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]*University of Texas at Austin* [4]*University of Chicago*
https://qmlsys.mit.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 has 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 from 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 14 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 the `TorchQuantum` library for fast training of parameterized quantum circuits to facilitate future research.

***Keywords*-Quantum Computing; Quantum Noise; Variational Quantum Algorithm; Quantum Machine Learning; Neural Networks; Qubit Mapping; VQE; QNN**

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 [1], database search [2], chemistry [3]–[5] and machine learning [6]–[10], etc. In the recent two decades, QC hardware has witnessed rapid progress by virtue of breakthroughs in physical implementation technologies.

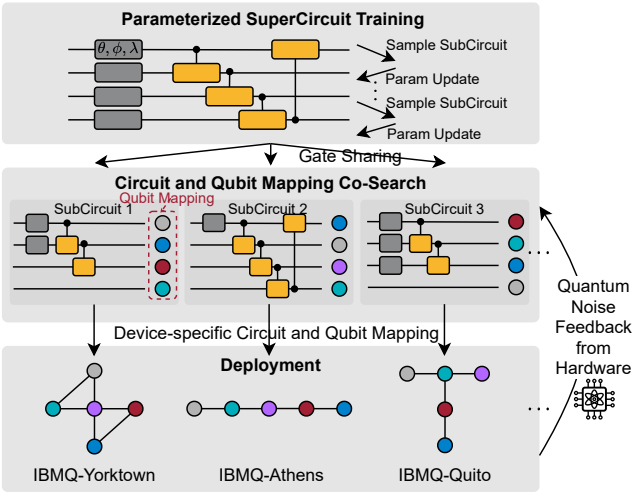

Figure 1. Noise-adaptive circuit and qubit mapping co-search. A gate-sharing SuperCircuit that contains numerous parameter subsets (SubCircuit) is firstly trained. Then we perform an evolutionary search with the quantum noise feedback to find the most robust circuit and qubit mapping.

Despite the exciting advancements, we are still expected to reside in the Noisy Intermediate Scale Quantum (NISQ) [11] stage for multiple years before entering the Fault-Tolerant era [12], [13]. 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 [14]–[16] 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 [17], [18] 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 improve the noise resilience of QC via a co-design strategy

2378-203X/22/$31.00 ©2022 IEEE
DOI 10.1109/HPCA53966.2022.00057

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 17:47:05 UTC from IEEE Xplore. Restrictions apply.

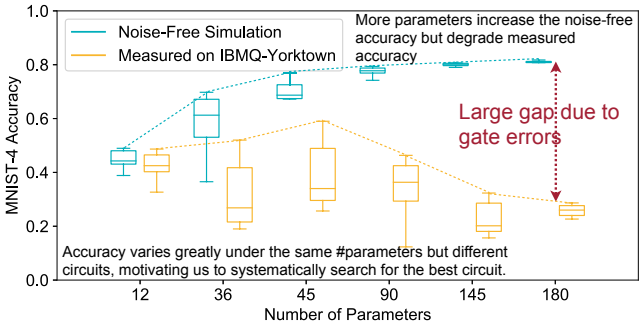

Figure 2. MNIST-4 on noise-free simulator / real QC. More parameters increase the noise-free accuracy but degrade measured accuracy due to larger gate errors. Accuracy varies greatly under the same #parameters but different circuits, motivating us to search for the best circuit systematically.

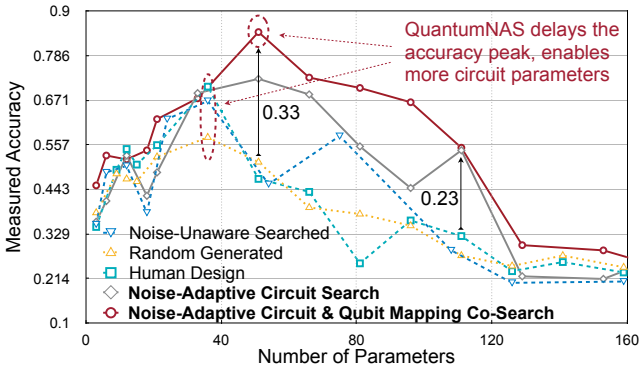

Figure 3. Accuracy vs. #parameters of multiple methods. The accuracy of conventional designs quickly saturates then drops. QuantumNAS mitigates the quantum noise and delays the peak of the curve, allowing larger model capacity and higher accuracy (up to 33% higher).

