Ma et al. *EPJ Quantum Technology* (2024) 11:54
https://doi.org/10.1140/epjqt/s40507-024-00265-7

EPJ Quantum Technology
a SpringerOpen Journal

RESEARCH Open Access

Continuous evolution for efficient quantum architecture search

QuanGong Ma[1], ChaoLong Hao[2*], XuKui Yang[2], LongLong Qian[3], Hao Zhang[1], NianWen Si[1,2], MinChen Xu[1] and Dan Qu[2*]

*Correspondence:
hcl_xdspeechlab@aliyun.com;
qudan_xd@163.com
[2]Laboratory for Advanced Computing and Intelligence Engineering, Zhengzhou, 450001, China
Full list of author information is available at the end of the article

Abstract

Variational quantum algorithms (VQAs) have been successfully applied to quantum approximate optimization algorithms, variational quantum compiling, and quantum machine learning models. The performance of VQAs is significantly influenced by the architecture of parameterized quantum circuits (PQCs). Quantum architecture search aims to automatically discover high-performance quantum circuits for specific VQA tasks. Quantum architecture search algorithms that utilize both SuperCircuit training and a parameter-sharing approach can save computational resources. If we directly follow the parameter-sharing approach, the SuperCircuit has to be trained to compensate for the worse search space. To address the challenges posed by the worse search space, we introduce an optimization strategy known as the efficient continuous evolutionary approach using Non-dominated Sorting Genetic Algorithm-II (NSGA-II). Then, we leverage prior information (symmetric property) designing Structure Symmetric Pruning for removing redundant gates of the searched ansatz. Experiments show that the efficient continuous evolutionary approach can search for more quantum architectures with better performance; the number of high-performance ansatzes obtained by our method is 10% higher than that in the literature (Du et al. in npj Quantum Inf. 8:62, 2022). The application of Structure Symmetric Pruning effectively reduces the number of parameters in quantum circuits without compromising their performance significantly. In binary classification tasks, the pruned quantum circuits exhibit an average accuracy reduction of 0.044 compared to their unpruned counterparts.

Keywords: Quantum architecture search; NSGA-II; Symmetric pruning

1 Introduction

Variational Quantum Algorithms (VQAs) are a class of algorithms that use quantum circuits with adjustable parameters to accomplish tasks similar to machine learning. The variational quantum algorithms [2, 3] including quantum neural networks [4, 5] and variational quantum eigensolvers (VQEs) [6–8], are a class of promising candidates to use noisy intermediate-scale quantum (NISQ) devices to solve practical tasks that are beyond the reach of classical computers [9]. The performance of variational quantum algorithms de-

pends heavily on the design of the quantum circuit architecture [10]. Developing a suitable quantum circuit architecture requires expertise. In the realm of neural network structures, the research focus has transitioned from manual design to automated machine-driven design processes [11]. Just as in neural network architecture design, the goal of quantum architecture search is to enable automated machine-driven design of the quantum circuit architecture tailored for specific VQAs.

Numerous scholars have conducted extensive research in quantum architecture search to facilitate the automated design of quantum circuits for VQAs by machines. The methods they propose include the differentiable circuit search method, deep reinforcement learning method, evolutionary methods, and so on [12]. Inspired by the work of differentiable architecture search (DARTS) [13], Zhang et al. proposed a framework for automatically building the parametrized quantum circuit for variational quantum algorithms [14]. Later, Wu et al. also proposed a search algorithm for differential quantum architecture search [15]. In addition to using differential quantum architecture search algorithms, He et al use meta-learning for quantum architecture search [16]. Inspired by the classic neural network architecture search [11], some researchers employ reinforcement learning to finish the quantum architecture search [17–19]. In the work of [1, 20], researchers have built SuperCircuit and trained SuperCircuit to update the parameters in the parameter space. SuperCircuit-based works are inspired by the supernet method in machine learning model [21, 22]. Similarly, employing evolution algorithms to perform quantum architecture search can also be a candidate strategy [23–25]. If we directly follow the parameter sharing approach proposed by [1, 20], the SuperCircuit has to be trained to compensate for the worse search space, i.e., dividing the search results for QAS into Sgood and Sbad, some of the worse performing ansatzes (Sbad) performed better than the ansatzes in Sgood after training independently [1]. Therefore, it is necessary to reform the existing search evolution algorithm based on SuperCircuit.

In this paper, we propose an efficient evolutionary algorithm-based quantum architecture search framework (CEQAS). A continuous evolution strategy is developed to utilize the knowledge we have learned in the last evolution generation. Specifically, we continue to use the training method of SuperCircuit in [1, 20], to complete the training of SuperCircuit and update the parameters of the parameter space. To better accomplish the task of updating the parameter space and looking for the quantum circuit architecture, we use NSGA-II in the search stage to create a strategy for "optimizing the parameter space while searching." We use NSGA-II to evaluate the quantum circuit architecture obtained during the search process. Individuals in the evolutionary algorithm representing architectures derived in the SuperCircuit will be generated through several benchmark operations (i.e., crossover and mutation). Non-dominated sort strategy is adopted to select several excellent architectures corresponding cells in the SuperCircuit will be updated for subsequent optimization. The general flow of our work is shown in Fig. 1. Furthermore, we introduce the Structure Symmetric Pruning to eliminate redundant parameters and gates in quantum circuits, coupled with fine-tuning to reduce the number of parameters while preserving performance.

Overall, the continuous evolution strategy can achieve better performance of the quantum architecture search algorithm when a small number of quantum circuits are sampled for training SuperCircuit. The contributions of this work are two-fold:

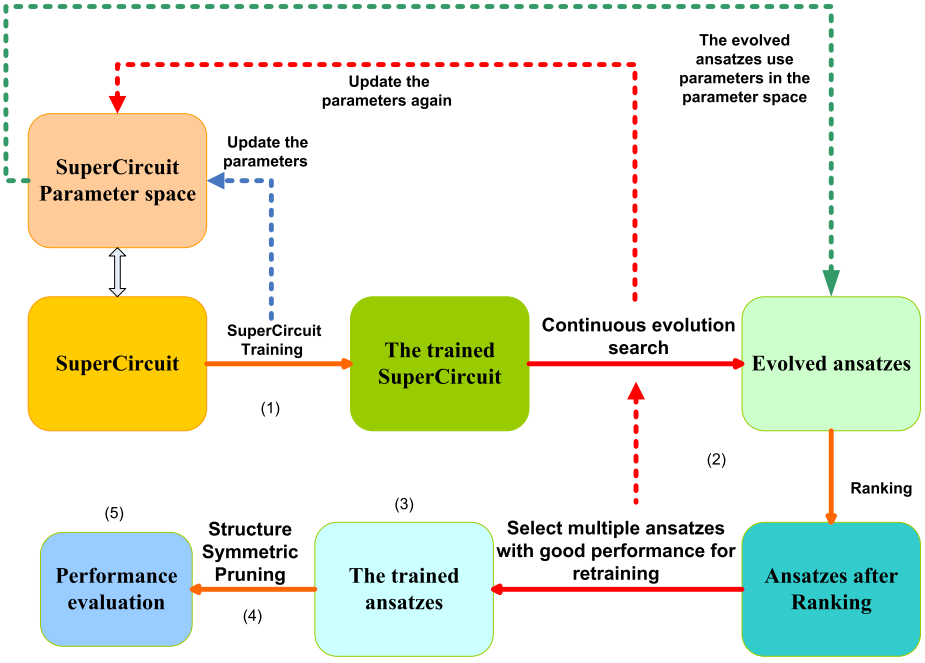

Figure 1 CEQAS Overview (1) A SuperCircuit is trained by iterative sampling and updating parameter subsets (SubCircuits). (2) Search for ansatzes with the best performance through continuous evolution (highest accurcay for QML). (3) Retrain the searched ansatzes. (4) Iterative pruning and finetuning to remove redundant gates. (5) Ansatzes performance evaluation after pruning

(1) Drawing insights from [1, 20], we employ NSGA-II to develop a strategy for "optimizing the parameter space while searching". This enhances the parameter space during the search stage and facilitates the more efficient generation of ansatzes on the Pareto front.

(2) After searching for the ansatzes with high performance, we introduce a "Structure Symmetric Pruning" scheme to reduce the number of parameters in ansatz. In contrast with classical pruning methods, our proposal does not require any gradient information to construct the symmetric ansatz. Instead, it removes the redundant gates and shrinks the solution space according to the information of the problem Hamiltonian.

Several works also propose to search circuits based on SuperCircuit. Our work is distinct from that presented in Ref. [20], where they conduct a noise-adaptive co-search for circuit optimization and qubit mapping. Their work also prunes quantum circuits, and they use a pruning strategy that puts quantum gate parameters close to 0 directly to 0. It is not always applicable in quantum circuits. In other words, the gradient information fails to provide any useful information to guide pruning [26]. Meanwhile, the output of quantum circuits can be regarded as a periodic function of parameters [27], which forbids employing the parameters' magnitude as the metric to guide the pruning. Therefore, it is inappropriate to straightforwardly apply classical pruning methods to quantum circuits, where the extracted ansatz may not promise the trainability. Ref. [1] also uses ideas based on SuperCircuit in their work. Following the initial training of the SuperCircuit, the work in Ref. [1] also mentions the worse search space, which has not yet been compensated for. Thus, our enhancement over the methodology presented in Ref. [1] lies in our proposition to develop a strategy that involves "optimizing the parameter space while searching" through NSGA-II to provide additional compensation for the search space. Our work distinguishes itself

from Ref. [20] by refraining from using the parameters' magnitude as the metric to guide pruning, and we construct "Structure Symmetric Pruning" scheme to shrink the solution space.

2 Related works

2.1 Quantum architecture search

SuperCircuit-based Quantum Architecture Search (QAS) methods contain two steps: SuperCircuit parameter optimization and architecture optimization [1, 20]. In SuperCircuit parameter optimization, they first construct a SuperCircuit by stacking a sufficient number of layers of pre-defined parameterized gates to cover a large design space. Then, they train the SuperCircuit by sampling and updating the parameter subsets (SubCircuits) from the SuperCircuit. The performance of a SubCircuit with inherited parameters from the SuperCircuit can provide a reliable relative performance estimation for the individual SubCircuit trained from scratch. This method is called parameter inheritance, which is applied in the work of [1, 20]. In this way, we only pay the training cost once but can evaluate all the SubCircuits efficiently. Hence, the search cost is greatly decreased. Particularly in [1], authors use multiple SuperCircuits to further improve the performance of QAS by building multiple expert parameter spaces.

In the study of architecture optimization, the architecture optimization step includes Reinforcement Learning-based [17–19], Evolution Algorithm-based [23–25, 28], and Differentiable-based approaches [14, 15, 29]. Reinforcement Learning-based idea behind is to investigate the reinforcement learning agent to place the circuit components and evaluate the composing circuit to reach the state-of-art human-designed ansatz. Evolution Algorithm-based approaches search architectures with the help of evolutionary algorithms. The basic idea of differentiable-based approaches is to relax the discrete architecture space to a continuous domain that enables us to employ gradient-based optimization methods for searching architecture.

Moreover, additional research is carried out to complete the task of quantum architecture search [30–34]. In Ref. [31], they propose a QAS evaluation protocol on basic tasks, making unified evaluation possible. This approach focuses on vanilla search methods without coupling with the domain prior. In Ref. [32], they present a new deep reinforcement learning-based QAS method, called Trust Region-based PPO with Rollback for QAS (QAS-TR-PPO-RB), to automatically build the quantum gates sequence from the density matrix only. In Ref. [33], they believe current QAS algorithms need to calculate the ground-truth performances of a large number of quantum circuits during the searching process, especially for large-scale quantum circuits, which is very time-consuming. Therefore, they propose a predictor based on a graph neural network (GNN), which can largely reduce the computational complexity of the performance evaluation and accelerate the QAS algorithm by estimating circuit performance directly based on their structures using a predictor trained on a set of labeled quantum circuits. In Ref. [34], aiming to address the limitations identified in [33], the authors introduce GSQAS, a self-supervised graph-based QAS that trains a predictor through self-supervised learning. Specifically, they first pretrain a graph encoder using a well-designed pretext task on a large number of unlabeled quantum circuits, aiming to generate meaningful representations of quantum circuits. In Ref. [30], they believe current Predictor-based QAS algorithms train a single predictor to fit the entire circuit space using limited training samples, which is inefficient and unnecessary, as QAS aims to identify the optimal quantum circuit. In Ref. [30], they propose a

progressive PQAS with active learning (PQAS-AL), which gradually trains more precise predictors for subspaces where high-performance circuits reside.

2.2 Search strategy based on evolutionary algorithms

Evolutionary Algorithm firstly uniformly samples the gates and generates K candidate circuits, then independently minimizes the loss function L within T iterations. Then, after the coarse-grained optimization, the evolutionary QAS chooses the top candidate circuit by ranking the others based on performance.

$$C = \arg\min_{j \in K} L(\theta_j, Z_j, \varepsilon_j) \tag{1}$$

where θ_j is the parameter used by the jth quantum circuit architecture, Z_j is the jth quantum circuit architecture, ε_j is the learning rate when optimizing the jth quantum circuit architecture. Finally, it continues to fine-tune the parameters of C and to search the final architecture. The framework is shown in Fig. 2.

To improve the efficiency and performance of the ranking stage, it can also employ evolutionary algorithms such as non-dominated sorting genetic algorithm II (NSGA-II) [35] instead of uniform sampling. Utilizing the parameter-sharing strategy can effectively reduce computational resources. NSGA-II, as employed in Ref. [1], utilizes an evolutionary algorithm to streamline the ansatz ranking process. The intuition behind employing NSGA-II is actively searching potential ansatzes with good performance instead of uniformly sampling ansatzes from all possible circuit architectures. Note that several recent studies, e.g., Refs [23, 25] have directly utilized the evolutionary and multi-objective genetic algorithms to complete ansatz design.

