QuantumNAS: Noise-Adaptive Search for Robust Quantum Circuits

Hanrui Wang[1], Yongshan Ding[2], Jiaqi Gu[3], Yujun Lin[1], David Z. Pan[3], Frederic T. Chong[4], Song Han[1]
[1]Massachusetts Institute of Technology
[2]Yale University
[3]The University of Texas at Austin
[4]University of Chicago
{hanrui,yujunlin,songhan}@mit.edu, yongshan.ding@yale.edu, jqgu@utexas.edu, dpan@ece.utexas.edu, chong@cs.uchicago.edu

***Abstract*— Quantum noise is the key challenge in Noisy Intermediate-Scale Quantum (NISQ) computers. Previous work for mitigating noise has primarily focused on gate-level or pulse-level noise-adaptive compilation. However, limited research efforts have explored a *higher level of optimization* by making the quantum circuits themselves resilient to noise.**

In this paper, we propose QuantumNAS, a comprehensive framework for noise-adaptive co-search of the variational circuit and qubit mapping. Variational quantum circuits are a promising approach for constructing quantum neural networks for machine learning and variational ansatzes for quantum simulation. However, finding the best variational circuit and its optimal parameters is challenging due to the large design space and parameter training cost. We propose to *decouple* the circuit search and parameter training by introducing a novel *SuperCircuit*. The SuperCircuit is constructed with multiple layers of pre-defined parameterized gates (e.g., `U3` and `CU3`) and trained by iteratively sampling and updating the parameter subsets (SubCircuits) of it. It provides an accurate estimation of SubCircuits performance trained from scratch. Then we perform an evolutionary co-search of SubCircuit and its qubit mapping. The SubCircuit performance is estimated with parameters inherited from SuperCircuit and simulated with real device noise models. Finally, we perform iterative gate pruning and finetuning to remove redundant gates in a fine-grained manner.

Extensively evaluated with 12 quantum machine learning (QML) and variational quantum eigensolver (VQE) benchmarks on 10 quantum computers, QuantumNAS significantly outperforms noise-unaware search, human, random, and existing noise-adaptive qubit mapping baselines. For QML tasks, QuantumNAS is the first to demonstrate over 95% 2-class, 85% 4-class, and 32% 10-class classification accuracy on real quantum computers. It also achieves the lowest eigenvalue for VQE tasks on H_2, H_2O, LiH, CH_4, BeH_2 compared with UCCSD baselines. We also open-source QuantumEngine for fast training of parameterized quantum circuits to facilitate future research.

I. Introduction

Quantum Computing (QC) is a new computational paradigm that aims to address classically intractable problems with considerably higher efficiency and speed. It has been shown to have exponential or polynomial advantage in various domains such as cryptography [56], database search [21], chemistry [13], [30], [51] and machine learning [7], [18], [24], [37], [54], etc. In the recent two decades, QC hardware has witnessed rapid progress by virtue of breakthroughs in physical implementation technologies.

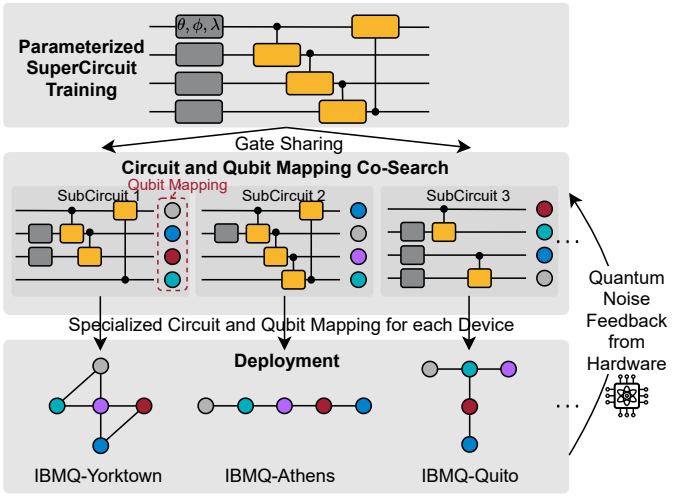

Fig. 1. Noise-adaptive circuit and qubit mapping co-search. A gate-sharing SuperCircuit that contains numerous parameter subsets (SubCircuits) is firstly trained. Then we perform an evolutionary search with the quantum noise feedback to find the most robust circuit and qubit mapping.

Despite the exciting advancements, we are still expected to reside in the Noisy Intermediate Scale Quantum (NISQ) stage for multiple years before entering the Fault-Tolerant era [5], [20]. In the NISQ era, quantum computers typically contain tens to hundreds of qubits, which are insufficient for quantum error correction. The qubits and quantum gates also suffer from high error rates of 10^{-3} to 10^{-2}. Therefore, reducing quantum error is of pressing demand to close the gap between the requirements from the quantum algorithm side and available QC capacity from the hardware side.

A series of works focusing on noise-adaptive quantum program compilation has been proposed to mitigate the noise impact. Noise-adaptive qubit mapping [34], [47], [57] aims to find the best mapping from logical qubits to physical qubits, which minimizes the gate error and SWAP insertion overhead. Noise-adaptive instruction scheduling and crosstalk mitigation techniques [16], [48] aim to reduce the undesired inter-qubit interference and the circuit depth. However, those techniques only explore a small design space by optimizing the compilation process with a *fixed* input quantum circuit. Limited research efforts have been made to explore how to

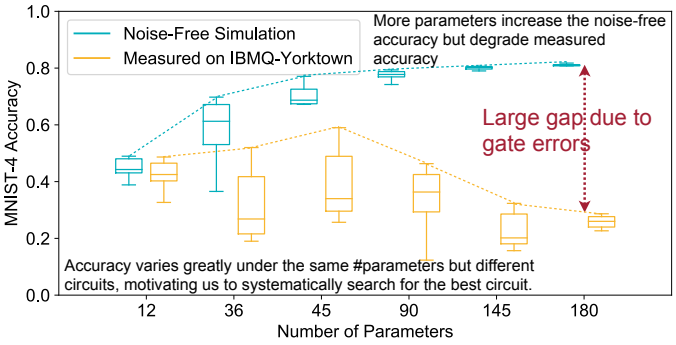

Fig. 2. MNIST-4 on noise-free simulator / real QC. More parameters increase the noise-free accuracy but degrade measured accuracy due to larger gate errors. Accuracy varies greatly under the same #parameters but different circuits, motivating us to search for the best circuit systematically.

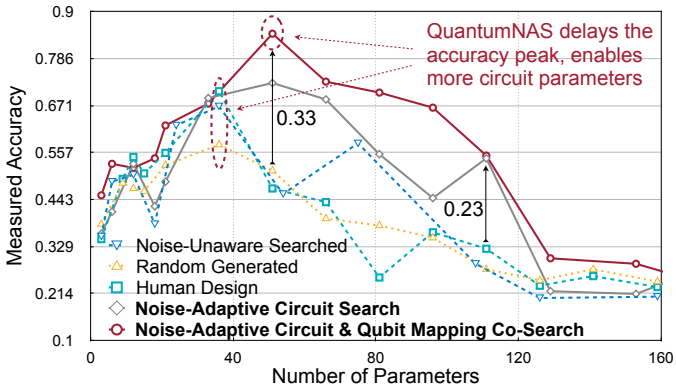

Fig. 3. Accuracy vs. #parameters of multiple methods. The accuracy of conventional designs quickly saturates then drops. QuantumNAS mitigates the quantum noise and delays the peak of the curve, allowing larger model capacity and higher accuracy (up to 33% higher).

improve the noise resilience of QC via a co-design strategy for searching, training, and compiling quantum circuits.

This work fills this blank by proposing QuantumNAS, a noise-adaptive quantum circuit and qubit mapping co-search framework to find the most robust quantum circuit and corresponding qubit mapping tailored for a given task on the target quantum device as in Figure 1. We study parameterized quantum circuits since they provide unique opportunities to alter circuit structures while performing the same functionality.

First, we are strongly motivated by the significant impacts of quantum noise on performance. In Figure 2, we show the accuracy of MNIST 4-class image classification simulated by the noise-free simulator and measured on the real IBMQ-Yorktown quantum computer. *Key observations*: (1) More parameters increase the model capacity, thus increasing noise-free simulation accuracy. Nevertheless, more parameters mean more gates, which introduces more noise, and the accumulated noise quickly offsets the capacity benefit. As a result, the measured accuracy peaks at 45 parameters. (2) To make things worse, quantum noise exacerbates the performance variance. The measured accuracy variance under the same #parameters is much higher than that of noise-free, e.g., [25%, 59%] vs. [67%, 77%] under 45 parameters. The observations both call for the noise-adaptive search for the most robust circuit.

One major challenge for this noise-adaptive search is the algorithmic scalability issue. It is almost intractable to solve the two-level optimization problem (for quantum circuit and qubit mapping) via iterative circuit sampling, parameter training, and evaluation in the large design space. To address this, we propose to *decouple the training and search* by introducing a novel *SuperCircuit*-based search approach (Figure 1). We first construct a SuperCircuit by stacking a sufficient number of layers of pre-defined parameterized gates to cover a large design space. Then, we 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. In this way, we only pay the training cost *once* but can evaluate *all* the SubCircuits fast and efficiently. Hence, the search cost is significantly reduced.

Furthermore, we perform an evolutionary co-search with noise information in the loop to find the most robust quantum circuit and qubit mapping jointly. In each iteration, the evolution engine samples a population of SubCircuit and qubit mapping pairs. Then the performance of each sampled SubCircuit can be evaluated by an estimator on two types of backends: a *noise-aware simulator* or a *real quantum hardware*. The estimator takes the inherited parameters from the SuperCircuit and assigns them to the SubCircuit. With a noise-aware simulator backend, the performance is evaluated with direct noise classical simulation with a realistic device noise model. Alternatively, we can replace the simulator with real quantum hardware. The requirement for such evaluation is *no harder than* any common variational quantum algorithms. After multiple evolutionary search iterations, we can obtain a pair of robust circuit and qubit mapping and then train the parameters from scratch. SuperCircuit-based search is inspired by the supernet method in classical ML model training [12], [22], [52]. However, we have four major differences: (1) The SuperCircuit is more general than the ML model and can be applied to various parameterized quantum algorithms such as VQE; (2) We co-search circuit with its qubit mapping; (3) Our search is aware of quantum noise to improve robustness; (4) We propose novel *front sampling* and *restricted sampling* specialized for quantum circuits.