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 variational quantum circuits (trainable circuits with parameterized quantum gates) 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 [19]–[24]. However, we have five 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. (5) We experimentally demonstrate the feasibility of training circuits on real QC with on-device gradient computation.

Finally, on top of the searched circuit and qubit mapping, we further propose a fine-grained *pruning* technique 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.

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 17:47:05 UTC from IEEE Xplore. Restrictions apply.

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 14 quantum computers, observing significant improvements over baselines. ❺ **Open-Source QC Library**: To facilitate future research in QML and variational quantum simulation, we release `TorchQuantum`, a PyTorch-based GPU-accelerated library to enable fast training of parameterized quantum circuits (over 200× faster than the PennyLane [25]). It also supports push-the-button deployment of trained circuits on real quantum devices.

II. Background and Motivation

A. Quantum Basics

Qubits. The power of quantum computation stems from its fundamentally unique way of storing and manipulating information [26], [27]. Unlike a conventional bit, a quantum bit (*qubit*) 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. Finding the best quantum circuits for a computational task is non-trivial [27] – a realistic, robust quantum circuit must: (1) faithfully express the desired transformation; (2) complete in an efficient number of steps; (3) be able to be implemented on hardware with reasonable fidelity.

Operational Noises. In real QC, errors occur due to imperfect control signals, unwanted interactions between qubits, or interference from the environment [28]–[30]. Thus, qubits undergo *decoherence error* over time, and quantum gates introduce *operation errors* (e.g., coherent/stochastic errors) into the system. These systems need to be characterized [30] and calibrated [31] frequently to mitigate noise impacts. So noise-adaptive techniques in QC algorithms, circuits, and devices are critical for operating quantum computers.

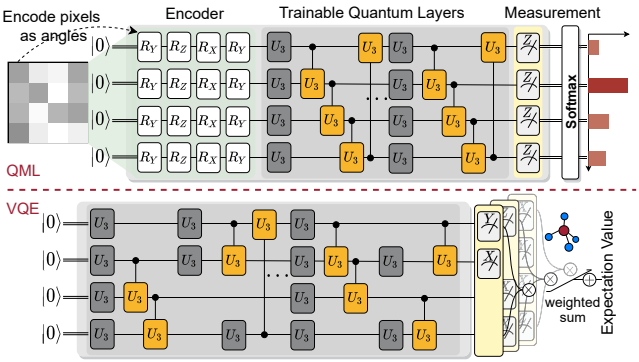

Figure 4. Example circuits for QML and VQE tasks.

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. Variational methods have shown huge potentials in applications such as quantum ML [6], [32]–[34], numerical analysis [35], [36], quantum simulation [3], [4], [37]–[39], and optimizations [8], [40].

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. 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" [41] is a phenomenon when the cost function landscape is flat, making a variational circuit untrainable with gradient-based optimizations.

Quantum Neural Network (QNN) is a promising application of variational quantum circuits [42]. 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

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 17:47:05 UTC from IEEE Xplore. Restrictions apply.