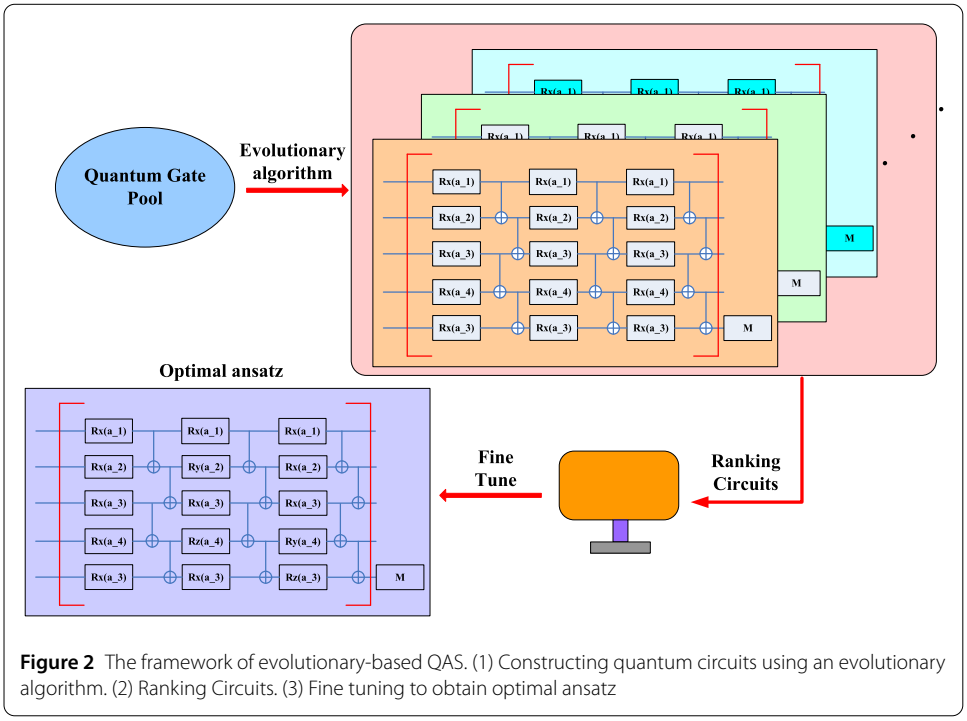

Figure 2 The framework of evolutionary-based QAS. (1) Constructing quantum circuits using an evolutionary algorithm. (2) Ranking Circuits. (3) Fine tuning to obtain optimal ansatz

2.3 Symmetric pruning in quantum circuit

Quantum neural networks (QNN) with symmetric ansatz only demand a polynomial number of trainable parameters and the circuit depth with the problem size to achieve a fast convergence rate [26, 36–44]. In the work of Sauvage et al., they build spatial symmetries into parameterized quantum circuits for faster training [37]. Wang et al. devise a Symmetric Pruning (SP) scheme to automatically tailor a symmetric ansatz from an asymmetric one with improved trainability and applicability [26]. The symmetric pruning algorithm proposed by Wang et al. is mainly divided into three steps: System symmetry. Structure symmetry. Spatial symmetry. SP assigns the system symmetry to by ansatz removing the redundant parameterized gates and the two-qubit gates associated with the last m qubit wires. SP assigns the structure symmetry on an ansatz by removing specific the single-qubit gates and two-qubit gates. SP assigns the spatial symmetry on ansatz by correlating the single-qubit parameterized gates on the equivalent qubits or the two-qubit parameterized gates on the equivalent qubit-pairs.

3 Approach

3.1 SuperCircuit construction and training

The initial stage of the Continuous Evolution for Efficient Quantum Architecture Search (CEQAS) involves SuperCircuit construction and training. Ref. [1] for the method of SuperCircuit construction. In the following, we elaborate on the essential details. During SuperCircuit construction and training, SuperCircuit has two important roles, which are constructing the ansatz pool *SupC* and parameterizing each ansatz in *SupC* via the specified parameter-sharing strategy. In other words, SuperCircuit defines the search space, which subsumes all candidate ansatzes, and the candidate ansatzes in *SupC* are evaluated through inheriting parameters from the SuperCircuit. Rather than training numerous separate ansatzes from scratch, QAS trains SuperCircuit just once, which significantly cuts down the search cost. Refs. [1, 20] use an approach for SuperCircuit training that consists of sampling subcircuits, training these subcircuits, and then updating parameters in the parameter space to complete the SuperCircuit training. This process also includes updating the parameter space, as illustrated in Fig. 3. The main goal of the parameter-sharing is to reduce the parameter space, which enhances QAS's learning performance while staying within reasonable memory and runtime limits. Intuitively, this strategy correlates parameters among different ansatzes in *SupC* based on a specified rule. CEQAS trains SuperCircuit in the same way as QAS. The details of how to build a SuperCircuit are described in the Experiments section.

3.2 Continuous evolution for quantum architecture search

The second phase of Continuous Evolution for Efficient Quantum Architecture Search involves evolutionary search. As for the Evolutionary search procedure, we use the evolution algorithm together with the non-dominated sorting strategy. The non-dominated sorting strategy has been introduced in the NSGA-II [35]. The essential details are described in Fig. 4.

Denote $\{N_1, \ldots, N_P\}$ as P different ansatzes and $\{F_1, \ldots, F_M\}$ as M different measurements we want to minimize or maximize. The measurements, for example, the number of two-qubit parameterized gates, and accuracy, could have some conflicts, which increase the difficulty in discovering an optimal solution that minimizes or maximizes all these metrics.

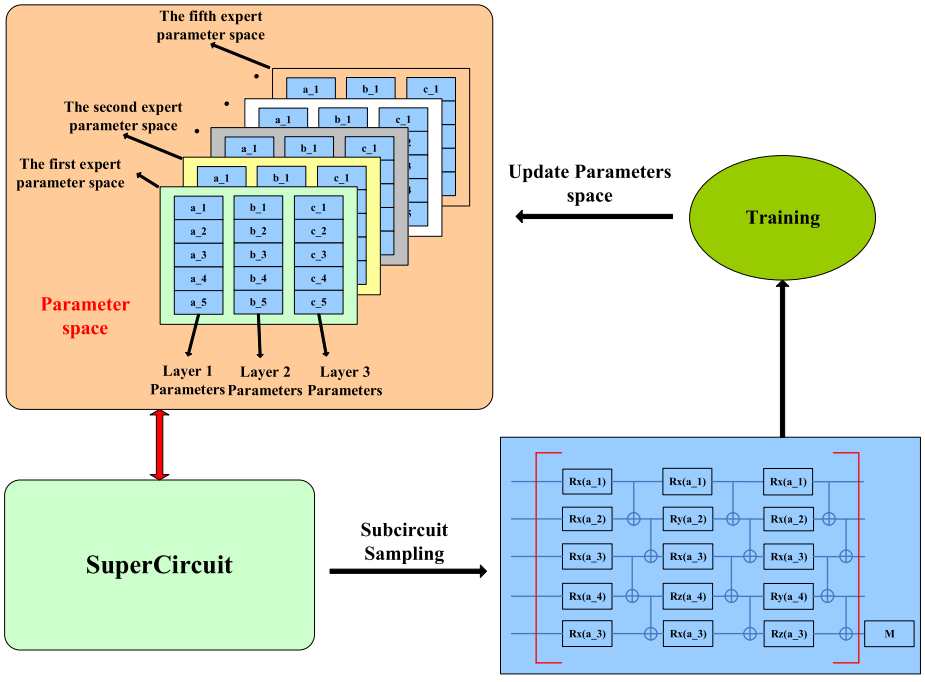

Figure 3 Training and parameter updating for SuperCircuit. (1) Subcircuit sampling and training. (2) Updating parameters space

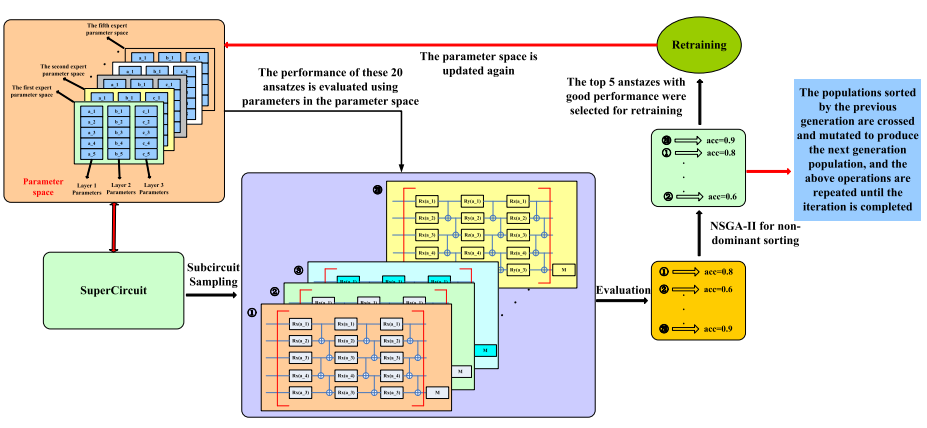

Figure 4 Schematic diagram of the detailed process of Continuous Evolution for Quantum Architecture Search. (1) Subcircuit sampling. (2) Ansatzes inherits parameters and performs performance evaluation. (3) NSGA-II for non-dominant sorting. (4) Retraining and updating parameter space

In practice, N_i dominates N_j if two conditions are satisfied: (1) for any of the measurements, the performance of N_i is not worse than that of N_j. (2) the model N_i behaves better than N_j on at least one measurement. Formally, the definition of domination can be summarized as below.

Definition 1 Considering two ansatzes N_i and N_j, and a series of measurements $\{F_1, \ldots, F_M\}$ we want to minimize. If

$$F_k(N_i) \leq F_k(N_j), \quad \forall k \in \{1, \ldots, M\} \tag{2}$$

$$F_k(N_i) < F_k(N_j), \quad \exists k \in \{1, \ldots, M\} \tag{3}$$

N_i is said to dominate N_j, i.e., $N_i \preceq N_j$.

According to the above definition, if N_i dominates N_j, N_j can be replaced by N_i during the evolution procedure since N_i performs better in terms of at least one metric and not worse on other metrics.

By utilizing this approach, we can choose a set of excellent ansatzes from the population in the current generation. These ansatzes can then be employed to update the corresponding parameters in the SuperCircuit.

In the context of NSGA-II for CEQAS, NSGA-II samples S_N ansatzes, evaluates their performance (e.g., accuracy in binary classification), and subsequently employs fast non-dominated sorting to rank the performance of the ansatzes. The process is illustrated in Fig. 4. For example, ansatzes with high accuracy are placed at the front of the population and then retrained to further optimize the parameter space. Through crossover, the mutation generates the next generation ansatzes. The above ranking and retraining are repeated until the iteration is completed.

3.3 Structure symmetric pruning

Suppose the problem Hamiltonian is $\tilde{H} = H \otimes \mathrm{II}^{\otimes m}$, where $H = \sum_{j=1}^{q} \alpha_j H_j$, α_j is the real coefficient and H_j is the tensor product of Pauli matrices on n qubits, Structure Symmetric Pruning builds the symmetric ansatz of $\tilde{H}$ with two primary steps, i.e., initialization and symmetry identification. A symmetry S of a Hamiltonian $\tilde{H}$ is a unitary operator leaving $\tilde{H}$ invariant,

$$S\tilde{H}S^{\dagger} = \tilde{H} \tag{4}$$

All of these symmetries form a symmetry group S where for any two symmetries $S_1, S_2 \in S$, their compositions $S_1 \circ S_2$ or $S_2 \circ S_1$ and their inverses S_1^{-1} and S_2^{-1} are also symmetries in S.

Suppose that the initialized asymmetric ansatz is $U(\theta)$, Structure Symmetric Pruning adopts the following method to tailor this ansatz to obey the above symmetries. *Structure symmetry.* The structure symmetry S_{str} refers to the symmetry for the effective Hamiltonian H, which satisfies

$$S_{str} H S_{str}^{\dagger} = H \tag{5}$$

Moreover, an ansatz $V(\theta)$ is said to be structure symmetric to the problem Hamiltonian H if there exists a non-trivial symmetry S_{str} (i.e., not the identity operation) and $\theta \in \Theta \backslash \{0\}$ such that $S_{str} V(\theta) S_{str}^{\dagger} = V(\theta)$. A feasible solution of constructing the structure symmetric ansatz is restricting the corresponding ansatz design that only contains the Pauli terms of H. Given the pruned ansatz U_{Pr} returned by Structure Symmetric Pruning assigns the

Algorithm 1 Structure Symmetric Pruning (SSP)

Input: Problem Hamiltonian $\tilde{H} = (\sum_{j=1}^{q} \alpha_j H_j) \otimes \mathrm{II}^{\otimes m}$, ansatz design A for CEQAS output and the parameter space Θ in Fig. 4.

Output: Pruned ansatz design A_{Pr} and parameter space Θ_{Pr}.