Finally, on top of the searched circuit and qubit mapping, we further propose a fine-grained *pruning* to remove redundant parameters and gates and *finetuning* to recover the performance. We end up with a slimmed circuit with similar noise-free performance but fewer noise sources, which in return improves the final measured performance.

Overall, QuantumNAS can mitigate the impact of quantum noise and delays the accuracy peak as shown in Figure 3. The contributions of QuantumNAS are five-fold: ❶ **Noise-Adaptive Quantum Circuit & Qubit Mapping Co-Search** to enable noise-resilient QC. ❷ **SuperCircuit-based Efficient Search Flow**: we propose a scalable quantum circuit search method based on SuperCircuit. Front sampling and restricted sampling are proposed for efficient exploration and stable

optimization in the huge design space. ❸ **Iterative Quantum Pruning** is introduced to remove redundant quantum gates in a fine-grained manner. ❹ **Extensive Real QC Evaluations**: we extensively evaluate QuantumNAS with 12 benchmarks in QML and VQE on 10 quantum computers, observing significant improvements over baselines. ❺ **Open-Source QC Library**: To facilitate future research in QML and variational quantum simulation, we release QuantumEngine, a PyTorch-based GPU-accelerated library to enable fast training of parameterized quantum circuits (over 200× faster than the PennyLane [6]). It also supports push-the-button deployment of trained circuits on real quantum devices.

II. Background and Motivation

A. Quantum Basics

Qubits – Unlike a conventional bit, a quantum bit (*qubit* [15], [49]) can be in a linear combination of the two basis states 0 and 1: $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, for $\alpha, \beta \in \mathbb{C}$, satisfying $|\alpha|^2 + |\beta|^2 = 1$. The ability to create a "superposition" of basis states allows us to use an n-qubit system to represent a linear combination of 2^n basis states. In contrast, a classical n-bit register can only store one of the 2^n states.

Quantum Circuits – To perform computation on a quantum system, we manipulate the qubits' state by applying a *quantum circuit*. A quantum circuit consists of a sequence of operations called *quantum gates*, which take one quantum state to another through unitary transformations, i.e., $|\psi\rangle \rightarrow U|\psi\rangle$, where U is a unitary matrix. Results of a quantum circuit are obtained by qubit readout operations called *measurements*, which collapse a qubit state $|\psi\rangle$ to either $|0\rangle$ or $|1\rangle$ probabilistically according to the amplitudes α and β.

Operational Noises – In real QC, errors occur due to imperfect control signals, unwanted interactions between qubits, or interference from the environment [11], [32]. Thus, qubits undergo *decoherence error* (spontaneous loss of its stored information) over time, and quantum gates introduce *operation errors* (e.g., coherent/stochastic errors) into the system. These systems need to be characterized [42] and calibrated [29] frequently to mitigate noise impacts. So noise-adaptive techniques in QC algorithms, circuits, and devices are critical.

B. Variational Quantum Circuits

A variational circuit is a trainable quantum circuit where its quantum gates are parameterized (e.g., by angles in quantum rotation gates). The parameterized quantum circuit $\Phi(x, \theta)$ is used to prepare a variational quantum state: $|\psi(x, \theta)\rangle = \Phi(x, \theta)|0 \ldots 0\rangle$, where x is the input data related to the computation and θ is a set of free variables for adaptive optimizations. The variational method has shown huge potentials in applications such as quantum machine learning [4], [7], [55], [61], numerical analysis [36], [38], quantum simulation [30], [31], [51], and optimizations [18], [46].

Typically, the training of variational circuits is performed by first selecting a hand-designed circuit for a computational task and, secondly, finding an optimal set of parameters for the circuit via a hybrid quantum-classical optimization procedure.

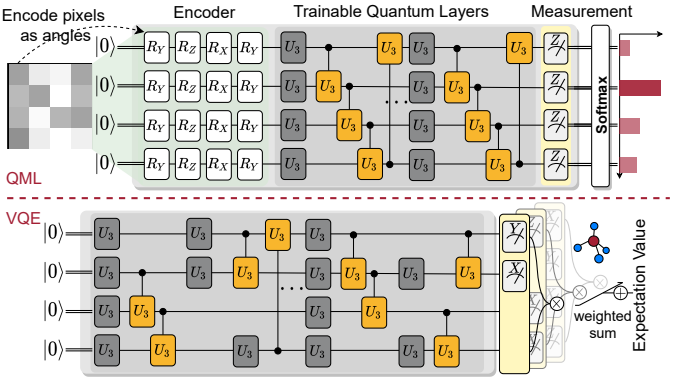

Fig. 4. Example circuits for QML and VQE tasks.

The optimization is usually an iterative process to search for the best candidates for the parameters in $\Phi(x, \theta)$. Whether a variational quantum algorithm is successful depends on how well the circuit can be trained. For example, "barren plateau" [43] is a phenomenon when the cost function landscape is flat, making a variational circuit untrainable using any gradient-based optimization algorithms.

Quantum Neural Network (QNN) is a promising application of variational quantum circuits. [1] shows that from an information geometry point of view, if carefully designed, QNNs have higher effective dimensions and faster training convergence speed over classical NNs. That highly motivates improvements of QNN robustness on real quantum machines.

Figure 4 shows the example circuits we used for QML (QNN) and VQE. For QML tasks such as image classification, we first encode the pixels using rotation gates and then use parameterized trainable quantum gates to process the information. We measure the qubits on Z-basis to obtain classical values, then compute Softmax of those values to get the probability for each class. For VQE, the parameterized circuit is used for state preparation, and the measurement part is constructed according to the molecule. We prepare for the state multiple times for measurements on different qubits and bases, multiply expectation values of qubits, and perform weighted sum. The final result is the expectation value for the ground state energy of the molecule. The parameters can be trained with backpropagation, in which we compute the derivative of each parameter (θ_i) on loss function (L) and update the parameters with a learning rate α, $\hat{\theta}_i = \theta_i - \alpha \frac{\partial L}{\partial \theta_i}$.

III. Noise-Adaptive QuantumNAS

A. Overview

Figure 5 shows QuantumNAS overview, time cost, and a simple example. Firstly, a SuperCircuit is trained as a fast estimation of SubCircuits performance ranking. *Front sampling* and *restricted sampling* are proposed to promote the reliability of estimations. Then a noise-adaptive evolutionary co-search is performed to find the best circuit and qubit mapping pair. A performance estimator is employed to provide fast and accurate feedback to the evolution engine. Redundant gates with small parameter magnitude are further pruned from the

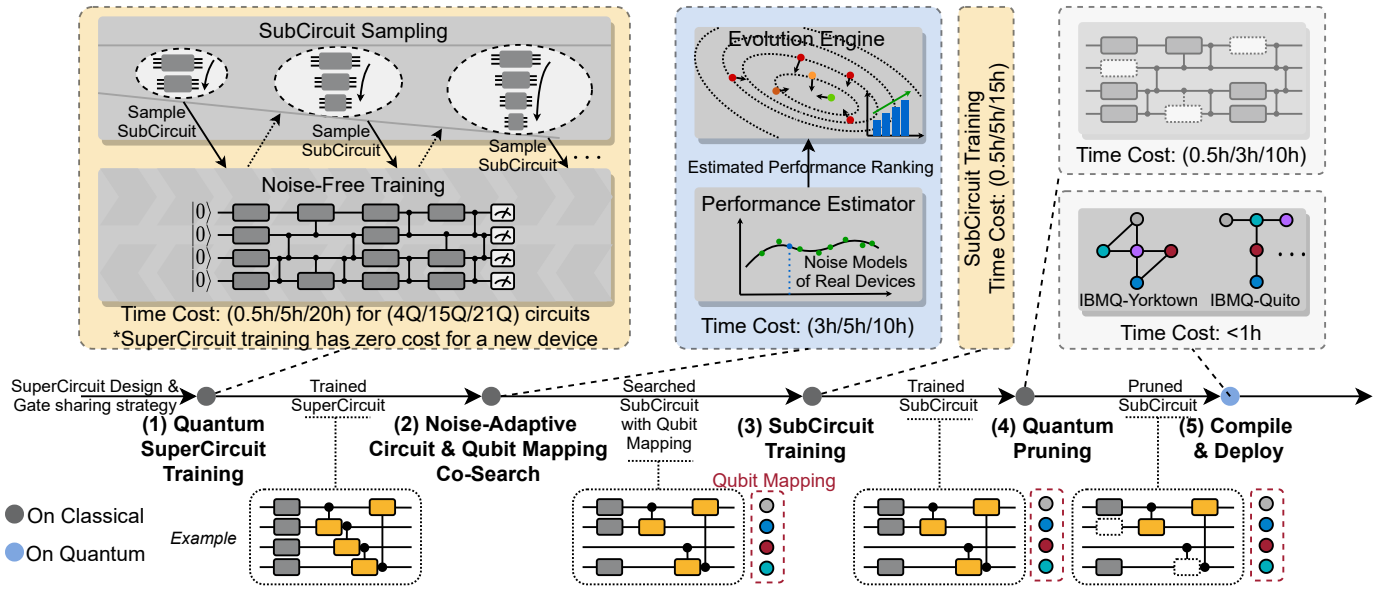

Fig. 5. QuantumNAS Overview. (1) A SuperCircuit is trained by iteratively sampling and updating parameter subsets (SubCircuits). The parameters from SuperCircuit and the simulator with practical noise models can provide an accurate final performance ranking estimation of SubCircuits. (2) Evolutionary co-search for circuit and qubit mapping pair of best estimated performance (lowest validation loss/eigenvalue for QML/VQE). (3) Train the searched SubCircuit. (4) Iterative pruning and finetuning to remove redundant gates. (5) Compile and deploy on real devices.

searched circuit. The pruned circuit is finally compiled and deployed on real quantum devices.