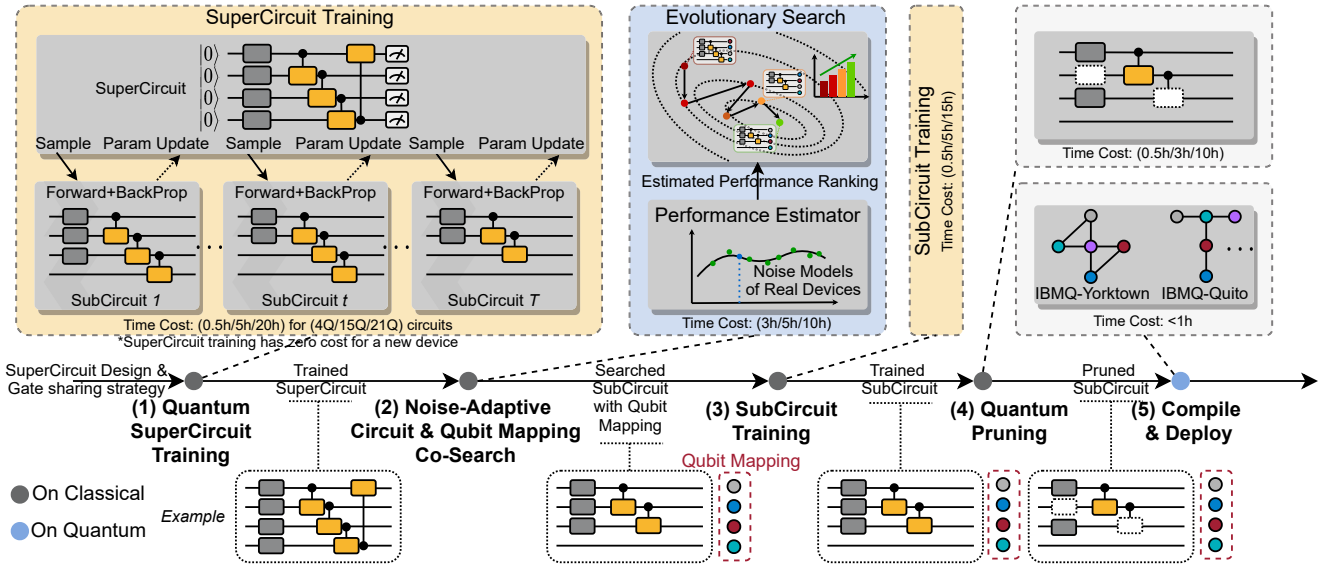

Figure 5. QuantumNAS Overview. (1) A SuperCircuit is trained by iteratively sampling and updating parameter subsets (SubCircuits). The parameters from SuperCircuit and the simulator with practical noise models can provide an accurate final performance ranking estimation of SubCircuits. (2) Evolutionary 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.

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. We show several sampled example SubCircuits in the diagram. *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 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 a 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 using the SubCircuit and update that subset of parameters of SuperCircuit. Intuitively, training a SuperCircuit is simultaneously training all SubCircuits in the design space. All gates in SuperCircuit are single/two-qubit gates, thus are local interactions in variational ansatz.

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} means #quantum devices to execute the circuit. N_{search} means #evaluated circuits during search. N_{train}/N_{eval} means #circuit running iterations in

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 17:47:05 UTC from IEEE Xplore. Restrictions apply.

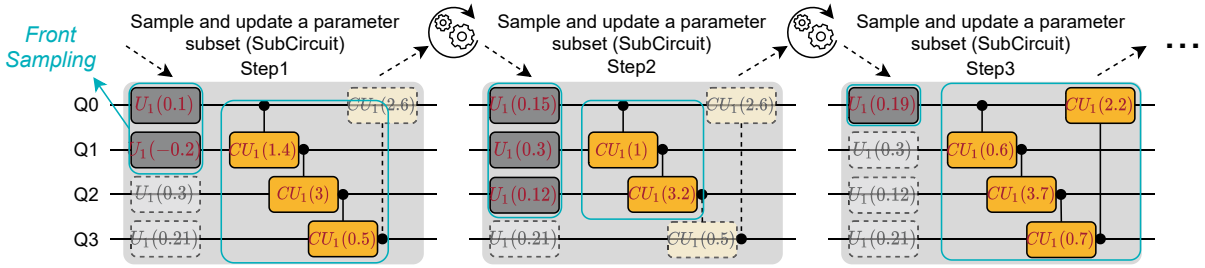

Figure 6. SuperCircuit training. At each step, a subset of SuperCircuit parameters (SubCircuit) is sampled and then updated.

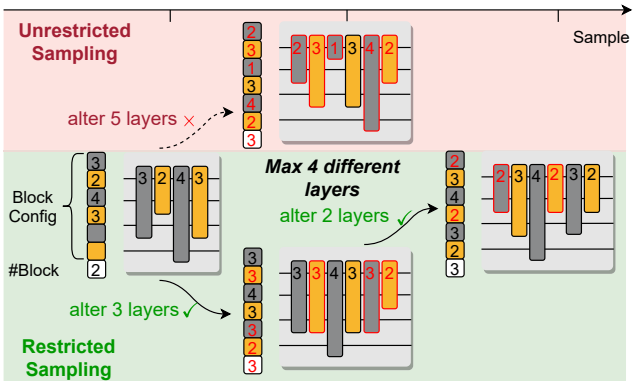

Figure 7. Restricted sampling constrains #layers that are different between two steps. It improves SubCircuit consistency thus stabilizes the SuperCircuit training process.

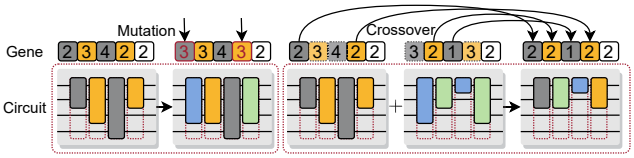

Figure 8. SubCircuit mutation and crossover in evolutionary search.

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. In the leftmost example, only the front two U_1 gates and the front three CU_1 gates are sampled. So in that step, only those five parameters are updated. 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.

