optimization algorithm
Yao J, Lin L, Bukov M(2020b) Reinforcement learning for many-body ground state preparation based on counter-diabatic driving
Zeng J, Wu Y, Liu JG, Wang L, Hu J (2019) Learning and inference on generative adversarial quantum circuits. Phys Rev A 99(5). https://doi.org/10.1103/PhysRevA.99.052306. ISSN 2469-9934
Zhang K, Hsieh MH, Liu L, Tao D (2020) Toward trainability of quantum neural networks
Zhu L, Tang HL, Barron GS, Mayhall NJ, Barnes E, Economou SE (2020) An adaptive quantum approximate optimization algorithm for solving combinatorial problems on a quantum computer