B. SuperCircuit Construction and Training

It is critical to encompass a large design space to include the most robust circuit. However, training all candidate circuits, evaluating their final performance, and selecting the best one is too costly. We thus propose SuperCircuit to evaluate each circuit in the design space (SubCircuit) *without* fully training it. Since we only need to find the best circuit, *relative performance* is sufficient, and can be estimated by the SuperCircuit.

With pre-specified basis gates and design space, the SuperCircuit is defined as the circuit with the largest number of gates in the space, whose parameters are trained by iteratively sampling and updating a subset of gates/parameters (SubCircuit). SuperCircuit contains multiple blocks, each with several layers of parameterized gates. A SubCircuit is a subset of the SuperCircuit that can have different number of blocks and gates inside blocks. Figure 6 shows one block of `U1+CU1` space containing one `U1` layer and one `CU1` layer. The SuperCircuit contains all gates, while the SubCircuit only contains gates with solid lines. In one SuperCircuit training step, we sample a SubCircuit and only compute gradients and update that subset of parameters of SuperCircuit. Intuitively, training a SuperCircuit is simultaneously training all SubCircuits in the design space.

SuperCircuit aims to facilitate the low-cost evaluation of SubCircuits in the design space. Given one SubCircuit, it is sufficient to inherit the gate parameters from the SuperCircuit and then perform evaluation *without* training. That provides an accurate estimation of the relative performance of the SubCircuit. Since the next stage is derivative-free optimization such as evolutionary search, using relative performance between SubCircuits is sufficient to find the best one. In addition, SuperCircuit can be reused for new devices or when noise changes. Thus, we only need to pay the noise-free SuperCircuit training cost for *once* but can use it for *all* devices. The number of circuits run for naïve search is $N_{device} \times N_{search} \times (N_{train} + N_{eval})$; while that for SuperCircuit search is $1 \times N_{train} + N_{device} \times N_{search} \times N_{eval}$. The overall search cost is significantly reduced by around $N_{device} \times N_{search}$ times which is $10 \times 1600 = 16,000$ in our setting. N_{device} is the number of quantum devices to execute the circuit. N_{search} is the number of evaluated circuits during search. N_{train}/N_{eval} is the number of circuit running iterations in training/evaluation.

A critical challenge in sampling-based SuperCircuit training is the large variance. Naïve random sampling often causes severe trainability issues due to intractable sampling variance from drastic SubCircuit change, leading to unreliable relative performance estimation. To address this, we propose *front sampling* and *restricted sampling*.

Front Sampling. In front sampling, only the subsets with the *several front blocks and front gates* can be sampled. For instance, if the subset contains three blocks, then blocks 0, 1, 2 will be sampled. Inside a block, if two gates are sampled in a layer, then the gates on qubits 0 and 1 will be sampled. Figure 6 shows several valid cases of front sampling. Front sampling helps improve SuperCircuit trainability as SubCircuits share the parameters of front blocks and gates.

Restricted Sampling is another essential technique we propose to boost training stability. We prevent the sampled SubCircuits from changing dramatically between two steps by constraining the maximum number of different layers. Therefore, the training process is stabilized as the sampling variance is under control. As in Figure 7, the upper path is

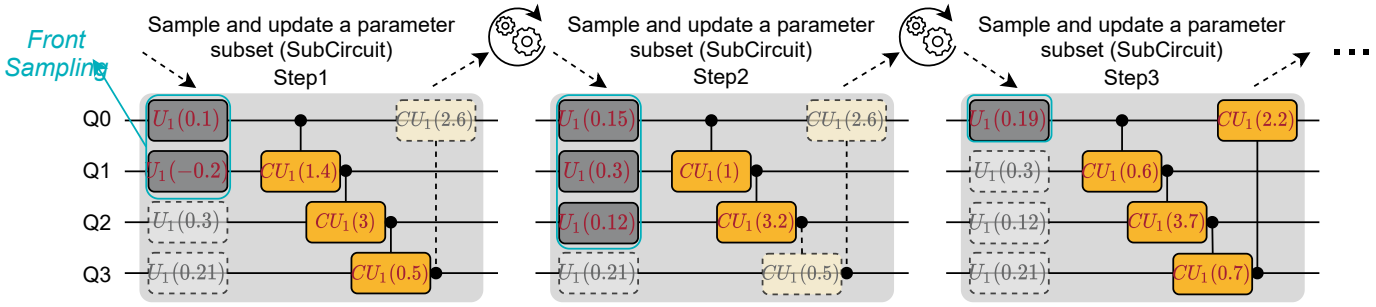

Fig. 6. SuperCircuit training. At each step, a subset of SuperCircuit parameters (SubCircuit) is sampled and then updated.

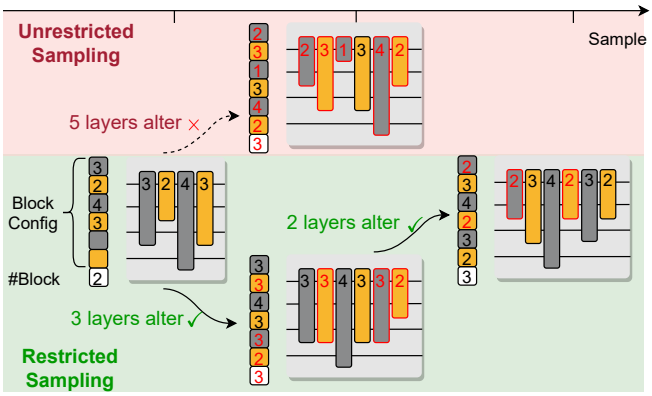

Fig. 7. Restricted sampling constrains #layers that are different between two steps. It improves SubCircuit consistency thus stabilizes SuperCircuit training.

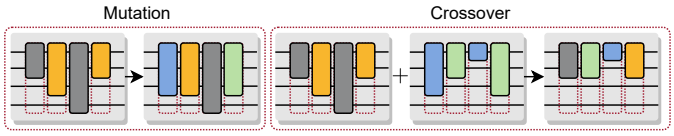

Fig. 8. SubCircuit mutation and crossover in evolutionary search.

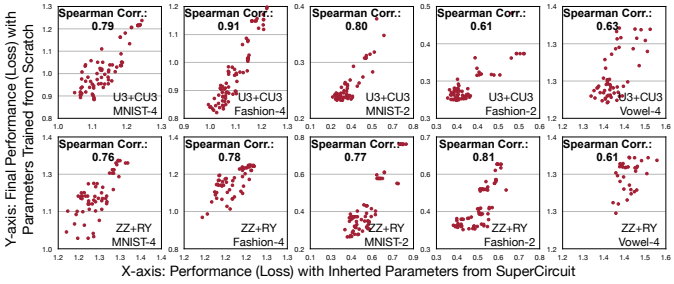

Fig. 9. Strong correlation between performance with inherited parameters from SuperCircuit and parameters trained from scratch.

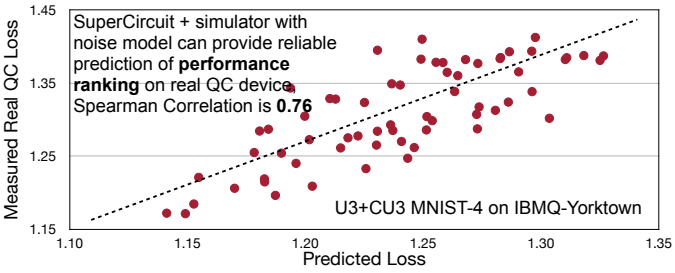

Fig. 10. Reliability of the performance estimator. The estimated loss can accurately indicate the final loss.

unrestricted sampling where the two SubCircuits differ by 5 layers. In the bottom path, restricted sampling limits the layer differences to 3, bringing much better cross-step consistency.

C. Noise-Adaptive Evolutionary Co-Search

SuperCircuit provides highly efficient relative performance estimations. We adopt a derivative-free optimization to explore the joint search space of circuit and qubit mapping.

Evolutionary Search. Genetic algorithm is employed in which the gene vector encodes circuit and qubit mapping. Each element in the circuit sub-gene represents the circuit width (#gates) in the layer. One additional gene sets the circuit depth (#blocks). Front sampling is also applied here. The qubit mapping sub-gene encodes the mapping between logical and physical qubits. We concatenate circuit and qubit mapping sub-genes as the pair's gene.

The evolution engine searches for high-performance pairs by keeping a population of pairs. In one iteration, it first evaluates all pairs by querying a performance estimator and selects multiple pairs with the highest performance (the lowest loss/eigenvalue for QML/VQE) as the parent population. Then mutation and crossover are conducted to generate the new population as in Figure 8. Mutation randomly alters several genes with a pre-defined probability. Crossover first selects two parent samples from the parent population; and then generates a new sample, each gene of which is randomly selected from one of the two parent samples. The new population is the ensemble of parent population, mutations, and crossovers. Then we sort the new population and select the ones with the highest performance as parents and enter the next iteration. The population of the very first iteration is from random sampling. Population size across iterations remains the same. For QML, we use validation set loss as the indicator. The lower the validation loss, the higher the final measured accuracy.

Performance Estimator. Ideally, the performance of circuit-qubit mapping pairs is directly evaluated on real quantum devices, which, however, could be extremely slow due to limited resources and queuing. Therefore we apply an estimator to provide *fast relative performance with noise*. It takes the query pairs from the evolution engine as inputs. It inherits the gate parameters of searched SubCircuit from SuperCircuit and sets the searched qubit mapping as the initial mapping of the compiler. There are two ways of estimation. One way is to use a simulator with a noise model from real devices. Noise models are from calibrations such as randomized benchmarking performed by the IBMQ team. They contain coherence (depolarizing), decoherence

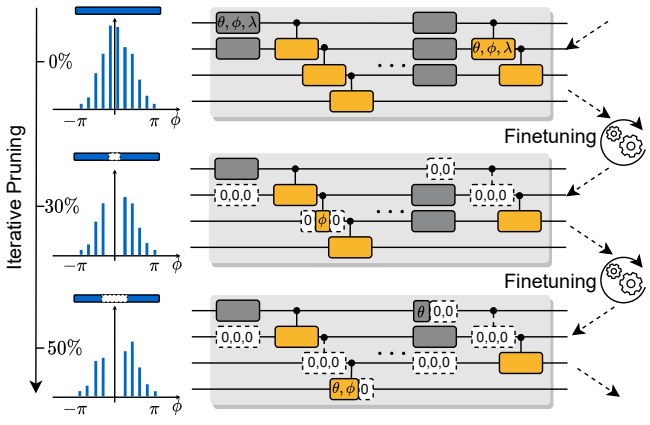

Fig. 11. Iterative pruning and finetuning remove small-magnitude parameters.