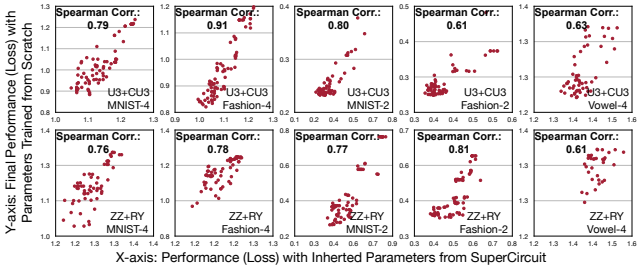

Figure 9. Strong correlation between performance with inherited parameters from SuperCircuit and parameters trained from scratch.

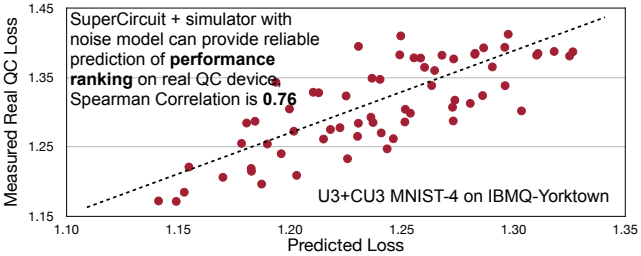

Figure 10. Reliability of the performance estimator. The estimated loss can accurately indicate the final loss.

Therefore, the training process is stabilized as the sampling variance is under control. As in Figure 7, the upper path is unrestricted sampling where the two SubCircuits differ by 5 layers. In the bottom path, restricted sampling limits the layer differences to 3, bringing 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 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 keeps a population of pairs and searches for high-performance candidates. 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

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 17:47:05 UTC from IEEE Xplore. Restrictions apply.

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. If the qubit mapping sub-gene contains a repeated qubit, we will replace the repeated one with the first unused qubit. 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 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. Then, 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 (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 the success rate, and l is the 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 method.

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

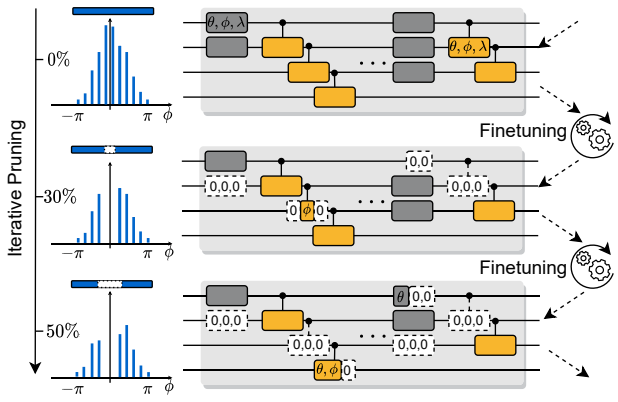

Figure 11. Iterative pruning and finetuning remove small-magnitude parameters. thus reducing the number of compiled gates.

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, a strong positive correlation. Thus, 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, inspired by the classical NN pruning [43]–[45]. 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 with iterative pruning and finetuning. Third, some gates, such as `U3`, contain multiple parameters. Partially removing the parameters can also bring benefits. #compiled gates of U3(θ, ϕ, λ), U3($\mathbf{0}, \phi, \lambda$), U3($\theta, \phi, \mathbf{0}$), U3($\theta, \mathbf{0}, \lambda$), U3($\theta, \mathbf{0}, \mathbf{0}$), U3($\mathbf{0}, \phi, \mathbf{0}$) and U3($\mathbf{0}, \mathbf{0}, \lambda$) are 5, 1, 4, 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 to remove the circuit parameters in a fine-grained manner, shown in Figure 11. 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 [46]: $r_{now} = r_{final} + (r_{initial} - r_{final})\left(1 - \frac{s_{now} - s_{begin}}{s_{end} - s_{begin}}\right)^3$ 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

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 17:47:05 UTC from IEEE Xplore. Restrictions apply.

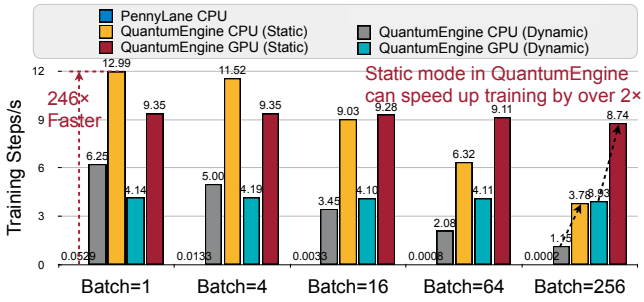

Figure 12. Training speed of `TorchQuantum` vs. PennyLane [25]. `TorchQuantum` provides at least 246× speedup.

gates and fewer noise sources after compilation, the accuracy of the circuit can be further increased by up to 9%.