1 Initialize an asymmetric ansatz via A and Θ.

2 for $i = 1$ to epochs:

3 Sample one new ansatz s_A from A, Obtain parameter Θ_s in Θ;

4 Compute objective function $L = \frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} (\tilde{y}^{(i)}(s_A, x^{(i)}, \Theta_s) - y^{(i)})^2$;

5 Optimize the objective function;

6 Update Θ_s and Θ.

7 Structure Symmetry identification: Remove the gates such that the pruned ansatz design.

8 Get the pruned ansatz design s_p, Obtain parameter Θ_p in Θ.

9 for $i = 1$ to five:

10 Compute objective function $L = \frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} (\tilde{y}^{(i)}(s_p, x^{(i)}, \Theta_s) - y^{(i)})^2$;

11 Optimize the objective function;

12 Update Θ_p and Θ.

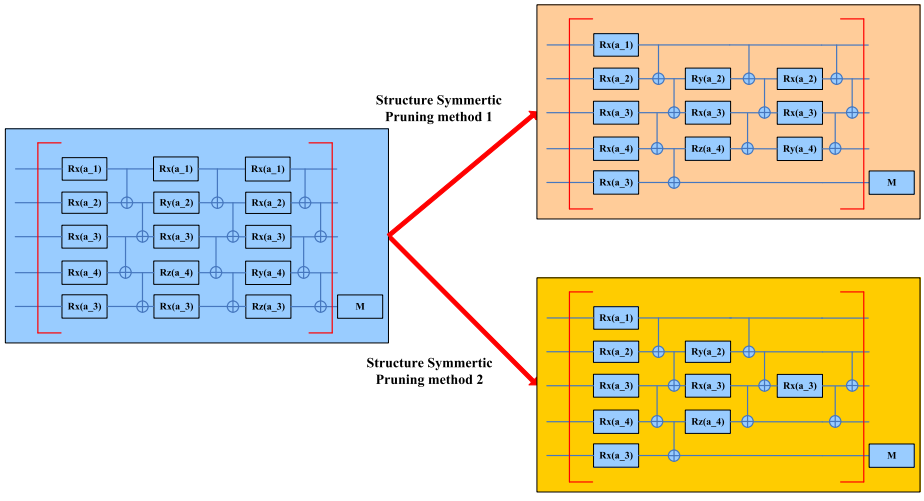

Figure 5 Schematic of Structure Symmetric pruning. Two Structure Symmetric Pruning methods are shown in the figure, both of which belong to symmetric pruning

structure symmetry on it by removing specific the single-qubit gates and the two-qubit gates [26]. Algorithm 1 provides a summary of the Structure Symmetric Pruning procedures and Fig. 5 depicts its schematic representation.

4 Experiments

In this section, we describe our experiments' implementations of Continuous Evolution for Quantum Architecture Search and Structure Symmetric Pruning. Subsequently, we select a few ansatzes exhibiting relatively optimal performance from the output CEQAS, retrain them, and proceed with Structure Symmetric Pruning to assess the performance

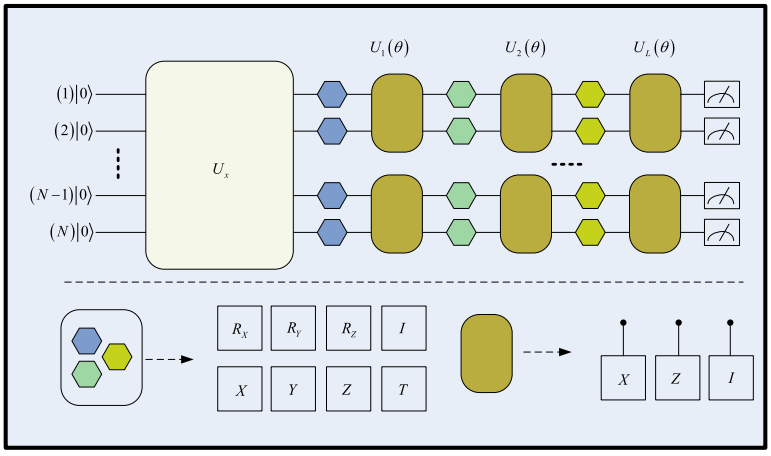

Figure 6 Paradigm of the SuperCircuit [1]

disparities before and after the pruning procedure. Here we apply CEQAS to achieve a binary classification task under the noiseless which is consistent with [1].

In Fig. 6, CEQAS and QAS use the same scheme to construct SuperCircuit, which defines the ansatze pool to be searched. Parameterizes each ansatz in SuperCircuit via the specified parameter-sharing strategy. All possible single-qubit gates are highlighted by hexagons and two-qubit gates are highlighted by the brown rectangle. The unitary U_x refers to the data encoding layer.

4.1 Experimental settings

Denote D as the synthetic dataset, where its construction rule follows [1]. The dataset D contains $n = 500$ samples. For each example $\{x^{(i)}, y^{(i)}\}$, the feature dimension of the input $x^{(i)}$ is 5 and the corresponding label $y^{(i)} \in \{0, 1\}$ is binary. At the data preprocessing stage, we split the dataset D into the training set D_{tr}, validation set D_{va}, and test set D_{te} with size $n_{tr} = 200$, $n_{va} = 100$, $n_{te} = 200$. The explicit form of the objective function is

$$L = \frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} \left(\tilde{y}^{(i)}\left(SupC, x^{(i)}, \theta\right) - y^{(i)}\right)^2 \tag{6}$$

where $\{x^{(i)}, y^{(i)}\} \in D_{tr}$ and $\tilde{y}^{(i)}(SupC, x^{(i)}, \theta) \in [0, 1]$ is the output of the quantum classifier (i.e., a function taking the input $x^{(i)}$, the SuperCircuit $SupC$, and the trainable parameters θ). The training (validation and test) accuracy is measured by $\sum_i 1_{g(\tilde{y}^{(i)})=y^{(i)}}/n_{tr}$ ($\sum_i 1_{g(\tilde{y}^{(i)})=y^{(i)}}/n_{va}$ and $\sum_i 1_{g(\tilde{y}^{(i)})=y^{(i)}}/n_{te}$) with $g(\tilde{y}^{(i)})$ being the predicted label for $x^{(i)}$.

The construction and training methods of SuperCircuit are consistent with those used in Ref. [1], except that Ref. [1] utilizes 3 qubits for quantum circuits, whereas this work employs 5 qubits to facilitate the verification of Structure Symmetric Pruning. In the experiments, the single-qubit gates are R_X, R_Y, R_Z, the two-qubit gates are CNOT gates.

About CEQAS hyperparameters, the circuit depth for all SuperCircuits is set to $L = 3$. The search space of CEQAS is formed by two types of quantum gates. Specifically, at each layer $U_l(\theta)$, the parameterized gates are fixed to be the rotational quantum gate along X-axis R_X, Y-axis R_Y, Z-axis R_Z. For the two-qubit gates, denoted the index of five qubits as (0, 1, 2, 3, 4), CEQAS explores whether applying CNOT gates to the qubits pair (0, 1), (1,

2), (2, 3), (3, 4) or not. Hence, the size of SuperCircuit equals to $|SupC| = 16^3$. The number of sampled ansatzes for ranking is set as $K = 4000$. The setting $K \approx 16^3$ enables us to understand how the number of epochs T, and the search strategies affect the learning performance of different ansatzes in the ranking stage. When the number of SuperCircuits is $W = 5$ (i.e. the number of expert parameter spaces is 5) QAS performs better [1], therefore, $W = 5$ is still used as the hyperparameter in this work. About the number of epochs T, in the experiments we set $T = 10$, $T = 100$, $T = 400$, $T = 1000$ to demonstrate that using CEQAS can achieve superior search performance compared to QAS without the need for excessive training epochs.

Evolution Details. After training the SuperCircuit and updating its parameters, the subsequent step involves the Continuous Evolution Search stage. In this study, NSGA-II serves as the evolution algorithm for the search stage, facilitating both evolutionary search and retraining. Both the crossover ratio and mutation ratio are set to 0.25, and we randomly generate new architectures with a probability of 0.5. Following non-dominated sorting, the top 20% are chosen for retraining, thereby optimizing the parameter space.

To execute Structure Symmetric Pruning, we manually pick the top 20 performing ansatzes for retraining from the CEQAS output. The effectiveness of Structure Symmetric Pruning is then verified by comparing the performance changes of these ansatzes before and after pruning.

All numerical simulations are implemented in Python in conjunction with the PennyLane [45] and the Qiskit packages [46].

4.2 Results of continuous evolution for quantum architecture search

The parameter settings for NSGA-II include a total of 4000 search ansatzes, with a population size of 50 and 80 generations. To obtain better output results from CEQAS, we set $W = 5$ and the number of retrained ansatzes to account for 20% of the population size. The performance of CEQAS with four different epochs is exhibited in Fig. 7. In particular, CEQAS with $T = 100$ attains the best performance, where the validation accuracy for most ansatz concentrates on 80–90% and 90–100%, highlighted by the orange bar. This also indicates that by increasing the number of epochs, the performance of CEQAS does not get a significant improvement, especially in the number of ansatzes with high accuracy. The underlying reason for this phenomenon could potentially stem from the irrational selection of the number of ansatzes chosen for the retraining of the Pareto boundary during the evolution process.

By increasing the number of epochs, the performance of QAS is slightly improved as [1]. Figure 8 shows the number of ansatzes of each accuracy output by CEQAS and QAS after training $T = 10$, $T = 100$, $T = 400$, $T = 1000$ epochs. In Fig. 8(a) $T = 10$, the number of ansatzes with validation accuracy of 80–100% in the CEQAS output is relatively higher than that of QAS, but the number of ansatzes obtained by QAS with validation accuracy of 90–100% is higher than that of CEQAS. This suggests that with a limited number of epochs for training the SuperCircuit, the parameter space may require further optimization. In Fig. 8(b) $T = 100$, the number of ansatzes with validation accuracy of 80–90% and 90–100% in CEQAS search results is relatively higher than that of QAS. Figure 8(c) $T = 400$, only the number of ansatz validation accuracy in the 90–100% is higher in the search results of CEQAS than in QAS. Figure 8(d) $T = 1000$, the number of ansatzes with validation accuracy of 70–100% in CEQAS search results is higher than that of QAS. This

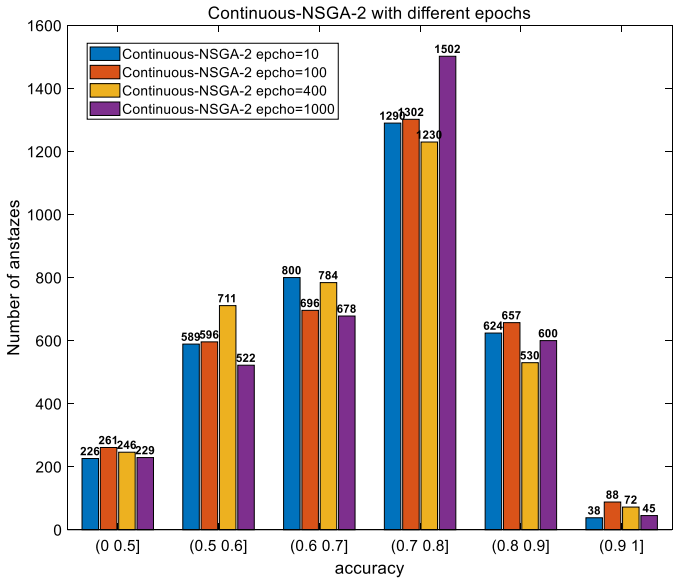

Figure 7 SuperCircuit is trained with different epochs, the number of search ansatz is 4000, and CEQAS outputs the number of ansatzes for each accuracy range

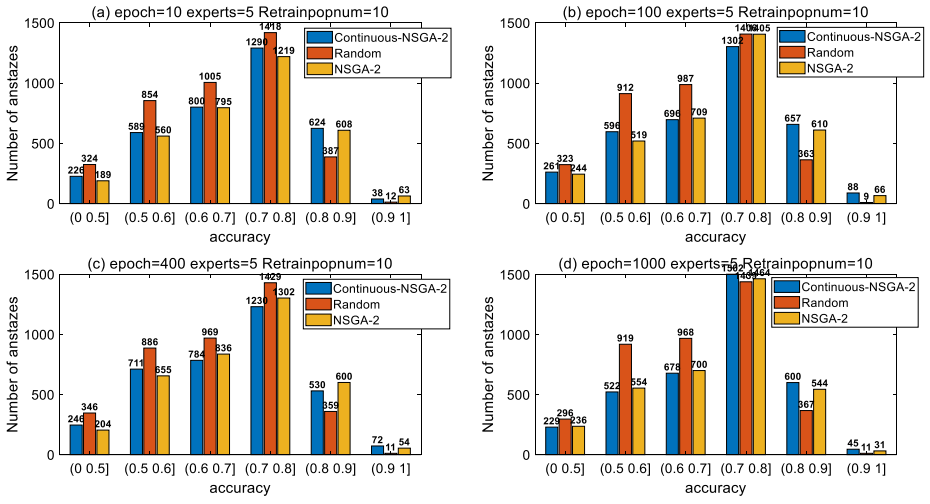

Figure 8 Different search strategies affect the number of ansatzes with different accuracy obtained from the search. (1) "Continuous-NSGA-2" stands for the use of the CEQAS method. (2) "NSGA-2" indicates that the SuperCircuit method is trained using QAS, and NSGA-2 is used in the search phase. (3) "Randon" represents the method of training SuperCircuit using QAS, and the search phase uses random search

suggests that the parameter space can be further optimized through the utilization of CEQAS. However, it is worth noting that as the T increases, the computational cost of training SuperCircuit increases. In the actual search process, we do not require an excessive number of training epochs to accomplish SuperCircuit training. As shown in Fig. 7, T = 100, the number of ansatzes with high accuracy (80-100%) ansatz is already better than that of T = 400, T = 1000. In fact, completing the SuperCircuit training does not require a large number of training epochs according to CEQAS. This is attributed to the utilization