(thermal relaxation), and SPAM (readout) errors. The models are updated around twice a day and can be directly accessed with Qiskit API; the second is to use a noise-free simulator and compute the overall success rate with the product of success rates of all gates. Then the augmented loss will be noise-free simulated loss divided by calculated success rate: $r_{overall} = \prod_i r_{gate_i}, l_{augmented} = \frac{l_{noise\_free}}{r_{overall}}$, where r is success rate and l is loss. The first method is more accurate but slower, while the second is less accurate but faster. Therefore, in QuantumNAS, small circuits ($\leq$10 Qubits) apply the first method; large circuits apply the second.

The estimator has two approximations. The first uses the performance of one SubCircuit with *inherited* parameters to estimate the performance of the same SubCircuit with parameters *trained from scratch*. The second uses the simulation results, either with noise model or success rate, to estimate the performance on real devices. Since we only care about relative performance, the two-level approximation still maintains enough reliability for the search engine. Figure 9 shows the effectiveness of the first approximation with five tasks in two design spaces. For each point, the x-axis value is the performance (loss) with inherited parameters from SuperCircuit; the y-axis value is that with parameters trained from scratch. The average Spearman's correlation score is 0.75, showing *strong positive correlations* thus accurate relative performance. Figure 10 further shows the final estimated loss and the real loss for MNIST-4 on IBMQ-Yorktown. The correlation between them is 0.76, which indicates a significant positive relation. Therefore, the estimated performance is reliable enough to search for the best circuit-mapping pair.

D. Iterative Quantum Pruning

We further propose to remove redundant quantum gates to reduce the noise. The motivations are three-fold. First, the sub-optimality of the evolutionary search stage leaves room for further optimization of the searched circuit by reducing the number of gates. Second, even with the same circuit, there exist multiple parameter sets to achieve similar noise-free performance. Some sets contain more parameters with a magnitude close to zero, which can be safely removed

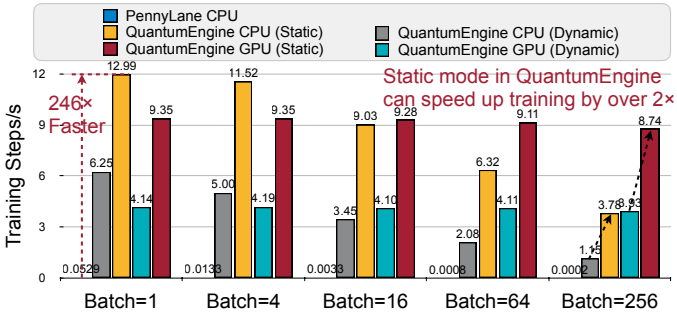

Fig. 12. Training speed of QuantumEngine vs. PennyLane [6]. QuantumEngine provides at least 246× speedup.

with iterative pruning and finetuning. Third, some gates, such as `U3`, contain multiple parameters. Partially removing the parameters can also bring benefits. We find that the number of compiled gates of U3(θ, ϕ, λ), U3$(\mathbf{0}, \phi, \lambda)$, U3$(\theta, \phi, \mathbf{0})$, U3$(\theta, \mathbf{0}, \mathbf{0})$, U3$(\mathbf{0}, \phi, \mathbf{0})$ and U3$(\mathbf{0}, \mathbf{0}, \lambda)$ are 5, 1, 4, 4, 1, 1, respectively. Therefore, having one or two parameters as zeros in the `U3` gates can reduce up to 80% gates compiled to the basis gate set (`CNOT`, `SX`, `RZ`).

Therefore, we propose iterative pruning as in Figure 11 to remove the circuit parameters in a fine-grained manner. Specifically, we first train the searched circuit from scratch to convergence. We rank all the normalized rotation angles $(\theta, \phi, \lambda) \in [-\pi, \pi)$ and remove part of angles that are closest to 0°. Then we finetune the rest parameters to recover the accuracy. We iteratively increase the pruning ratio and finetune the circuit parameters until achieving the desired ratio. In practice, we adopt polynomial pruning ratio decay [67]: $r_{now} = r_{final} + (r_{initial} - r_{final})\left(1 - \frac{s_{now} - s_{begin}}{s_{end} - s_{begin}}\right)^3$ where r is pruning ratio and s is training step. For final pruning ratio selection, we make sure that the noise-free simulation performance is not degraded compared with the un-pruned circuit. Thus, due to fewer gates after compilation, the accuracy of the circuit can be further increased by up to 9%.

E. QuantumEngine

To accelerate parameterized quantum circuit training in this work, we build a PyTorch library named QuantumEngine. Its APIs are implemented similarly to existing operations in PyTorch. So it makes quantum circuit construction as easy as a standard neural network model. It supports all common quantum gates. The state vector and unitary matrix of each gate are implemented with a native `torch.Tensor` data type. The simulations are achieved with complex-valued differentiable matrix multiplication operators such as `torch.bmm`.

Compared with existing training frameworks such as PennyLane [6], it has several unique advantages: (1) It supports both dynamic and static computational graphs. Dynamic mode simulates each gate individually, so the state vector after each gate can be obtained for easy debugging. Static mode optimizes tensor network simulation by fusing unitary of multiple gates before applying to the state vector, reducing the computation amount. (2) It supports training in batch mode, which is important for QML tasks, while PennyLane cannot support

batched training because it processes batch with the 'For' loop. (3) All simulations can be accelerated with PyTorch's GPU acceleration support. (4) PyTorch's native automatic differentiation can be applied to train the gate parameters. Furthermore, QuantumEngine supports push-the-button conversion between PyTorch quantum circuit and IBM Qiskit QuantumCircuit, such that we can support convenient end-to-end training-to-deployment flow. It contains many ready-to-use templates, e.g., random and strongly-entangled layers. Parameter shift is also supported for gradient computations.

Figure 12 shows the training speed of 10-qubit parameterized quantum circuits containing 100 RX and 100 CRY gates vs. PennyLane. Since PennyLane processes batch with the 'For' loop, the training speed reduces linearly with the batch size. QuantumEngine supports tensorized batch processing on CPU/GPU, so the speed is not severely influenced. The training speed is 246 to 10^4 times faster than PennyLane.

IV. Evaluation

A. Evaluation Methodology

Benchmarks. We conduct experiments on 6 QML and 6 VQE tasks. QML are classification tasks including MNIST [33] 10, 4-class (0, 1, 2, 3), 2-class (3, 6); Vowel [14] 4-class (hid, hId, had, hOd); Fashion [63] 4-class (t-shirt/top, trouser, pullover, dress), and 2-class (dress, shirt). MNIST and Fashion use 95% images in 'train' split as the training set and 5% as the validation set. Due to the limited real QC resources, we randomly sample 300 images from the 'test' split as our test set and report their accuracy on the real quantum devices. However, we find 300 images can already have comparable accuracy to the whole testing set: on four circuits, the whole testing set acc/300-sample acc are 0.505/0.497, 0.284/0.283, 0.564/0.547, 0.272/0.287. The input images are 28×28. We center-crop them to 24×24 and down-sample them to 4×4 for 2 and 4 classifications; and 6×6 for MNIST-10, both with average pooling. Vowel-4 dataset (990 samples) is separated to train:validation:test = 6:1:3 and test with the whole test set. We perform principal component analysis (PCA) for the vowel features and take 10 most significant dimensions.

MNIST-2(-4) and Fashion-2(-4) use four logical qubits. MNIST-10 uses 10. To embed the classical images and vowel features to the quantum domain, we flatten them and encode them with rotation gates. For down-sampled 4×4 images, we use 4 qubits and 4 layers with 4 RY, 4 RZ, 4 RX, and 4 RY gates on each layer, respectively. There are total 16 gates to encode the 16 classical values as the rotation phases. For 6×6 images, we use 10 qubits and four layers with 10 RY, 10 RZ, 10 RX, and 6 RY gates on each layer, respectively. For 10 vowel features, we use 4 qubits and 3 layers with 4 RY, 4 RZ and 2 RX gates on each layer for encoding. For readout, we measure the expectation values on Pauli-Z basis and obtain a value [-1, 1] from each qubit. For 2-class, we sum the qubit 0 and 1, 2 and 3 respectively to get two values, which will be processed by Softmax to get probabilities. For 4 and 10-class, we use Softmax on expectation values to obtain probabilities.

TABLE I
Compiled circuit properties for QuantumNAS and baselines.

	Depth	#Gates (#1Q+#CNOT)	#Params	Acc.
Noise-Unaware	237	365 (299+66)	120	0.48
Random	45	100 (94+6)	36	0.86
Human	64	135 (124+11)	36	0.88
QuantumNAS	70	133 (123+10)	36	**0.89**
+ Pruning	59	116 (106+10)	22	**0.92**

For VQE, the goal is to find the low-energy eigenvalue of a target molecule by repeated measurements of the expectation value of the Hamiltonian of the molecule (as detailed in Section V). The molecules we study in this work contain H_2, LiH, H_2O, CH_4, and BeH_2. We use the Bravyi-Kitaev mapping [9] to transform a molecule Hamiltonian from its fermionic form to the qubit form. H_2, H_2O, LiH, and CH_4-6Q, CH_4-10Q, BeH_2, need 2, 6, 6, 6, 10, 15 logical qubits, respectively. VQE circuits are searched and trained on classical machines then deployed on real QC to obtain the eigenvalues.

Quantum Devices and Compiler Configurations. We use IBMQ quantum computers via Qiskit [28] APIs. We study 10 devices, with #qubits from 5 to 65 and Quantum Volume from 8 to 128. We also employ Qiskit for compilation. The optimization level is set to 2 except for level 3 for Noise-adaptive and Sabre baselines in Figure 13 and Table III. For searched qubit mapping, we set it as the 'initial_layout' of the compiler. QML/VQE experiments run 8192/2048 shots.