E. `TorchQuantum` Library

To accelerate parameterized quantum circuit training in this work, we build a PyTorch library named `TorchQuantum`. 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`.

There exists several QNN training frameworks such as PennyLane [25]. The major advantage is its flexible interfaces to various frameworks such as PyTorch, TensorFlow, JAX, Keras, and NumPy. However, the operations are not implemented using native ones in those frameworks, so off-the-shelf optimizations of those frameworks cannot be used. Moreover, Pennylane can only use *parameter shift* to obtain gradients, which is inherently sequential, so no parallelization on batch and gate dimension can be achieved. Compared with Pennylane, QuantumEngine 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 (statevector simulation). Static mode optimizes tensor network simulation by fusing unitary of multiple gates before applying to the state vector, reducing the computation amount (tensor network simulation). (2) It supports both parameter shift and back-propagation training. In back-propagation mode, training is highly parallelized. PennyLane only supports parameter shift and processes batch with inefficient '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 parameters.

Furthermore, `TorchQuantum` 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. All steps in QuantumNAS are implemented with it. The library has great potential to accelerate research in parameterized QC, especially for QML, VQE, etc.

Table I
BENCHMARK INFORMATION SUMMARY.

Task	Class	Input Size	#qubit	Encoder (Mapping)
MNIST-10	`0-9`	6×6	10	10`RY`,10`RZ`,10`RX`,6`RY`
MNIST-4	`0,1,2,3`	4×4	4	4`RY`,4`RZ`,4`RX`,4`RY`
MNIST-2	`3,6`	4×4	4	4`RY`,4`RZ`,4`RX`,4`RY`
Fashion-4	`t-shirt,trouser pullover,dress`	4×4	4	4`RY`,4`RZ`,4`RX`,4`RY`
Fashion-2	`dress,shirt`	4×4	4	4`RY`,4`RZ`,4`RX`,4`RY`
Vowel-4	`hid,hId,had,hOd`	10	4	4`RY`,4`RZ`,2`RX`
H_2	VQE	–	2	Bravyi-Kitaev [47]
H_2O	VQE	–	6	Bravyi-Kitaev
LiH	VQE	–	6	Bravyi-Kitaev
CH_4-6Q	VQE	–	6	Bravyi-Kitaev
CH_4-10Q	VQE	–	10	Bravyi-Kitaev
BeH_2	VQE	–	15	Bravyi-Kitaev

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. `TorchQuantum` 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 benchmark information is shown in Table I. 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 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. 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.

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 as in Table I. VQE

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 17:47:05 UTC from IEEE Xplore. Restrictions apply.

Table II
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**

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 [48] 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 IV. 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 6 circuit design spaces, 4 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' [49] – One block contains one layer of ZZ gate, also with ring connections, and one RY layer.
3) 'RXYZ' [41] – One block has four layers: RX, RY, RZ, and CZ. There is one $\sqrt{\text{H}}$ layer before the blocks.
4) 'ZX+XX' [50] – according to their MNIST circuit design, one block has two layers: ZX and XX.
5) 'RXYZ+U1+CU3' [51] – 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' [31] – 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 $4^{11\times4} = 3\times10^{26}$ 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*: 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. The last layer of human designs may not have full width to make sure the same total number of parameters. (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 [15]. (5) *Human design+Sabre mapping*: the circuit has the same #parameters with QuantumNAS, the qubit mapping is optimized with Sabre [14]. (6) *Human design(1/2 #Param)+Sabre mapping*: similar to (6) with half #parameters. (7) For VQE, we have an additional *UCCSD* [52] 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 for real QC experiments and report the results of the full circuit using the Qiskit noisy simulator.

Table III
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**

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 for 200 epochs with batch size 256 for 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 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

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 17:47:05 UTC from IEEE Xplore. Restrictions apply.