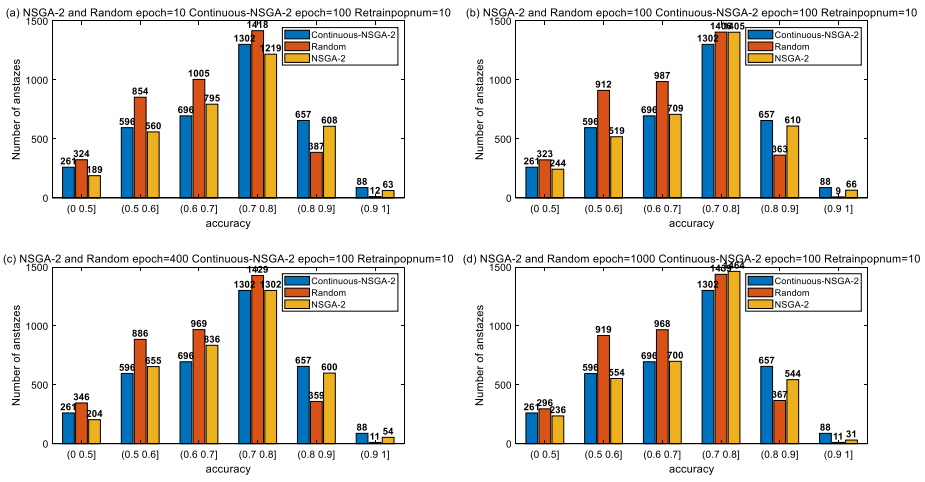

Figure 9 Comparison of search results CEQAS with $T = 100$ vs. QAS with $T = 10$, $T = 100$, $T = 400$, $T = 1000$. (a) CEQAS with $T = 100$, QAS with $T = 10$. (b) CEQAS with $T = 100$, QAS with $T = 100$. (c) CEQAS with $T = 100$, QAS with $T = 400$. (d) CEQAS with $T = 100$, QAS with $T = 1000$

of the "optimizing the parameter space while searching" method during the search stage: (1) the results of the Pareto front (which can be understood as the approximate optimal solution) can be accurately retained, and the parameters in the parameter space of the SuperCircuit can be updated more accurately when training again; (2) the NSGA-II can be used to better generate the next generation of populations by using the results of the previous generation population, better structural optimization operations can be completed; (3) in the final search results, the search method of "optimizing parameter space while searching" can more efficiently provide a series of ansatzes on the Pareto front.

In Fig. 9, we further demonstrate the superiority of CEQAS with $T = 100$ search results. CEQAS with $T = 100$ compares to QAS with $T = 10$, $T = 100$, $T = 400$ and $T = 1000$, the validation accuracy for most ansatzes concentrates on 80–90% and 90–100%. In Fig. 9(b)(c)(d) it is more evident that the CEQAS can search for more quantum circuit architectures with better performance (mainly reflected in the number of ansatzes with the accuracy of 80–90% and 90–100%) through the search method of "optimizing parameter space while searching."

How many ansatzes with good performance in the population are selected for retraining to complete the parameter space parameter optimization? Here we give some experimental results: when the search number is 4000, train epochs $T = 100$, the population size is 50, and the generation number is 80, the number of retrained ansatzes changes from 10, 20, 30, and 40 (20%, 40%,60%, 80% of the population), the result is shown in Fig. 10.

With the increase of the number of retrained ansatzes, the number of low-accuracy ansatzes gradually decreases, mainly reflected in the number of ansatzes in the 0–50% accuracy, and the number of high-accuracy ansatzes gradually increases, mainly reflected in the number of ansatzes in the 80–100% accuracy. This observation indicates that as the number of retrained ansatzes increases, the parameter space can be further optimized, but at the cost of increasing the processing cost of the search process.

The above experiments are performed under the noiseless setting. In Appendix A, we give the results of the experiments under the noisy setting. Then, we apply CEQAS to find

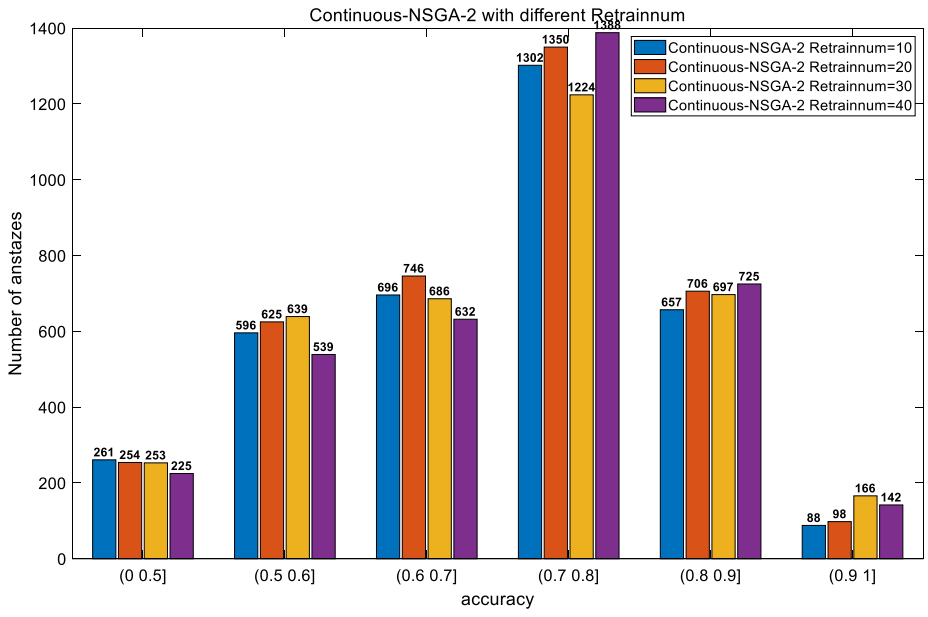

Figure 10 The effect of the number of retrained ansatzes on CEQA performance

the ground state energy of the Hydrogen molecule [47, 48] in Appendix B to demonstrate the effectiveness of our method.

4.3 Quantum structure symmetric pruning

Symmetry is a very important concept in physics especially in quantum mechanics. Applying symmetry in quantum circuits of QNNs enables an improved trainability of QNNs, including alleviating the barren plateaus and reducing the number of parameters and the circuit depth [26]. Therefore, after retraining the quantum circuit with good performance searched by CEQAS, the above two Structure Symmetric Pruning methods are used in Fig. 5, reducing the number of parameters and improving the trainability of the quantum circuit. In this section, we present the experimental results of Quantum Structure Symmetric Pruning.

Structure Symmetric Pruning method 1 In the CEQAS output, we selected the top 20 ansatzes with the highest performance for retraining. Subsequently, Structure Symmetric Pruning was carried out. To elucidate a clearer retraining process, only the retraining of 4 ansatzes is depicted in Fig. 11, while the retraining of all 20 ansatzes is displayed in Appendix C (Fig. 19). On the one hand, we can verify that the ansatzes obtained by CEQAS can achieve better performance after individual training from scratch. On the other hand, we can verify the effectiveness of Structure Symmetric Pruning on ansatzes. Doing this work will allow us to show that CEQAS and Structure Symmetric Pruning can be well connected to a certain extent. Figure 11 displays the training accuracy and test accuracy of the top-performing ansatzes after training from scratch (100 epochs), as well as the training accuracy, test accuracy, and ansatzes (utilizing Structure Symmetric Pruning method 1) trained for an additional 5 epochs.

From Fig. 11, it is evident that the accuracy of the top-performing ansatzes achieved by CEQAS reaches convergence to 1 after training from scratch. In Fig. 11, a more detailed re-

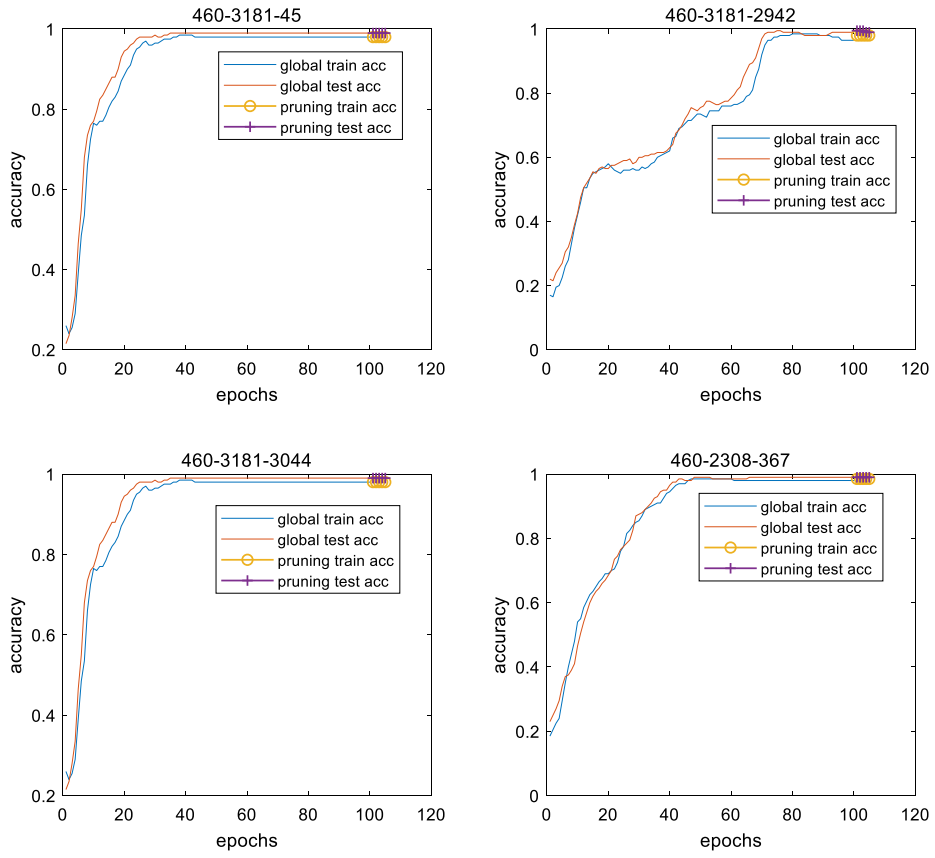

Figure 11 The training accuracy, test accuracy of quantum circuit retraining, and the training accuracy, test accuracy of the quantum circuit (trained for 5 additional epochs) after using the structure symmetric pruning method 1. Different xxx-xxx-xxx represent different quantum circuit structures. *global train/test acc* represents the accuracy change curve of the quantum circuit during training with the original structure, while *pruning train/test acc* represents the accuracy change curve of the quantum circuit after pruning

training process is provided for 4 ansatzes, while the retraining process for all 20 ansatzes is depicted in Appendix C (Fig. 19). The average test accuracy of the 20 ansatzes converged to 0.9830, which indicates that the ansatzes with good performance in the CEQAS search results can achieve better performance after training from scratch. After the ansatzes were trained again, we perform Structure Symmetric Pruning method 1 on these ansatzes. Figure 11's results demonstrate that, both before and after using structure symmetry pruning method 1, ansatzes' performance remained almost unaltered. The *pruning train acc* and *pruning test acc* demonstrated a similar trend to the *global train acc* and *global test acc*. The trend of change in training accuracy and test accuracy post-pruning aligns with the values and trends observed in training accuracy and test accuracy before pruning. The average test accuracy of the 20 ansatzes after utilizing Structure Symmetric Pruning method 1 continues to converge at 0.9830. Upon the completion of 5 more training epochs on the pruned ansatzes, the average test accuracy converges to 0.9875. The number of parameters has been decreased from 15 to 11, a reduction of about $\frac{1}{3}$. This reduction effectively minimizes the parameter count without compromising the performance of the ansatzes.