Circuit Design Spaces. We select six circuit design spaces, four from previous QML work, and name them with gates:

1) 'U3+CU3' – One block has a U3 layer with one U3 gate on each qubit, and a CU3 layer with ring connections, e.g., CU3(0, 1), CU3(1, 2), CU3(2, 3), CU3(3, 0).
2) 'ZZ+RY' [39] – One block contains one layer of ZZ gate, also with ring connections, and one RY layer.
3) 'RXYZ' [43] – One block has four layers: RX, RY, RZ, and CZ. There is one $\sqrt{\text{H}}$ layer before the blocks.
4) 'ZX+XX' [19] – according to their MNIST circuit design, one block has two layers: ZX and XX.
5) 'RXYZ+U1+CU3' [27] – according to their random circuit basis gate set, we design SuperCircuit in which one block has 11 layers in the order of RX, S, CNOT, RY, T, SWAP, RZ, H, $\sqrt{\text{SWAP}}$, U1 and CU3.
6) 'IBMQ Basis' [29] – we design SuperCircuit with basis gate set of IBMQ devices, in which one block has 6 layers in the order of RZ, X, RZ, SX, RZ, CNOT.

The SuperCircuits for space 1 to 4 contain 8 blocks; space 5 has 4 blocks; space 6 has 20 and does not have front sampling. The design spaces contain numerous SubCircuits, e.g.: RXYZ+U1+CU3 contains $4^{11\times4} = 3 \times 10^{26}$ SubCircuits.

Baselines. We have six baselines: (1) *Noise-unaware search*: the SubCircuits are searched with noise-free simulation. No noise information is involved. (2) *Random generation*: with the same gate set, we generate random circuits and constrain their #parameters the *same* as the QuantumNAS searched circuit for fair comparisons. We generate three different circuits and report the best. (3) *Human design*:

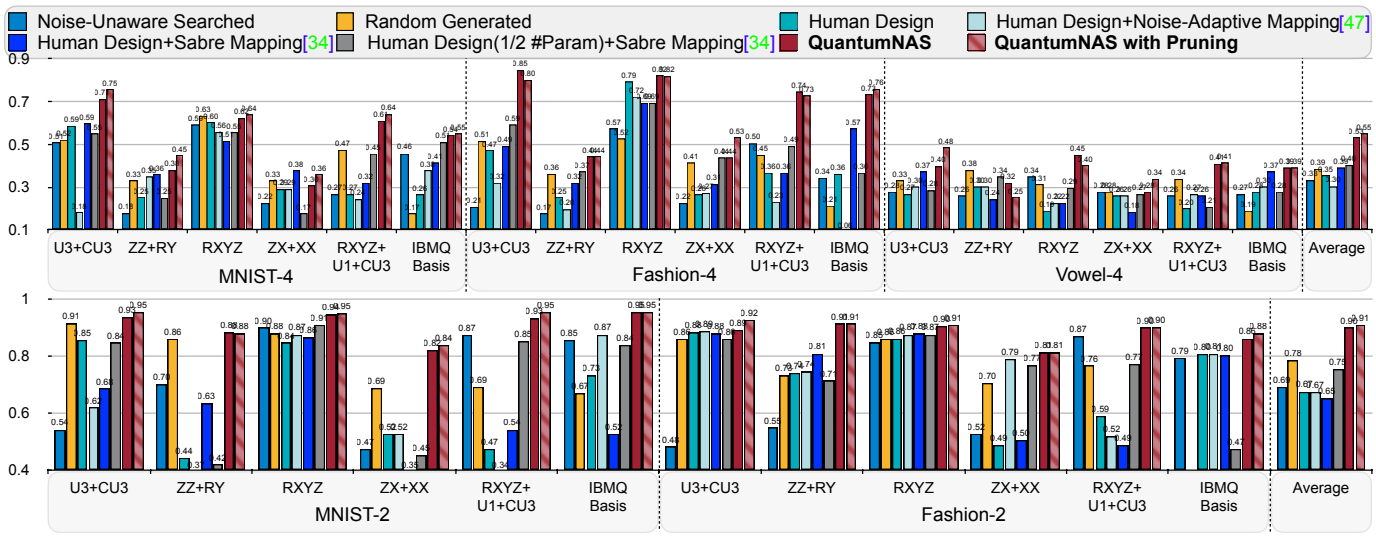

Fig. 13. QuantumNAS achieves highest accuracy on real QC devices (IBMQ-Yorktown) compared with baselines. Pruning further improves 2% on average.

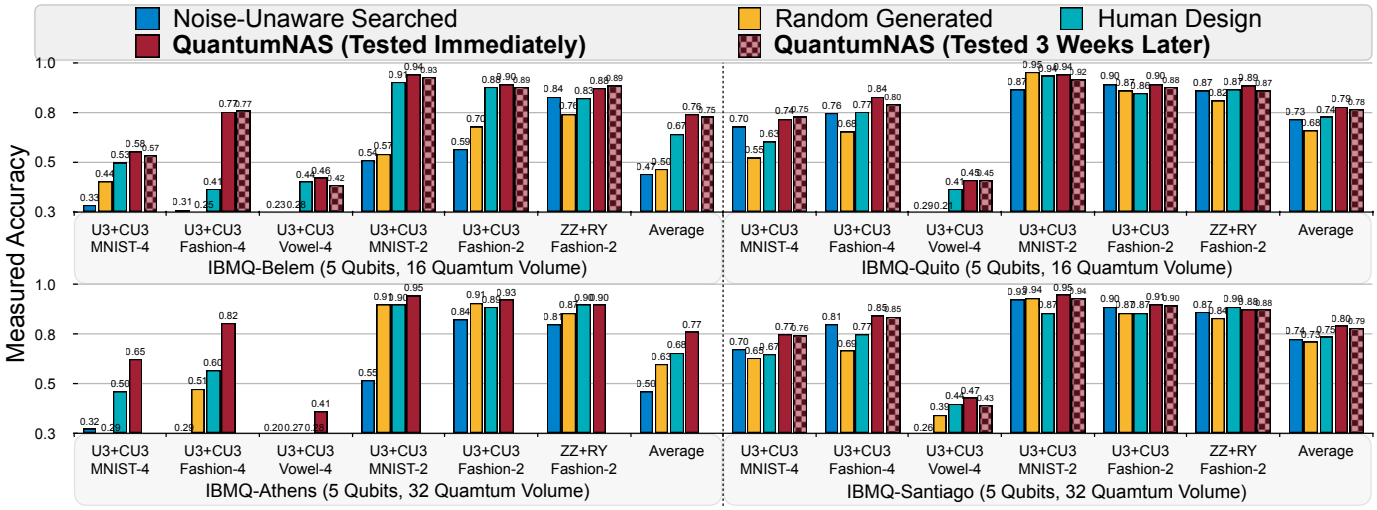

Fig. 14. On various 5-Qubit devices, QuantumNAS outperforms baseline designs.

we also make sure the same #parameters. For `U3+CU3`, `RXYZ+U1+CU3` and `IBMQ Basis` spaces, human design has full width in the several front blocks. For `ZZ+RY`, `RXYZ`, and `ZX+XX` spaces, we stack multiple blocks introduced in the original paper. (4) *Human design+noise-adaptive mapping*: the circuit has the same #parameters with QuantumNAS. The qubit mapping is optimized with state-of-the-art technique [47]. (5) *Human design+Sabre mapping*: the circuit has the same #parameters with QuantumNAS, the qubit mapping is optimized with Sabre [34]. (6) *Human design(1/2 #Param)+Sabre mapping*: similar to (6) with half #parameters. (7) For VQE, we have an additional *UCCSD* [3] baseline. For UCCSD of CH_4-10Q and BeH_2, the original circuit cannot be successfully run on IBMQ machines because of too many gates (>10,000), so we only take the front 1,000 gates.

SuperCircuit and SubCircuit Training Setups. For all searched SubCircuits and baselines, we use the same training setting for fair comparisons. We use Adam optimizer with initial learning rate 5e-3 and weight decay 1e-4, cosine learning rate scheduler. We train 200 epochs with batch size 256 for

TABLE II
DEVICE-SPECIFIC CIRCUIT HAS THE BEST ACCURACY.

Run on ↓ Searched for →	Yorktown	Belem	Santiago
Yorktown	**0.85**	0.60	0.54
Belem	0.67	**0.77**	0.43
Santiago	0.82	0.81	**0.85**

QML tasks; 1000 steps for VQE tasks with batch size 1. For QML, the objective is to minimize training loss, while VQE minimizes the eigenvalue.

SuperCircuits training has the same settings with SubCircuits, except adding a linear learning rate warm-up from 0 to 5e-3 in the first 30 epochs for QML and 150 steps for VQE. Restricted sampling is applied during the whole training process. We set the largest number of different layers as seven. An additional technique is to progressively shrink the lower bound of possible sampled SubCircuit #blocks to stabilize training. We use Nvidia TITAN RTX 2080 GPU. The time cost is shown in Figure 5.

Noise-Adaptive Evolutionary Co-Search Setups. The evolutionary search is conducted with inherited gate parameters

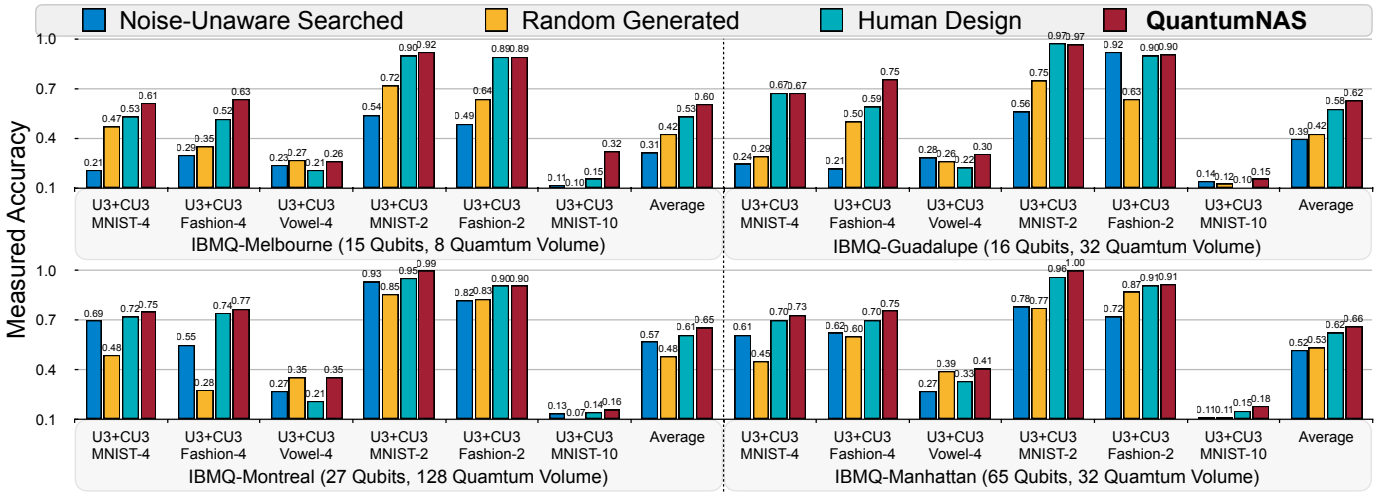

Fig. 15. QuantumNAS has high scalability and improves performance on large quantum machines.

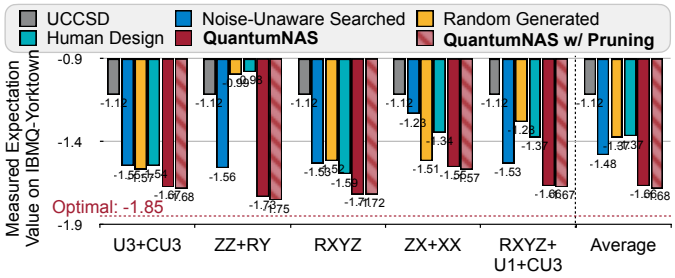

Fig. 16. H_2 VQE expectation value in different spaces. QuantumNAS consistently obtains the lowest expectation value.

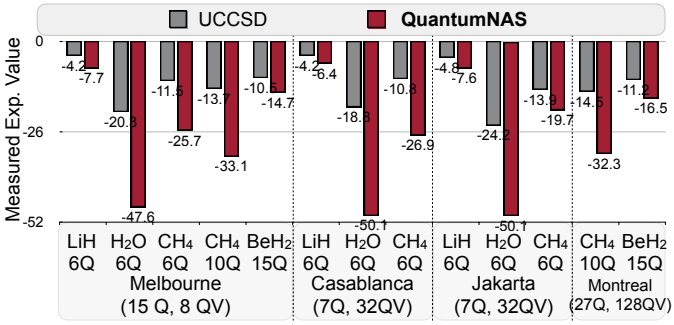

Fig. 17. VQE expectation value comparisons on various molecules.

on the validation set of QML tasks. For QML and VQE, the evolution engine searches 40 iterations with a population of 40, parents population 10, mutation population 20 with 0.4 mutation probability, and crossover population 10. The noise model is obtained from IBM's calibration data for the performance estimator, and the noise simulator is the Qiskit QASM simulator. We also run 8192 shots on simulators.

Iterative Pruning Setups. The searched SubCircuit is firstly trained from scratch. In pruning, we set five final pruning ratios, 0.1, 0.2, 0.3, 0.4, and 0.5. The starting ratio is 0.05. Pruning starts at step 0 and ends at half of total steps. We report the highest measured accuracy among the five ratios.

B. Experimental Results

Results on Four and Two Classifications. Figure 13 shows the measured accuracy on IBMQ-Yorktown (5Q) of QuantumNAS and 6 baselines on 5 QML tasks in 6 different design spaces. QuantumNAS achieves over 85% 4- and 95% 2-class accuracy and consistently outperforms baselines except for Vowel-4 in `ZZ+RY` space and MNIST-4 in `ZX+ XX` space. The statistics for Fashion-2 `U3+CU3` space are in Table I. The noise-unaware search only optimizes noise-free accuracy, which results in a deep circuit (237 depth) with low measured accuracy. `U3+CU3`, `RXYZ`, `RXYZ+U1+U3` and `IBM Basis` are better spaces as they always outperform the remaining two design spaces, and thus they are considered more noise-resilient. In addition, pruning brings an average of 2% for 4-class and 1% improvement for 2-class tasks. When the searched circuits contain only a small number of parameters, such as 7 in Vowel-4 `ZZ+RY`, removing any parameter will hurt the accuracy. For circuits with more parameters, such as 36 for MNIST-4 `U3+CU3`, the pruning ratio can be 50% while increasing accuracy by 4%. For `IBMQ Basis`, although its space is larger than `U3+CU3`, the accuracy is sometimes lower. Hence, a larger design space does not necessarily bring better final performance because of the higher search difficulty.

TABLE III
SEARCH WITH ESTIMATORS VS. REAL QC FOR FASHION-4 `U3+CU3`.

Method	Optimization Level 2					Optimization Level 3				
	York.	Bel.	Qui.	Ath.	Sant.	York.	Bel.	Qui.	Ath.	Sant.
Est.	**0.85**	0.77	**0.84**	**0.82**	**0.77**	0.69	**0.63**	**0.82**	0.84	**0.86**
Real QC	0.66	**0.80**	0.76	0.77	0.73	**0.70**	0.54	0.72	0.84	0.85

Results on Different Quantum Devices and Noise. Figure 14 shows QuantumNAS performance on various devices. For one task, QuantumNAS SubCircuits for each device are searched with the same SuperCircuit, but with noise models tailored for each device. For the machine with the smallest noise, IBMQ-Santiago, QuantumNAS also delivers 5% better accuracy on average. Additionally, we show the accuracy of QuantumNAS tested 3 weeks after search, which is slightly lower than tested immediately but still much higher than baselines. Therefore, even the noise characteristics change on a machine, QuantumNAS circuits are still noise-resilient. The results on Athens are unavailable since it is retired. Table II

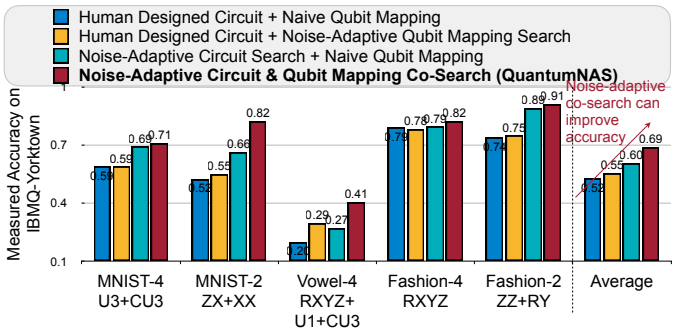

Fig. 18. Effect of circuit & qubit mapping co-search.

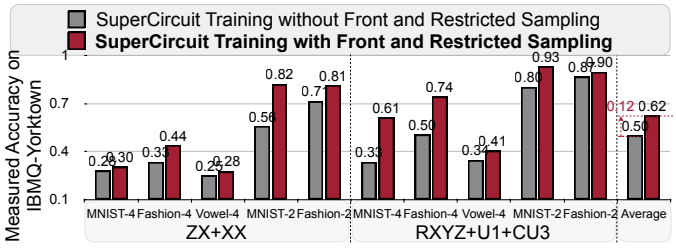

Fig. 19. Front and restricted sampling improves final accuracy.

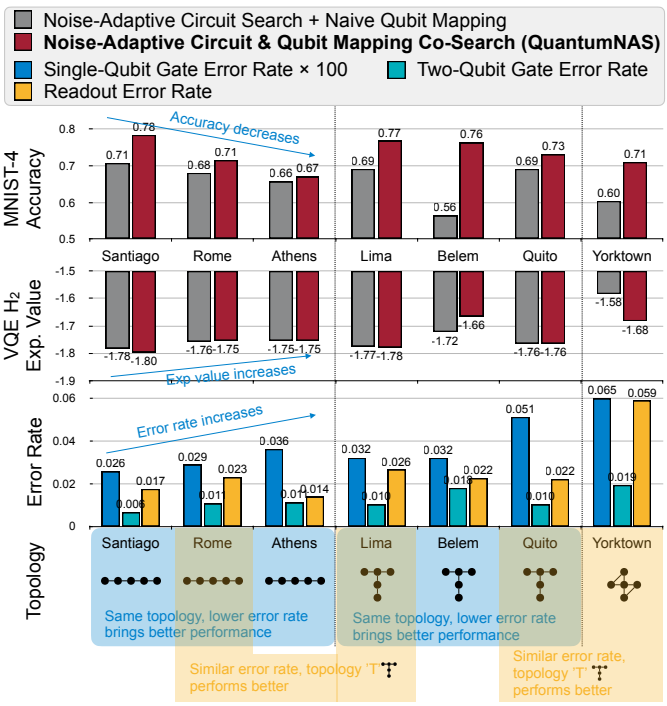

Fig. 20. Effect of qubit topology/error rate/qubit mapping to performance.

shows performance of circuits searched and run on different devices. Best performance is achieved when two devices are the same, which shows the necessity of device-specific circuits.

Scalability. We further show QuantumNAS results on larger machines with larger circuits. We search for circuits with 15, 16, 21, 21 qubits in `U3+CU3` space for machines with 15, 16, 27, 65 qubits in Figure 15. For the 21 qubit model, the SuperCircuit contains 1 block. QuantumNAS can achieve over 5% better accuracy. For even larger circuits for which classical simulations are infeasible, we can move the *whole pipeline* to quantum machines. Super/Subcircuit training can be done with parameter shift, and evolutionary search can directly evaluate SubCircuits on quantum machines. We add experiments on using real QC devices to evaluate SubCircuits in search as in Table III and compare with using noisy simulator. We experiment with Qiskit optimization levels 2 and 3. Due to queuing, we can only afford 20 search iterations which take ∼3 days. The accuracy of using real QC is similar to using simulators. In addition, the opt. level 3 cannot consistently improve accuracy over level 2 because QuantumNAS has already found a good mapping that is hard to optimize further.