Figure 12 illustrates the training accuracy and test accuracy of the ansatzes before and after performing random pruning. The retraining process for four ansatzes is depicted

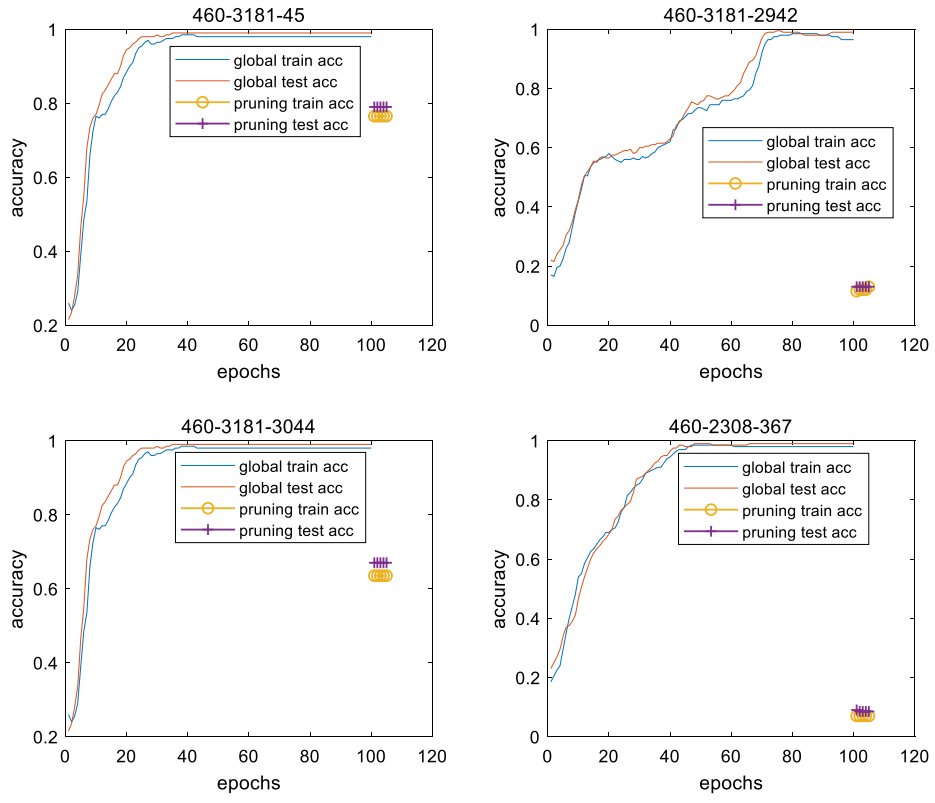

Figure 12 The training accuracy, test accuracy of quantum circuit retraining, and the training accuracy, test accuracy of the quantum circuit (trained for 5 additional epochs) after using the random pruning while maintaining 11 parameters. Different xxx-xxx-xxx represent different quantum circuit structures. *global train/test acc* represents the accuracy change curve of the quantum circuit during training with the original structure, while *pruning train/test acc* represents the accuracy change curve of the quantum circuit after pruning

Table 1 A comparison of the accuracy (Structure Symmetric Pruning method 1 and random pruning)

	Structure Symmetric Pruning	Random pruning
Average test accuracy	0.9875	0.8655

in Fig. 12, while the retraining process for 20 ansatzes is detailed in Appendix C Fig. 20. Here, the random pruning of ansatzes maintains an equal number of parameters as the ansatzes after pruning with Structure Symmetric Pruning Method 1. Most of the ansatzes exhibit a significant performance change, with both training accuracy and test accuracy decreasing to below 0.7. The average test accuracy convergence of the remaining ansatzes that were not significantly altered after random pruning was 0.86 (shown in Appendix C Fig. 20). This observation indicates that utilizing Structure Symmetric Pruning is more effective than random pruning in preserving the performance of ansatzes while retaining the same number of parameters. Structure Symmetric Pruning is more suitable for the output of CEQAS. The accuracy of ansatzes using Structure Symmetric Pruning method 1 and random pruning is compared in Table 1.

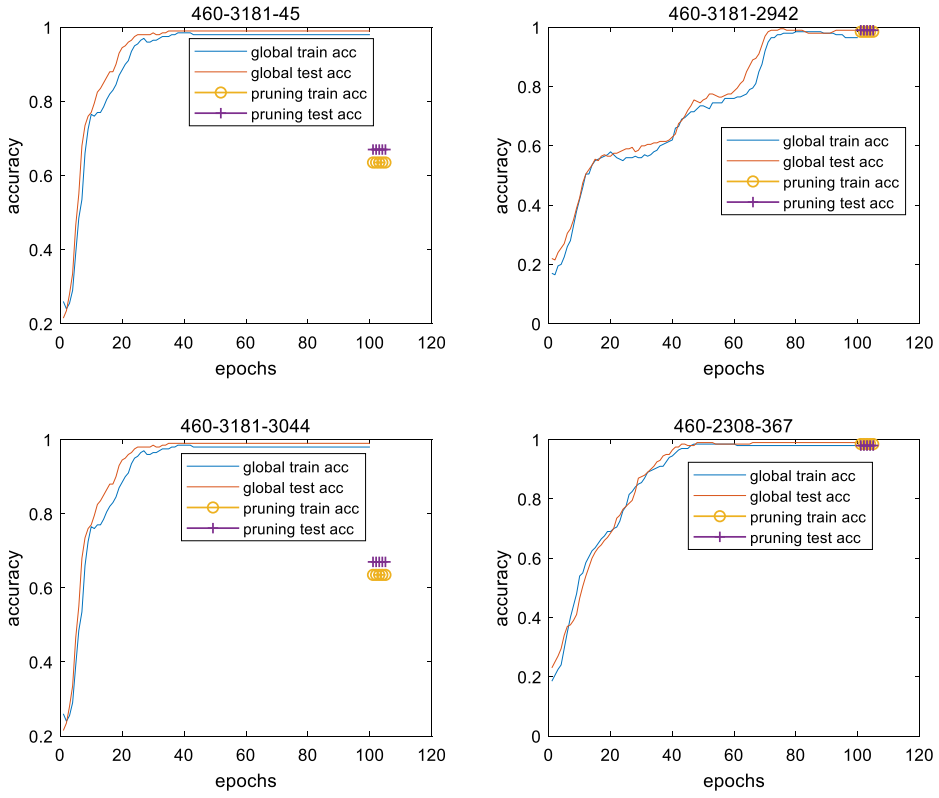

Figure 13 The training accuracy, test accuracy of quantum circuit retraining, and the training accuracy, test accuracy of the quantum circuit (trained for 5 additional epochs) after using the structure symmetric pruning method 2. Different xxx-xxx-xxx represent different quantum circuit structures. *global train/test acc* represents the accuracy change curve of the quantum circuit during training with the original structure, while *pruning train/test acc* represents the accuracy change curve of the quantum circuit after pruning

Structure Symmetric Pruning method 2 The retraining process for 4 ansatzes is visualized in Fig. 13, while the process for all 20 ansatzes is detailed in Appendix C Fig. 21. This representation illustrates the performance before and after implementing Structure Symmetric Pruning Method 2. From Fig. 13, while some ansatzes have experienced changes in their test accuracy following pruning, none of them have decreased to below 0.7. Structure Symmetric Pruning method 2 preserves 60% of the parameters in the original ansatz. The average test accuracy converges to 0.9305, which is 0.0525 lower than the average test accuracy before pruning. The performance of the ansatzes performed by Structure Symmetric Pruning method 2 is worse than that of the ansatzes performed by Structure Symmetric Pruning method 1, but after all, it reduces more parameters in ansatzes. After 5 additional epochs of training following the implementation of Structure Symmetric Pruning Method 2 on the ansatzes, the average exact test accuracy converges to 0.9392. This result represents a decrease of 0.0438 compared to the accuracy of 0.9830 before pruning. Figure 14 shows the performance curve of ansatzes with random pruning while maintaining 9 parameters.

Figure 14 illustrates the outcomes of random pruning applied to ansatzes, maintaining an equal number of parameters as Structure Symmetric Pruning Method 2. In Fig. 14, the retraining process for a selective group of four ansatzes is depicted to ensure a clear

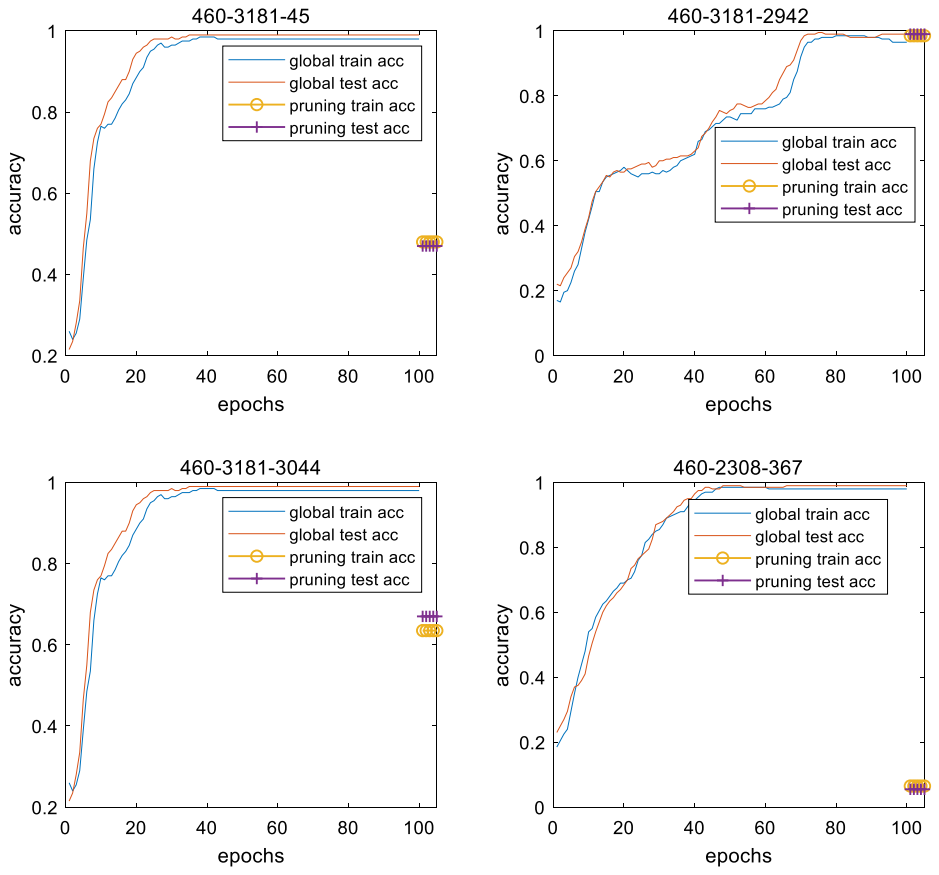

Figure 14 The training accuracy, test accuracy of quantum circuit retraining, and the training accuracy, test accuracy of the quantum circuit (trained for 5 additional epochs) after using the random pruning while maintaining 9 parameters. Different xxx-xxx-xxx represent different quantum circuit structures. *global train/test acc* represents the accuracy change curve of the quantum circuit during training with the original structure, while *pruning train/test acc* represents the accuracy change curve of the quantum circuit after pruning

Table 2 Another comparison of the accuracy (Structure Symmetric Pruning method 2 and random pruning)

	Structure Symmetric Pruning	Random pruning
Average test accuracy	0.9392	0.8386

display, while the detailed process for all 20 ansatzes can be found in Appendix C Fig. 22. For the remaining ansatzes that have not undergone significant changes, the average accuracy converges to 0.8386, aligning with the number of parameters compared to Structure Symmetric Pruning Method 2. Notably, the performance of ansatzes declines further after random pruning. Table 2 provides a comparison of the accuracy between ansatzes using Structure Symmetric Pruning Method 2 and random pruning.

Following the application of either Structure Symmetric Pruning Method 1 or Method 2 on the ansatzes generated by CEQAS, and subsequent retraining of the pruned ansatzes, the average test accuracy demonstrates an increase compared to before retraining. This observation, as depicted in Fig. 19 and Fig. 21, highlights that Structure Symmetric Pruning contributes to maintaining the trainability of ansatzes.

In terms of overall adaptability, there is minimal variance between the test accuracy of all ansatzes subjected to Structure Symmetric Pruning Method 1 and the test accuracy of the ansatzes before pruning. Some ansatzes even demonstrate improved performance following pruning. The accuracy of certain ansatzes (6 out of 20) that underwent Structure Symmetric Pruning Method 2 significantly differs from their accuracy before pruning. This suggests that employing Structure Symmetric Pruning Method 1 in ansatzes can better maintain consistent performance without significant changes and enhance adaptability when pruning the ansatzes generated by CEQAS, as Structure Symmetric Pruning Method 1 retains a greater number of parameters in the ansatzes. Regarding the number of parameters in ansatzes, the performance may remain relatively unaffected if a substantial number of parameters are retained following the application of Structure Symmetric Pruning, exemplified by Structure Symmetric Pruning method 1. If ansatzes retain a limited number of parameters after undergoing Structure Symmetric Pruning, the performance of the ansatzes will be somewhat affected.

4.4 Runtime complexity

We analyze the runtime complexity of CEQAS. In particular, at the first step, the setup of SuperCircuit, i.e., configuring out the ansatz pool and the correlating rule, takes $O(1)$ runtime. In the second step, CEQAS proceeds T iterations to optimize trainable parameters. The runtime cost of QAS at each iteration scales with $O(d)$, where d refers to the number of trainable parameters in Eq. (1). Such cost origins from the calculation of gradients via parameter shift rule, which is similar to the optimization of VQAs with a fixed ansatz. In the strategy of "optimizing the parameter space while searching," CEQAS proceeds T_R iterations to optimize trainable parameters again, where T_R refers to the number of retrained ansatzes. Therefore, the total runtime cost of the second step is $O(dTT_R)$. In the ranking step, CEQAS samples K ansatzes and compares their objective values using the optimized parameters. This step takes at most $O(K)$ runtime. CEQAS fine-tunes the parameters based on the searched ansatz with few iterations (i.e., a very small constant). The required runtime is identical to conventional VQAs, which satisfies $O(d)$. The total runtime complexity of CEQAS is hence $O(dTT_R + K)$.

Compared to the total runtime complexity $O(dT + K)$ in Ref. [1], our algorithm does increase the complexity of runtime, which is mainly reflected in the number of retrained ansatzes. The runtime complexity of our algorithm will grow relative to that in Ref. [1], while the number of trainable parameters d in quantum circuits and the number of retrained ansatzes T_R increase. The ansatzes with good performance output by CEQAS can be further refined using Structure Symmetric Pruning to reduce the number of gates within the circuit. For a quantum circuit with great performance obtained by CEQAS, Structure Symmetric Pruning can guarantee minimal changes in performance by reducing the gate parameters by about $\frac{1}{3}$. Utilizing quantum circuits with good performance, both from the method outlined in Ref. [1] and those obtained through CEQAS, we have applied Structure Symmetric Pruning to these circuits. We anticipate that the training process for these pruned circuits will be significantly faster compared to those that have not undergone this optimization technique. Studies of the same kind have also been carried out in the Refs. [26] and [37]. In further work, we will investigate further how much training time may be saved by utilizing Structure Symmetric Pruning in our CEQAS algorithm.

5 Conclusion

The SuperCircuit-based QAS methods excel at discovering high-performance models (Quantum Circuits) with high-performance [1, 20], However, there is still potential for optimizing the parameter space related to the SuperCircuit. For this reason, we propose a continuous evolution architecture search method, namely, CEQAS. During evolution, CEQAS maximally utilizes the learned knowledge in the latest evolution generation, such as architectures and parameters. A SuperCircuit is constructed with considerable single-qubit gates and two-qubit gates. Individuals are generated through the benchmark operations in the evolutionary algorithm. The non-dominated sorting strategy (NSGA-II) is used to select ansatzes for updating the parameter space of the SuperCircuit. The experiments demonstrate that CEQAS effectively generates multiple ansatzes on the Pareto front. Structure Symmetric Pruning can be applied to the ansatzes produced by CEQAS, effectively reducing the parameter count while maintaining ansatz performance within acceptable limits. There are still many issues to be answered in the CEQAS investigation. First, We admit that compared to the total runtime complexity $O(dT + K)$ in Ref. [1], our algorithm does increase the complexity of runtime. Our CEQAS aims to propose a strategy of "optimizing the parameter space while searching," which does indeed sacrifice runtime for better performance. For the searched quantum circuits, CEQAS may compensate for the complexity of runtime to some extent when they are trained from scratch after using Structure Symmetric Pruning. We also refer to Ref. [1] for the time in a noisy environment, and only do experimental verification in a specific noisy environment.

Our future work includes the following several directions. First, with every iteration, we shall investigate more effective methods for sampling ansatz. For example, we can consider using NSGA-III. Next, for the "Structure Symmetric Pruning" strategy of shrinking the parameter space, we still need to further consider the actual situation of this strategy in ground state energy estimation. We intend to investigate the impact of noise on CEQAS for the noisy environment experiment.

Appendix A: Simulation results for the classification task under the noisy setting

In this section, we give simulation results of CEQAS for the classification task under the noisy setting. The experimental parameters are set as described in 4.1 Experimental Settings. It is important to note that since these experiments are conducted in a noisy environment. The noisy environment we used was provided by the Qiskit package, which can approximately simulate the quantum gates error and readout error in 'Ibmq_ourense'. The error probabilities of the 1-qubit gate were set as 0.05, and the noise probabilities of the 2-qubit gate were set as 0.2. The setting in noisy simulation was also referred to [1].

More training epochs for the SuperCircuit could theoretically improve parameter space optimization and reduce the effect of noise. Therefore, we set the epochs $T = 400$. In Fig. 15, $T = 400$, the number of ansatzes with validation accuracy of 70–80% and 90–100% of in CEQAS search results is relatively higher than that of QAS. The count of ansatzes with validation accuracy ranging from 80% to 90% in the CEQAS search outcomes is nearly equivalent to that of QAS. This outcome further demonstrates that, even in a noisy environment, the performance of CEQAS surpasses that of QAS.

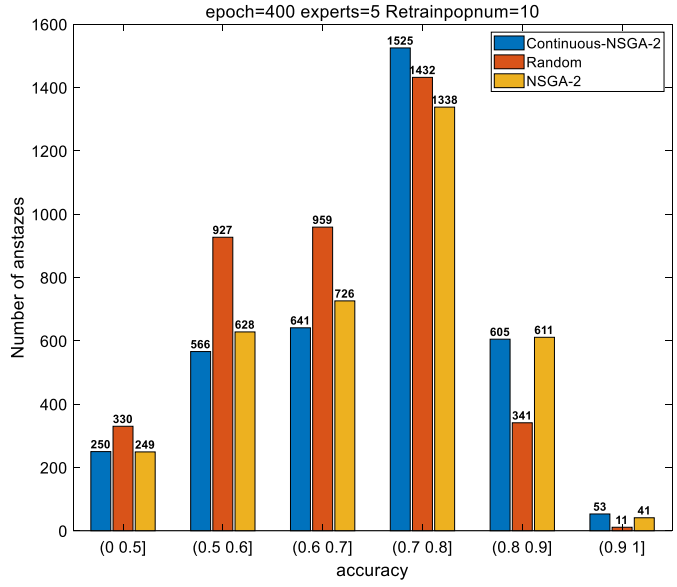

Figure 15 Simulation result for the classification task under the noisy setting. CEQAS and QAS with $T = 400$

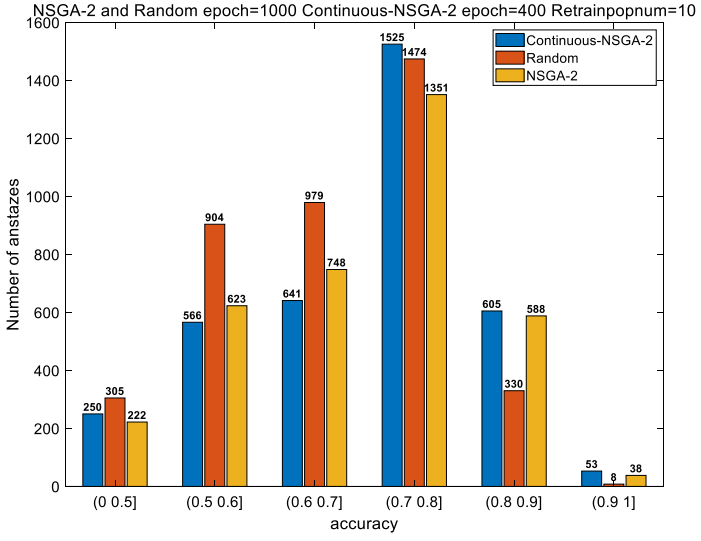

Figure 16 Comparison of search results CEQAS with $T = 400$, QAS with $T = 1000$ under the noisy setting

In Fig. 16, we further demonstrate the superiority of CEQAS with $T = 400$ search results. CEQAS with $T = 400$ compares to QAS with $T = 1000$, the validation accuracy for most ansatzes concentrates on 70–80%, 80–90% and 90–100%.

Despite a reduction in the count of ansatzes with validation accuracy between 80% and 90% in CEQAS search results compared to noiseless conditions, there remains a higher number of ansatzes achieving validation accuracy between 70% and 80% as opposed to QAS. These experimental outcomes in noisy conditions further validate the efficacy of our proposed method to a certain extent.

Appendix B: Simulation results for the ground state energy estimation of Hydrogen

Here we apply CEQAS to find the ground state energy of the Hydrogen molecule [47, 48] under the noiseless and noisy setting. The molecular hydrogen Hamiltonian is formulated as:

$$\begin{aligned} H_h = g + \sum_{i=0}^{3} g_i Z_i + \sum_{i=1,k=1,i<k}^{3} g_{i,k} Z_i Z_k \\ + g_a Y_0 X_1 X_2 Y_3 + g_b Y_0 Y_1 X_2 X_3 + g_c X_0 X_1 Y_2 Y_3 + g_d X_0 Y_1 Y_2 X_3 \end{aligned} \tag{7}$$

where $\{X_i, Y_i, Z_i\}$ denote the Pauli matrices acting on the i-th qubit and the real scalars g with or without subscripts are efficiently computable functions of the hydrogen–hydrogen bond length. In particular, the explicit form of the molecular hydrogen Hamiltonian H_h is:

$$\begin{aligned} H_h = &-0.042 + 0.178(Z_0 + Z_1) - 0.243(Z_2 + Z_3) + 0.171 Z_0 Z_1 \\ &+ 0.123(Z_0 Z_2 + Z_1 Z_3) + 0.168(Z_0 Z_3 + Z_1 Z_2) \\ &+ 0.176 Z_2 Z_3 + 0.045(Y_0 X_1 X_2 Y - Y_0 Y_1 X_2 X_3 - X_0 X_1 Y_2 Y_3 + X_0 Y_1 Y_2 X_3) \end{aligned} \tag{8}$$

The goal of the variational Eigen-solver is generating a parameterized wave-function $|\Psi(\theta)\rangle$ to achieve:

$$\min_{\theta} \left| \left\langle \Psi(\theta) \middle| H_h \middle| \Psi(\theta) \right\rangle - E_m \right| \tag{9}$$

The linear property of H_h in Eqn.(9) implies that the value $\langle \Psi(\theta)|H_h|\Psi(\theta)\rangle$ can be obtained by iteratively measuring $|\Psi(\theta)\rangle$ using Pauli operators in H_h, e.g., such as $|\langle \Psi(\theta)|\mathrm{II}_8 \otimes Z_0|\Psi(\theta)\rangle$, $|\langle \Psi(\theta)|X_0 Y_1 Y_1 X_3|\Psi(\theta)\rangle$. The lowest energy of H_h equals to $E_m = -1.136Ha$ where 'Ha' is the abbreviation of Hartree, i.e., a unit of energy used in molecular orbital calculations with $1Ha = 627.5$ kcal/mol. The exact value of E_m is acquired from a full configuration-interaction calculation [49].

The hyper-parameters of CEQAS to compute the lowest energy eigenvalues of H_h are as follows. The number of SuperCircuit has one setting, i.e., $W = 5$. The layer number for all ansatzes is $L = 3$. The number of iterations and the number of search ansatzes for ranking is $T = 100$ and $K = 500$, respectively. The search space of CEQAS for the single-qubit gates is fixed to be the rotational quantum gates along Y and Z axis. For the two-qubit gates, denoted by the index of four qubits as $(0, 1, 2, 3)$, QAS explores whether applying CNOT gates to the qubits pair $(0, 1)$, $(1, 2)$, $(2, 3)$ or not.

Figure 17 is the output ansatz of QAS under the noiseless setting. Figure 18 is the output ansatz of QAS under the noisy setting. In Fig. 17 and Fig. 18, the x-axis means that the estimated energy of the sampled ansatz is in the range of $(a, b]$, e.g., $a = -0.6Ha$, and $b = -0.4Ha$, the y-axis means that the number of sampled ansatz. "Continuous-NSGA-2" stands for the use of the CEQAS method. "Randon" represents the method of training SuperCircuit using QAS, and the search phase uses random search. "evolution" indicates that the SuperCircuit method is trained using QAS, and NSGA-II is employed in the search phase.

The lowest energy of H_h equals to $E_m = -1.136Ha$, $(-1.2Ha, -1Ha]$ is the interval with the smallest error in estimating energy. In Fig. 17, under the noiseless setting, CEQAS

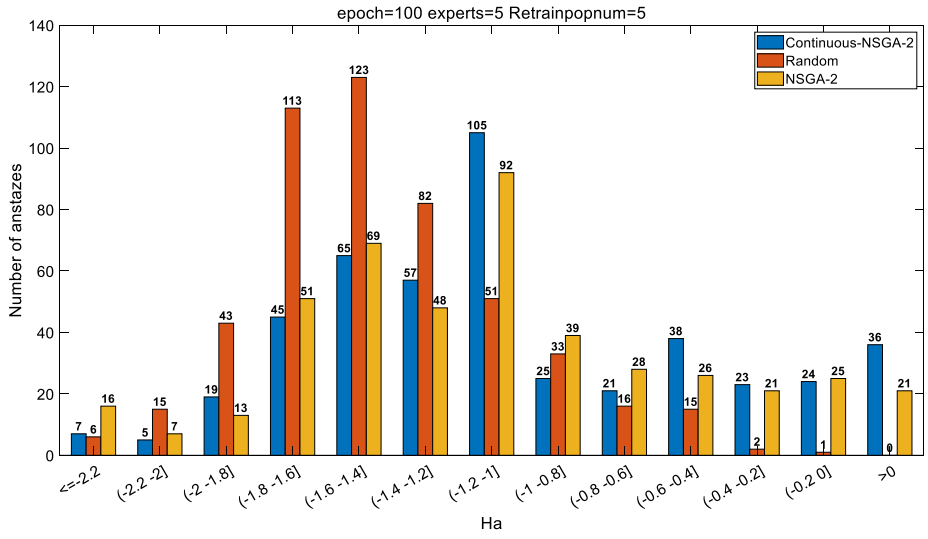

Figure 17 Simulation results for the ground state energy estimation of Hydrogen under the noiseless setting

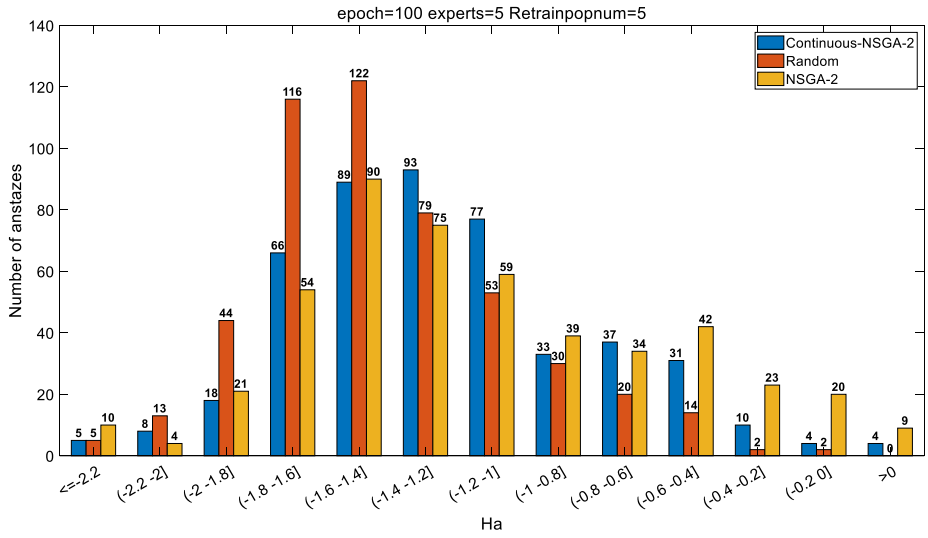

Figure 18 Simulation results for the ground state energy estimation of Hydrogen under the noisy setting

gives the number of ansatzes in the energy interval $(-1.2Ha, -1Ha]$ higher than QAS (105 > 92). In Fig. 18 under the noisy setting, the number of ansatzes in the energy interval $(-1.2Ha, -1Ha]$ obtained by CEQAS is lower than that under the noiseless setting (77 < 105), but it is still better than that of QAS (77 > 59). This shows that CEQAS can not only achieve better optimization of parameter space in binary classification task, but also has certain effectiveness in ground state energy estimation.