Results on VQE Tasks. Figure 16 shows the VQE performance for H_2 in different spaces, measured on the IBMQ-Yorktown. The theoretical optimal value is -1.85. Estimated eigenvalues obtained by QuantumNAS are *consistently lower* than any other baselines. The UCCSD ansatz baseline is far from the optimal value as it is not adapted to the hardware noises. Pruning removes 50% parameters for all five circuit design spaces and can steadily reduce eigenvalues. Thus VQE circuits have a higher degree of redundancy over QML ones, making them *more amenable to pruning*. Figure 17 further shows the comparison results of QuantumNAS and UCCSD on LiH (6Q), H_2O (6Q), CH_4(4Q and 10Q) and BeH_2(15Q) on machines with 7Q, 15Q, and 27Q. Besides achieving lower measured expectation values, QuantumNAS can also reduce the theoretically trained values. For H_2O, the UCCSD noise-free trained expectation value is -49.6 while QuantumNAS has -52.4, indicating that QuantumNAS ansatz adapts to both the device and the molecule – a *hybrid device and problem ansatz*.

C. Performance Analysis

Accuracy Improvement Breakdown. We select five tasks and five design spaces to show the breakdown of accuracy improvements in Figure 18. We compare the QuantumNAS co-search to three baselines: (1) human baseline with no circuit or qubit mapping search, (2) noise-adaptive mapping search only, and (3) noise-adaptive circuit search only. Only searching circuit has larger accuracy improvements than only searching qubit mapping, as the space for circuit search is much larger, echoing our motivation. *The co-design of both aspects can further unlock 9% accuracy gain on average.* Figure 3 further shows the #parameters vs. accuracy curves. QuantumNAS can mitigate the negative effect of gate errors, and *delays the accuracy peak*, enabling more effective circuits.

Effect of Front and Restricted Sampling. Figure 19 shows the measured performance of SubCircuits on five tasks in `ZX+XX` and `RXYZ+U1+CU3` spaces. Since the front and restricted sampling improve the reliability of estimated relative performance, the searched SubCircuit is closer to the optimal one and achieves on average 12% higher final accuracy.

Effect of qubit topology/error rate/qubit mapping to performance and design choice. Figure 20 shows MNIST-4 and H_2 VQE performance on devices with various topologies and error rates. We have observations: (1) Comparing Santiago, Rome, and Athens, with the same topology, a lower error rate brings better performance. (2) Comparing Rome ('–') and

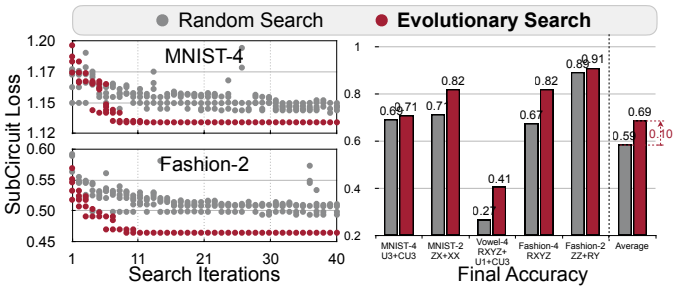

Fig. 21. Optimization curves and final accuracy of random/evo search.

TABLE IV
QUANTUMNAS SHOWS HIGHER ACCURACY THAN SHADOW CIRCUITS.

Device	Space	MNIST-4		Fashion-4		Vowel-4		MNIST-2		Fashion-2	
		D	Acc.	*D*	Acc.	*D*	Acc.	*D*	Acc	*D*	Acc
Santiago	Small	50	0.55	58	0.56	35	0.27	28	0.94	30	0.87
	Ours	**73**	**0.77**	**107**	**0.85**	**116**	**0.47**	**191**	**0.95**	**74**	**0.91**
Belem	Small	29	0.54	30	0.57	35	0.27	28	0.94	30	0.87
	Ours	**50**	**0.58**	**68**	**0.77**	**77**	**0.46**	**81**	**0.94**	**62**	**0.90**
Yorktown	Small	29	0.60	30	0.56	39	0.27	51	0.91	30	0.89
	Ours	**71**	**0.71**	**82**	**0.85**	**119**	**0.40**	**83**	**0.93**	**70**	**0.89**

Lima ('T'), Quito ('T') and Yorktown('+'), under similar error rates, 'T' topology brings better performance than the other two. (3) For qubit choices (mapping), the co-searched mapping can consistently outperform the naïve mapping. (4) For design choices of co-search, the average convergence iteration is 13.5, 14, 9.2 for 'T', '+', and '–' respectively. Therefore, we need a relatively larger iteration number for co-search on topology 'T' and '+' machines. That may be due to their more complicated connections than '–' topology.

Search in Small Design Space. We construct a small `U3+CU3` space that does not break into multiple blocks. All SubCircuits can be arbitrarily sampled without front sampling. The circuit depth is around 40. Comparisons with larger space with multiple blocks are shown in Table IV. Small space has consistently worse accuracy: although small circuits have less noise, it also has *smaller learning capacity*. QuantumNAS can find a better trade-off between noise and capacity.

Random Search vs. Evolutionary Search. Multiple candidate algorithms are applicable for the search stage. We compare the evolutionary with random search in Figure 21. The best performance of random search quickly saturates, while evolutionary can find SubCircuit and qubit mapping pair with lower loss, which delivers higher accuracy.

Effect of Pruning Ratios. Figure 22 shows the effect of different final pruning ratio for MNIST-2 `ZZ+RY` space and Fashion-2 `U3+CU3` spaces. As the final ratio increases, there exists a sweet spot where the positive effect of gate error reduction can overcome the negative effect of smaller circuit capacity so we can observe a peak accuracy.

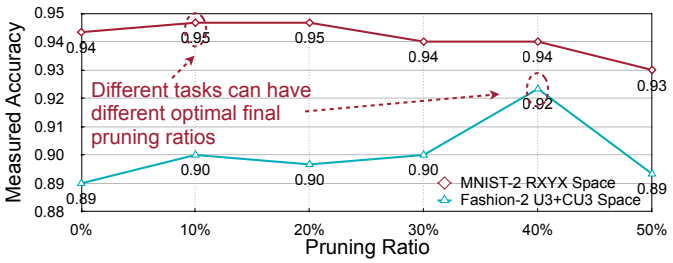

Fig. 22. Measured accuracy with different final pruning ratios.

V. Related Work

A. Quantum Machine Learning

Quantum machine learning (QML) [7], [25], [36], [38], [39] explores the training and evaluation of ML models on quantum devices. They have been shown to have potential speed-up over their classical counterparts in various tasks, including metric learning [39], data analysis [36], and principal component analysis [38]. In modern designs, QML models use variational quantum circuits with trainable parameters – quantum neural networks (QNNs). Various theoretical formulations for QNN have been proposed, e.g., quantum classifier [19], quantum convolution [27], and quantum Boltzmann machine [2], etc. Most prior works are exploratory and rely on classical simulation of small quantum systems [19]. Several works also propose to search circuits [17], [41], [65], [66] but they neither perform noise-adaptive co-search of the circuit and qubit mapping nor have extensive evaluations on real QC devices as in QuantumNAS.

B. Noise-Adaptive Quantum Compiling

A quantum compiler translates a quantum program written in high-level programming languages to hardware instructions. For NISQ systems [53], such translation needs to be noise-adaptive. As such, many noise-adaptive quantum compilers have been proposed. For example, various gate errors can be suppressed by dynamical decoupling [8], [23], [35], [59], composite pulses [10], [40], [45], randomized compiling [60], hidden inverses [64], qubit mapping [34], [47], [57], instruction scheduling [48], [62], and frequency tuning [16], [26], [58]. Typically, the key to these techniques is to find opportunities for local error cancellation within a quantum circuit. Instead, we propose to search for a quantum circuit and its qubit mapping pair that is the most resilient to noise. The flexibility in changing the quantum circuit itself gives us more freedom to build robustness into the quantum algorithms.

C. Quantum Simulation

Beyond ML, variational circuits can also be used to explore challenging quantum many-body physics problems. The first implementation of variational circuits was the Variational Quantum Eigensolver (VQE) [30], [44], [51] for quantum simulation of physical systems. Prior work showed that finding such an ansatz and learning their parameters is challenging. Different classes of ansatz designs have been proposed: (1) *Problem ansatz* [50], [51] is adapted to a target problem. E.g., UCCSD ansatz [3] is a design based on the structures

in a quantum system using computational chemistry models. (2) *Hardware ansatz* [30] is adapted to the properties of the computing hardware. Problem ansatz is shown to be typically more resilient to barren plateau than hardware ansatz [43]. In this work, our QuantumNAS aims to find a balanced and robust ansatz design via SuperCircuit-based search.

VI. Conclusion

We propose QuantumNAS, a framework for noise-adaptive co-search for the most robust variational circuit and qubit mapping. We leverage the SuperCircuit based search to explore in a ample design space efficiently. Iterative pruning is further leveraged to remove redundant gates in the searched circuits. Extensive experiments on QML and VQE tasks demonstrate the higher robustness and performance of QuantumNAS searched circuits over baseline designs. We also open-source our circuit training library, serving as a convenient infrastructure for future variational quantum research.

References

[1] A. Abbas, D. Sutter, C. Zoufal, A. Lucchi, A. Figalli, and S. Woerner, "The power of quantum neural networks," *Nature Computational Science*, vol. 1, no. 6, pp. 403–409, 2021.

[2] M. H. Amin, E. Andriyash, J. Rolfe, B. Kulchytskyy, and R. Melko, "Quantum boltzmann machine," *Physical Review X*, vol. 8, no. 2, p. 021050, 2018.

[3] R. J. Bartlett and M. Musiał, "Coupled-cluster theory in quantum chemistry," *Reviews of Modern Physics*, vol. 79, no. 1, p. 291, 2007.

[4] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini, "Parameterized quantum circuits as machine learning models," *Quantum Science and Technology*, vol. 4, no. 4, p. 043001, 2019.

[5] C. H. Bennett, D. P. DiVincenzo, J. A. Smolin, and W. K. Wootters, "Mixed-state entanglement and quantum error correction," *Physical Review A*, vol. 54, no. 5, p. 3824, 1996.

[6] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, M. S. Alam, S. Ahmed, J. M. Arrazola, C. Blank, A. Delgado, S. Jahangiri *et al.*, "Pennylane: Automatic differentiation of hybrid quantum-classical computations," *arXiv preprint arXiv:1811.04968*, 2018.

[7] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.