Appendix C: Extended version of Fig. 11–14

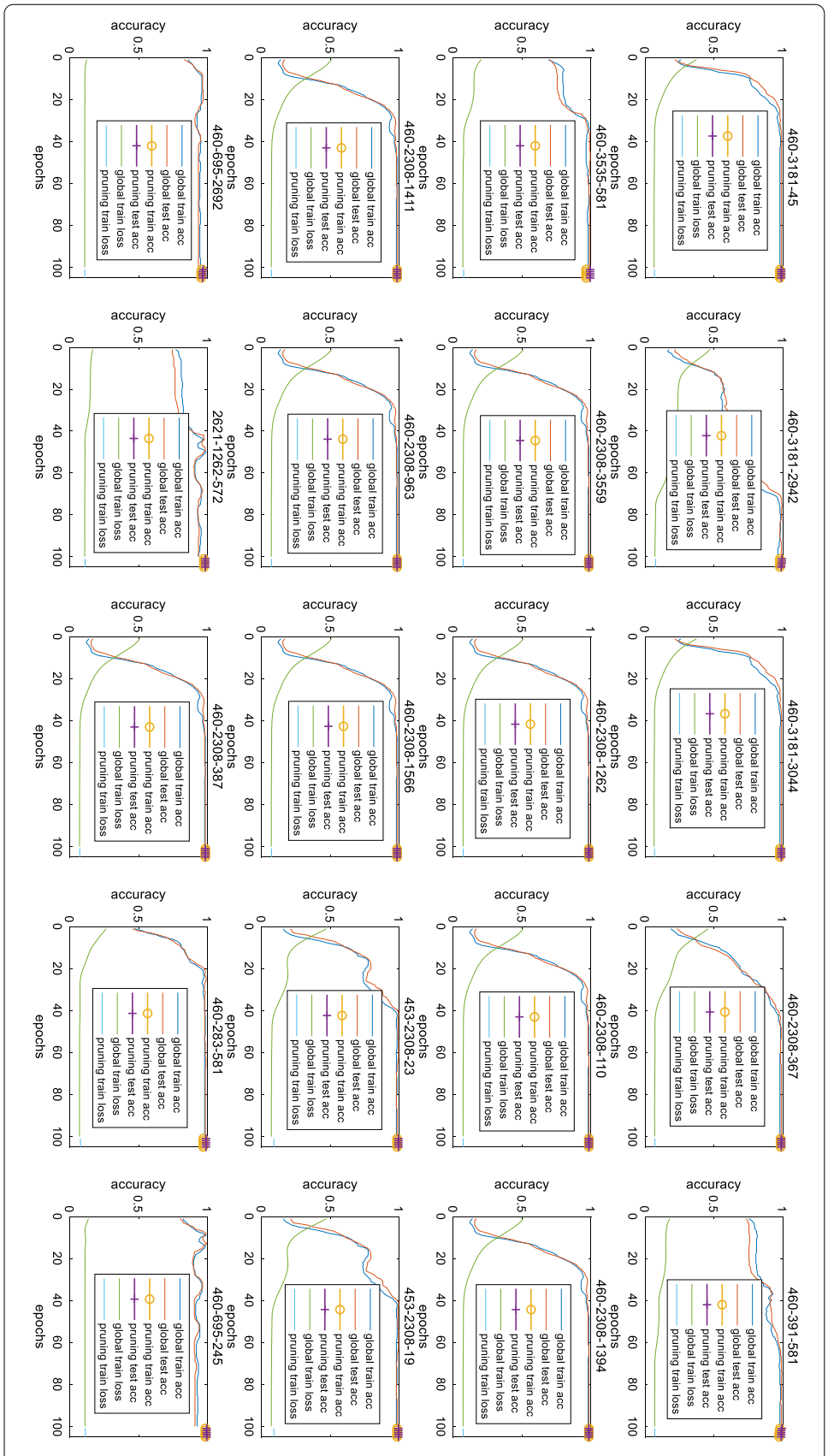

Figure 19 An extended version of Fig. 11

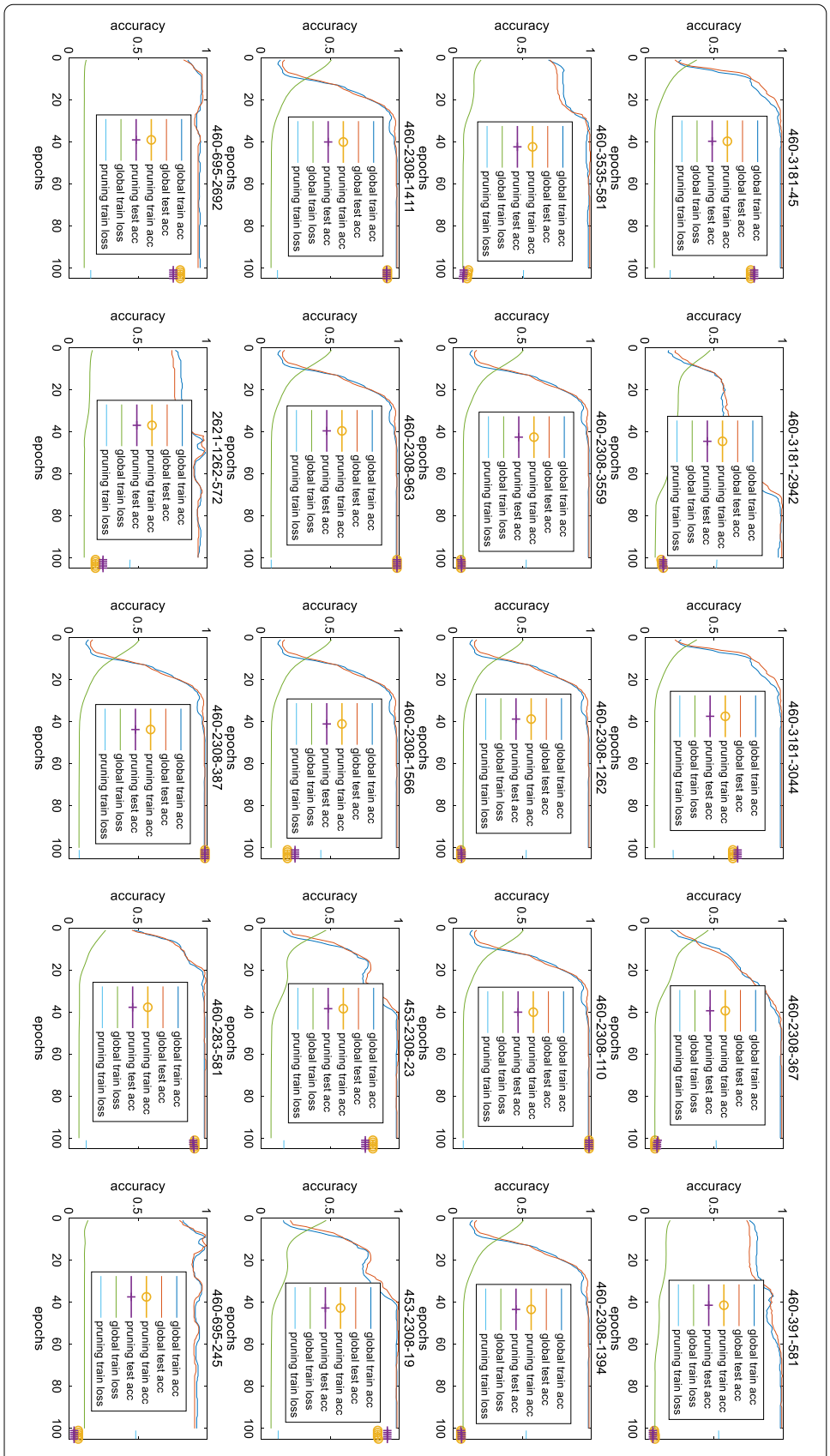

Figure 20 An extended version of Fig. 12

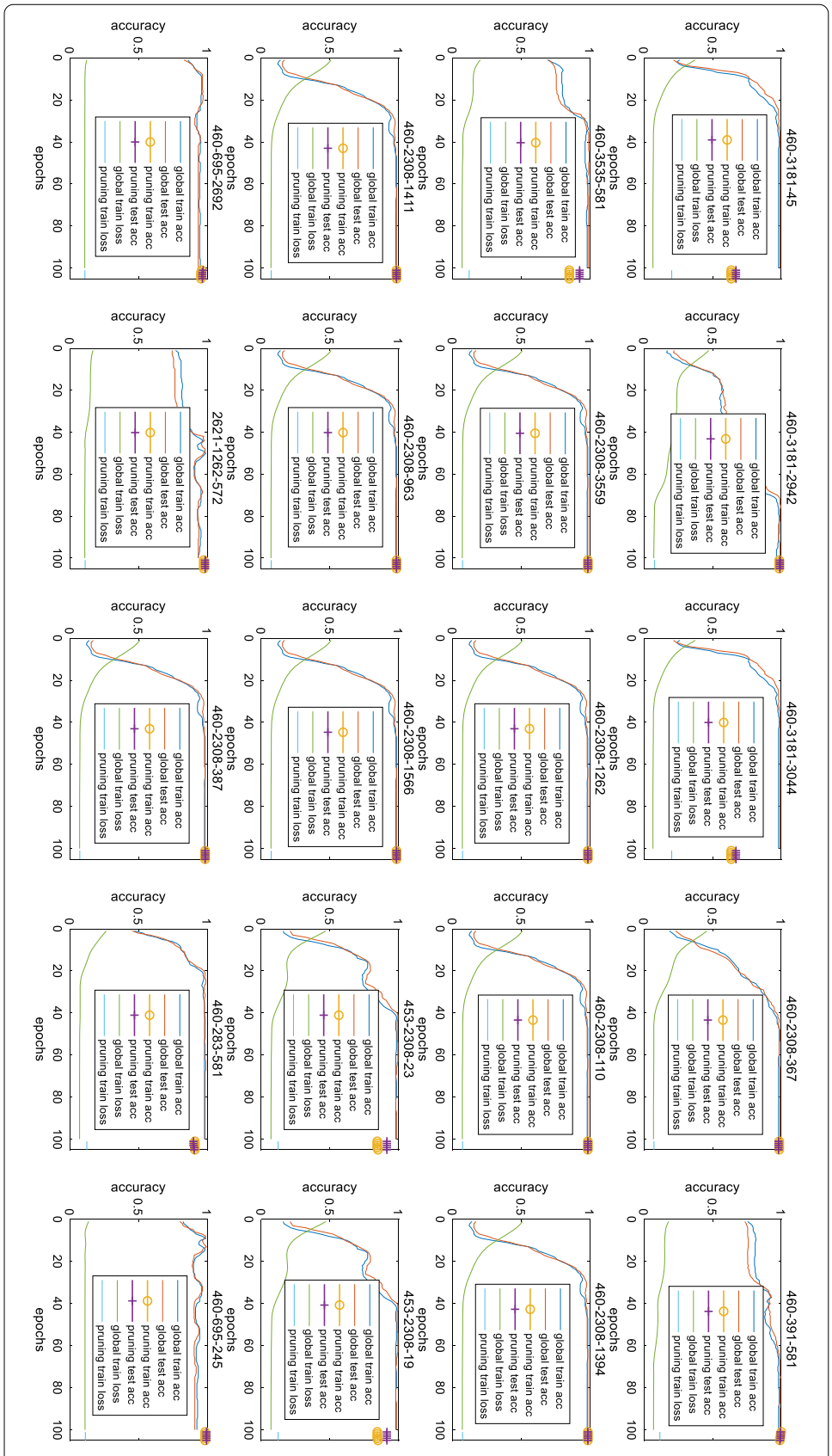

Figure 21 An extended version of Fig. 13

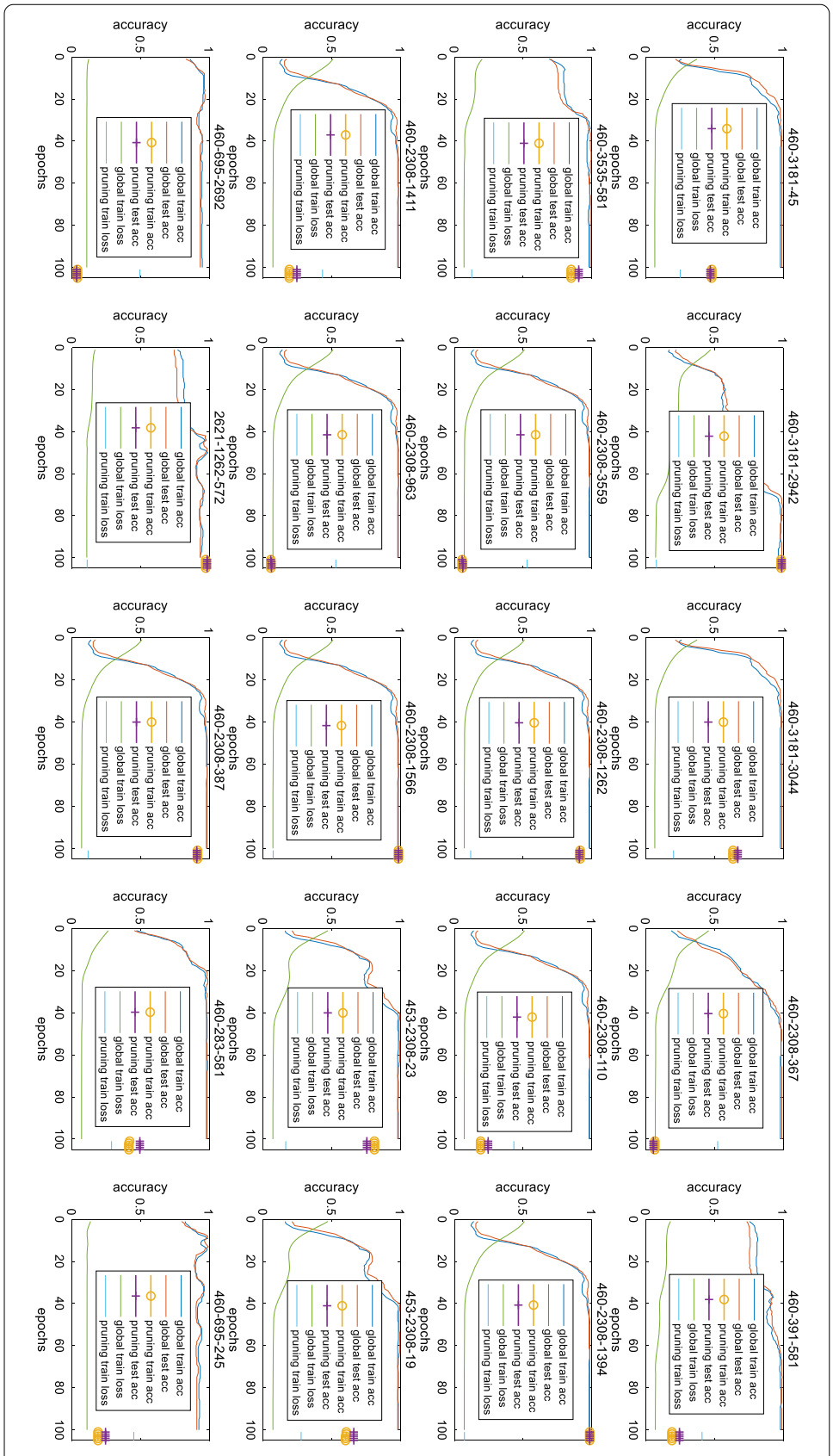

Figure 22 An extended version of Fig. 14

Abbreviations
VQAs, Variational Quantum Algorithms; PQCs, Parameterized Quantum Circuits; NSGA-II, Non-dominated Sorting Genetic Algorithm-II; NISQ, Noisy Intermediate-Scale Quantum; DARTS, Differentiable Architecture Search; CEQAS, Continuous Evolution for Efficient Quantum Architecture Search; QML, Quantum Machine Learning; QAS, Quantum Architecture Search; QAS-TR-PPO-RB, Trust Region-based PPO with Rollback for QAS; GNN, Graph Neural Network; GSQAS, Graph Self-supervised Quantum Architecture Search; PQAS, Predictor-based Quantum Architecture Search; PQAS-AL, Predictor-based Quantum Architecture Search with active learning; QNN, Quantum Neural Networks; SP, Symmetric Pruning; SSP, Structure Symmetric Pruning.

Author contributions
QGM and CLH designed all the experiments. XKY, LLQ and HZ conceived the idea and the scheme. QGM and MCX derived the theoretical framework and code. DQ, NWS and CLH contributed to reviewing, editing and assessing the results. The manuscript was written with contributions from all authors. All authors read and approved the final manuscript.

Funding
This work is supported by the National Natural Science Foundation of China under Grant No. 62171470, Henan Province Central Plains Science and Technology Innovation Leading Talent Project (No. 234200510019), Natural Science Foundation of Henan Province (No. 232300421240).

Data Availability
No datasets were generated or analysed during the current study.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
The Author confirms: • that the work described has not been published before; • that it is not under consideration for publication elsewhere.

Competing interests
The authors declare no competing interests.

Author details
[1]School of Information Systems Engineering, Information Engineering University, 62 Science Avenue, Zhengzhou, 450001, China. [2]Laboratory for Advanced Computing and Intelligence Engineering, Zhengzhou, 450001, China. [3]School of Electrical and Information Engineering, Zhengzhou University, 100 Science Avenue, Zhengzhou, 450001, China.

Received: 20 November 2023 Accepted: 26 August 2024 Published online: 06 September 2024

References

1. Du Y, Huang T, You S, Hsieh M-H, Tao D. Quantum circuit architecture search for variational quantum algorithms. npj Quantum Inf. 2022;8:62.
2. Schuld M, Killoran N. Quantum machine learning in feature Hilbert spaces. Phys Rev Lett. 2019;122:040504.
3. Cerezo M, Arrasmith A, Babbush R, Benjamin SC, Endo S, Fujii K, et al. Variational quantum algorithms. Nat Rev Phys. 2021;3:625–44.
4. Farhi E, Neven H. Classification with quantum neural networks on near term processors. arXiv. 2018. Available from http://arxiv.org/abs/1802.06002.
5. Beer K, Bondarenko D, Farrelly T, Osborne TJ, Salzmann R, Scheiermann D, et al. Training deep quantum neural networks. Nat Commun. 2020;11:808.
6. Wang D, Higgott O, Brierley, S. Accelerated variational quantum eigensolver. Phys Rev Lett. 2019;122:140504.
7. Mitarai K, Yan T, Fujii K. Generalization of the output of a variational quantum eigensolver by parameter interpolation with a low-depth ansatz. Phys Rev Appl. 2019;11:044087.
8. Peruzzo A, McClean J, Shadbolt P, Yung M-H, Zhou X-Q, Love PJ, et al. A variational eigenvalue solver on a photonic quantum processor. Nat Commun. 2014;5:4213.
9. Preskill J. Quantum computing in the NISQ era and beyond. Quantum. 2018;2:79.
10. Sim S, Johnson PD, Aspuru-Guzik A. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. Adv Quantum Technol. 2019;2:1900070.
11. Zoph B, Le, QV. Neural Architecture Search with Reinforcement Learning. arXiv. 2017. Available from http://arxiv.org/abs/1611.01578.
12. Zhu W, Pi J, Peng Q. A brief survey of quantum architecture search [Internet]. Proceedings of the 6th international conference on algorithms, computing and systems. Larissa: Assoc. Comput. Mach.; 2023. https://doi.org/10.1145/3564982.3564989.
13. Liu H, Simonyan K, Yang YY. DARTS: differentiable architecture search. arXiv. 2018. Available from https://api.semanticscholar.org/CorpusID:49411844.
14. Zhang S-X, Hsieh C-Y, Zhang S, Yao H. Differentiable quantum architecture search. Quantum Sci Technol. 2022;7:045023.
15. Wu W, Yan G, Lu X, Pan K, Yan J. QuantumDARTS: Differentiable Quantum Architecture Search for Variational Quantum Algorithms.

16. He Z, Chen C, Li L, Zheng S, Situ H. Quantum architecture search with meta-learning. Adv Quantum Technol. 2022;5:2100134.
17. Ostaszewski M, Trenkwalder LM, Masarczyk W, Scerri E, Dunjko V. Reinforcement learning for optimization of variational quantum circuit architectures. Neural Information Processing Systems. 2021. Available from https://api.semanticscholar.org/CorpusID:229166765.