[8] M. J. Biercuk, H. Uys, A. P. VanDevender, N. Shiga, W. M. Itano, and J. J. Bollinger, "Optimized dynamical decoupling in a model quantum memory," *Nature*, vol. 458, no. 7241, pp. 996–1000, 2009.

[9] S. B. Bravyi and A. Y. Kitaev, "Fermionic quantum computation," *Annals of Physics*, vol. 298, no. 1, pp. 210–226, 2002.

[10] K. R. Brown, A. W. Harrow, and I. L. Chuang, "Arbitrarily accurate composite pulse sequences," *Physical Review A*, vol. 70, no. 5, p. 052318, 2004.

[11] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, "Trapped-ion quantum computing: Progress and challenges," *Applied Physics Reviews*, vol. 6, no. 2, p. 021314, 2019.

[12] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," *arXiv preprint arXiv:1908.09791*, 2019.

[13] Y. Cao, J. Romero, J. P. Olson, M. Degroote, P. D. Johnson, M. Kieferová, I. D. Kivlichan, T. Menke, B. Peropadre, N. P. Sawaya *et al.*, "Quantum chemistry in the age of quantum computing," *Chemical reviews*, vol. 119, no. 19, pp. 10 856–10 915, 2019.

[14] D. Deterding, "Speaker normalisation for automatic speech recognition," PhD thesis, University of Cambridge, 1989.

[15] Y. Ding and F. T. Chong, "Quantum computer systems: Research for noisy intermediate-scale quantum computers," *Synthesis Lectures on Computer Architecture*, vol. 15, no. 2, pp. 1–227, 2020.

[16] Y. Ding, P. Gokhale, S. F. Lin, R. Rines, T. Propson, and F. T. Chong, "Systematic crosstalk mitigation for superconducting qubits via frequency-aware compilation," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2020, pp. 201–214.

[17] Y. Du, T. Huang, S. You, M.-H. Hsieh, and D. Tao, "Quantum circuit architecture search: error mitigation and trainability enhancement for variational quantum solvers," *arXiv preprint arXiv:2010.10217*, 2020.

[18] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," *arXiv preprint arXiv:1411.4028*, 2014.

[19] E. Farhi and H. Neven, "Classification with quantum neural networks on near term processors," *arXiv preprint arXiv:1802.06002*, 2018.

[20] D. Gottesman, "An introduction to quantum error correction and fault-tolerant quantum computation," in *Quantum information science and its contributions to mathematics, Proceedings of Symposia in Applied Mathematics*, vol. 68, 2010, pp. 13–58.

[21] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 212–219.

[22] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, "Single path one-shot neural architecture search with uniform sampling," in *European Conference on Computer Vision*. Springer, 2020, pp. 544–560.

[23] E. L. Hahn, "Spin echoes," *Physical review*, vol. 80, no. 4, p. 580, 1950.

[24] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations," *Physical review letters*, vol. 103, no. 15, p. 150502, 2009.

[25] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, no. 7747, pp. 209–212, 2019.

[26] F. Helmer, M. Mariantoni, A. G. Fowler, J. von Delft, E. Solano, and F. Marquardt, "Cavity grid for scalable quantum computation with superconducting circuits," *EPL (Europhysics Letters)*, vol. 85, no. 5, p. 50007, 2009.

[27] M. Henderson, S. Shakya, S. Pradhan, and T. Cook, "Quanvolutional neural networks: powering image recognition with quantum circuits," *Quantum Machine Intelligence*, vol. 2, no. 1, pp. 1–9, 2020.

[28] IBM, "Ibm quantum." [Online]. Available: https://quantum-computing.ibm.com/

[29] Q. IBM, Apr 2021. [Online]. Available: https://qiskit.org/textbook/ch-quantum-hardware/calibrating-qubits-pulse.html

[30] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature*, vol. 549, no. 7671, pp. 242–246, 2017.

[31] C. Kokail, C. Maier, R. van Bijnen, T. Brydges, M. K. Joshi, P. Jurcevic, C. A. Muschik, P. Silvi, R. Blatt, C. F. Roos *et al.*, "Self-verifying variational quantum simulation of lattice models," *Nature*, vol. 569, no. 7756, pp. 355–360, 2019.

[32] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver, "A quantum engineer's guide to superconducting qubits," *Applied Physics Reviews*, vol. 6, no. 2, p. 021318, 2019.

[33] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[34] G. Li, Y. Ding, and Y. Xie, "Tackling the qubit mapping problem for nisq-era quantum devices," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 1001–1014.

[35] D. A. Lidar, "Review of decoherence free subspaces, noiseless subsystems, and dynamical decoupling," *Adv. Chem. Phys*, vol. 154, pp. 295–354, 2014.

[36] S. Lloyd, S. Garnerone, and P. Zanardi, "Quantum algorithms for topological and geometric analysis of data," *Nature communications*, vol. 7, no. 1, pp. 1–7, 2016.

[37] S. Lloyd, M. Mohseni, and P. Rebentrost, "Quantum algorithms for supervised and unsupervised machine learning," *arXiv preprint arXiv:1307.0411*, 2013.

[38] S. Lloyd, M. Mohseni, and P. Rebentrost, "Quantum principal component analysis," *Nature Physics*, vol. 10, no. 9, pp. 631–633, 2014.

[39] S. Lloyd, M. Schuld, A. Ijaz, J. Izaac, and N. Killoran, "Quantum embeddings for machine learning," *arXiv preprint arXiv:2001.03622*, 2020.

[40] G. H. Low, T. J. Yoder, and I. L. Chuang, "Optimal arbitrarily accurate composite pulse sequences," *Physical Review A*, vol. 89, no. 2, p. 022341, 2014.

[41] Z. Lu, P.-X. Shen, and D.-L. Deng, "Markovian quantum neuroevolution for machine learning," *arXiv preprint arXiv:2012.15131*, 2020.

[42] E. Magesan, J. M. Gambetta, and J. Emerson, "Characterizing quantum gates via randomized benchmarking," *Physical Review A*, vol. 85, no. 4, p. 042311, 2012.

[43] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, "Barren plateaus in quantum neural network training landscapes," *Nature communications*, vol. 9, no. 1, pp. 1–6, 2018.

[44] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, "The theory of variational hybrid quantum-classical algorithms," *New Journal of Physics*, vol. 18, no. 2, p. 023023, 2016.

[45] J. T. Merrill and K. R. Brown, "Progress in compensating pulse sequences for quantum computation," *Quantum Information and Computation for Chemistry*, pp. 241–294, 2014.

[46] N. Moll, P. Barkoutsos, L. S. Bishop, J. M. Chow, A. Cross, D. J. Egger, S. Filipp, A. Fuhrer, J. M. Gambetta, M. Ganzhorn *et al.*, "Quantum optimization using variational algorithms on near-term quantum devices," *Quantum Science and Technology*, vol. 3, no. 3, p. 030503, 2018.

[47] P. Murali, J. M. Baker, A. Javadi-Abhari, F. T. Chong, and M. Martonosi, "Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 1015–1029.

[48] P. Murali, D. C. McKay, M. Martonosi, and A. Javadi-Abhari, "Software mitigation of crosstalk on noisy intermediate-scale quantum computers," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 1001–1016.

[49] M. A. Nielsen and I. Chuang, "Quantum computation and quantum information," 2002.

[50] P. J. O'Malley, R. Babbush, I. D. Kivlichan, J. Romero, J. R. McClean, R. Barends, J. Kelly, P. Roushan, A. Tranter, N. Ding *et al.*, "Scalable quantum simulation of molecular energies," *Physical Review X*, vol. 6, no. 3, p. 031007, 2016.

[51] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'brien, "A variational eigenvalue solver on a photonic quantum processor," *Nature communications*, vol. 5, no. 1, pp. 1–7, 2014.

[52] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4095–4104.

[53] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, p. 79, 2018.

[54] P. Rebentrost, M. Mohseni, and S. Lloyd, "Quantum support vector machine for big data classification," *Physical review letters*, vol. 113, no. 13, p. 130503, 2014.

[55] M. Schuld, *Supervised learning with quantum computers*. Springer, 2018.

[56] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[57] S. S. Tannu and M. K. Qureshi, "Not all qubits are created equal: a case for variability-aware policies for nisq-era quantum computers," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 987–999.

[58] R. Versluis, S. Poletto, N. Khammassi, B. Tarasinski, N. Haider, D. J. Michalak, A. Bruno, K. Bertels, and L. DiCarlo, "Scalable quantum circuit and control for a superconducting surface code," *Physical Review Applied*, vol. 8, no. 3, p. 034021, 2017.

[59] L. Viola, E. Knill, and S. Lloyd, "Dynamical decoupling of open quantum systems," *Physical Review Letters*, vol. 82, no. 12, p. 2417, 1999.

[60] J. J. Wallman and J. Emerson, "Noise tailoring for scalable quantum computation via randomized compiling," *Physical Review A*, vol. 94, no. 5, p. 052325, 2016.

[61] P. Wittek, *Quantum machine learning: what quantum computing means to data mining*. Academic Press, 2014.

[62] X.-C. Wu, D. M. Debroy, Y. Ding, J. M. Baker, Y. Alexeev, K. R. Brown, and F. T. Chong, "Tilt: Achieving higher fidelity on a trapped-ion linear-tape quantum computing architecture," *arXiv preprint arXiv:2010.15876*, 2020.

[63] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[64] B. Zhang, S. Majumder, P. H. Leung, S. Crain, Y. Wang, C. Fang, D. M. Debroy, J. Kim, and K. R. Brown, "Hidden inverses: Coherent error cancellation at the circuit level," *arXiv preprint arXiv:2104.01119*, 2021.

[65] S.-X. Zhang, C.-Y. Hsieh, S. Zhang, and H. Yao, "Differentiable quantum architecture search," *arXiv preprint arXiv:2010.08561*, 2020.

[66] S.-X. Zhang, C.-Y. Hsieh, S. Zhang, and H. Yao, "Neural predictor based quantum architecture search," *arXiv preprint arXiv:2103.06524*, 2021.

[67] M. Zhu and S. Gupta, "To prune, or not to prune: exploring the efficacy of pruning for model compression," *arXiv preprint arXiv:1710.01878*, 2017.