18. Ye E, Chen SY-C. Quantum architecture search via continual reinforcement learning. arXiv. 2021. Available from http://arxiv.org/abs/2112.05779.
19. Kuo E-J, Fang Y-LL, Chen SY-C. Quantum architecture search via deep reinforcement learning. arXiv. 2021. Available from http://arxiv.org/abs/2104.07715.
20. Wang H, Ding Y, Gu J, Li Z, Lin Y, Pan DZ, et al. QuantumNAS: noise-Adaptive Search for Robust Quantum Circuits. arXiv. 2022. Available from http://arxiv.org/abs/2107.10845.
21. Pham H, Guan MY, Zoph B, Le QV, Dean J. Efficient Neural Architecture Search via Parameter Sharing. arXiv. 2018. Available from http://arxiv.org/abs/1802.03268.
22. Cai H, Gan C, Wang T, Zhang Z, Han, S. Once-for-All: train One Network and Specialize it for Efficient Deployment. arXiv. 2020. Available from http://arxiv.org/abs/1908.09791.
23. Rattew AG, Hu S, Pistoia M, Chen R, Wood S. A Domain-agnostic, Noise-resistant, Hardware-efficient Evolutionary Variational Quantum Eigensolver. arXiv. 2020. Available from. http://arxiv.org/abs/1910.09694.
24. Ding L, Evolutionary SL. Quantum architecture search for parametrized quantum circuits. Entropy. 2023;25:93.
25. Chivilikhin D, Samarin A, Ulyantsev V, Iorsh I, Oganov AR, Kyriienko O. MoG-VQE: multiobjective genetic variational quantum eigensolver. arXiv. 2020. Available from http://arxiv.org/abs/2007.04424.
26. Wang X, Liu J, Liu T, Luo Y, Du Y, Symmetric TD. Pruning in Quantum Neural Networks. arXiv. 2023. Available from http://arxiv.org/abs/2208.14057.
27. Schuld M, Sweke R, Meyer JJ. Effect of data encoding on the expressive power of variational quantum-machine-learning models. Phys Rev A. 2021;103:032430.
28. Li Y, Liu R, Hao X, Shang R, Zhao P, Jiao L. EQNAS: evolutionary quantum neural architecture search for image classification. Neural Netw. 2023;168:471–83.
29. Sun Y, Ma Y, Differentiable TV. Quantum Architecture Search for Quantum Reinforcement Learning. arXiv. 2023. Available from http://arxiv.org/abs/2309.10392.
30. Deng M, He Z, Zheng S, Zhou Y, Zhang F, Situ H. A progressive predictor-based quantum architecture search with active learning. Eur Phys J Plus. 2023;138:905.
31. Lu X, Pan K, Yan G, Shan J, Wu W, Yan J. QAS-Bench: benchmark. In: Krause A, Brunskill E, Cho K, Engelhardt B, Sabato S, Scarlett J, editors. Rethinking quantum architecture search and A. International Conference on Machine Learning, ICML 2023. 23–29 July 2023. Honolulu, Hawaii, USA. PMLR. 2023. p. 22880–98. Available from https://proceedings.mlr.press/v202/lu23f.html.
32. Zhu X, Hou X. Quantum architecture search via truly proximal policy optimization. Sci Rep. 2023;13:5157.
33. He Z, Zhang X, Chen C, Huang Z, Zhou Y, Situ H. A GNN-based predictor for quantum architecture search. Quantum Inf Process. 2023;22:128.
34. He Z, Deng M, Zheng S, Li L, Situ H. GSQAS: graph self-supervised quantum architecture search. Phys A, Stat Mech Appl. 2023;630:129286.
35. Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput. 2002;6:182–97.
36. Meyer JJ, Mularski M, Gil-Fuster E, Mele AA, Arzani F, Wilms A, et al. Exploiting symmetry in variational quantum machine learning. PRX Quantum. 2023;4:010328.
37. Sauvage F, Larocca M, Coles PJ, Cerezo M. Building spatial symmetries into parameterized quantum circuits for faster training. arXiv. 2023. Available from http://arxiv.org/abs/2207.14413.
38. Herasymenko Y, O'Brien TE. A diagrammatic approach to variational quantum ansatz construction. arXiv. 2021. Available from http://arxiv.org/abs/1907.08157.
39. Gard BT. Efficient symmetry-preserving state preparation circuits for the variational quantum eigensolver algorithm. npj Quantum Inf. 2020.
40. Zheng H, Li Z, Liu J, Strelchuk S, Kondor R. Speeding up learning quantum states through group equivariant convolutional quantum ansätze. PRX Quantum. 2023;4:020327.
41. Zheng H, Li Z, Liu J, Strelchuk S, Kondor R. On the Super-exponential Quantum Speedup of Equivariant Quantum Machine Learning Algorithms with SU(*d*) Symmetry. arXiv. 2022. Available from http://arxiv.org/abs/2207.07250.
42. Shaydulin R, Wild SM. Exploiting symmetry reduces the cost of training QAOA. IEEE Trans Quantum Eng. 2021;2:1–9.
43. Mernyei P, Meichanetzidis K, Ceylan İİ. Equivariant Quantum Graph Circuits. arXiv. 2022. Available from http://arxiv.org/abs/2112.05261.
44. Marvian I. Restrictions on realizable unitary operations imposed by symmetry and locality. Nat Phys. 2022;18:283–9.
45. Bergholm V, Izaac J, Schuld M, Gogolin C, Ahmed S, Ajith V, et al. PennyLane: automatic differentiation of hybrid quantum-classical computations. arXiv. 2022. Available from http://arxiv.org/abs/1811.04968.
46. IBM Quantum Computing Qiskit. Available from https://www.ibm.com/quantum/www.ibm.com/quantum/qiskit.
47. O'Malley PJJ, Babbush R, Kivlichan ID, Romero J, McClean JR, Barends R, et al. Scalable quantum simulation of molecular energies. Phys Rev X. 2016;6:031007.
48. Kandala A, Mezzacapo A, Temme K, Takita M, Brink M, Chow JM, et al. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. Nature. 2017;549:242–6.
49. McArdle S, Endo S, Aspuru-Guzik A, Benjamin SC, Yuan X. Quantum computational chemistry. Rev Mod Phys. 2020;92:015003.

Publisher's Note