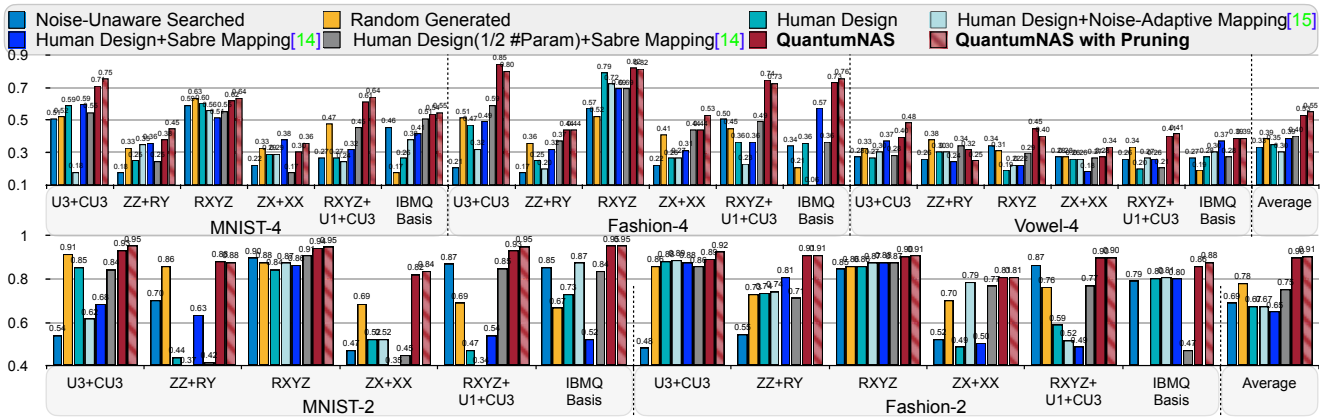

Figure 13. QuantumNAS achieves the highest accuracy on real QC devices (IBMQ-Yorktown). Pruning further improves 2% on average.

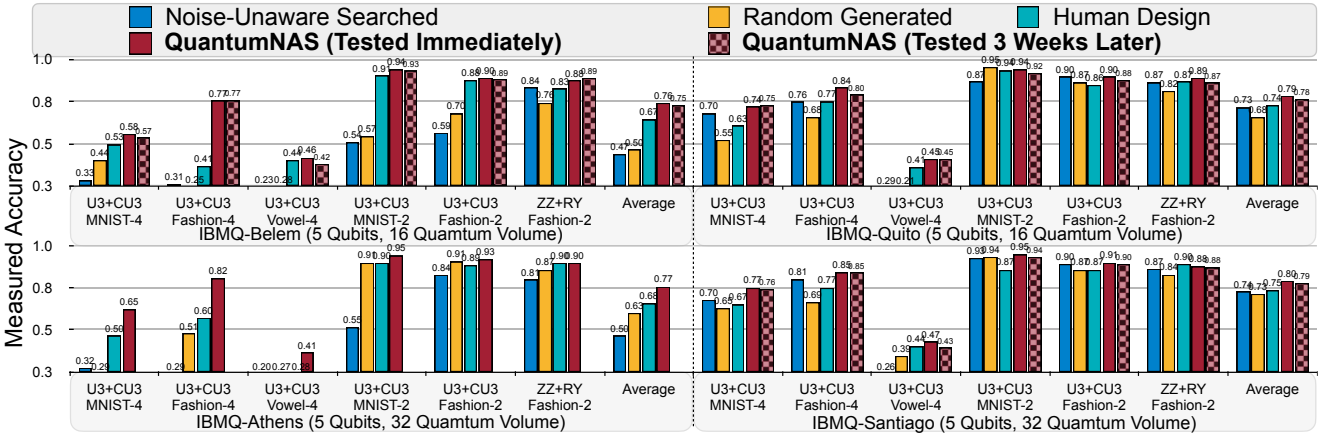

Figure 14. On four 5-Qubit real QC devices, the design searched by QuantumNAS outperforms baseline designs with higher measured accuracy.

QuantumNAS and 6 baselines on 5 QML tasks in 6 different design spaces. QuantumNAS achieves over 85% 4-class 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 II. 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%. In Table II, pruning removes 14 parameters and reduces depth by 11. The accuracy is improved by 3% since the pruned circuit has similar noise-free accuracy but fewer gates and less noise. 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.

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, although the baseline methods achieve higher accuracy than Melbourne and Guadalupe, QuantumNAS can still deliver 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. One reason is that the machines are calibrated by the IBMQ at least once a day, so noises are not far from the calibration point. Therefore, even the noise characteristics change on a machine in calibration interval, QuantumNAS circuits are still noise-resilient. The results on Athens are unavailable since it is retired. Table III 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

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 17:47:05 UTC from IEEE Xplore. Restrictions apply.

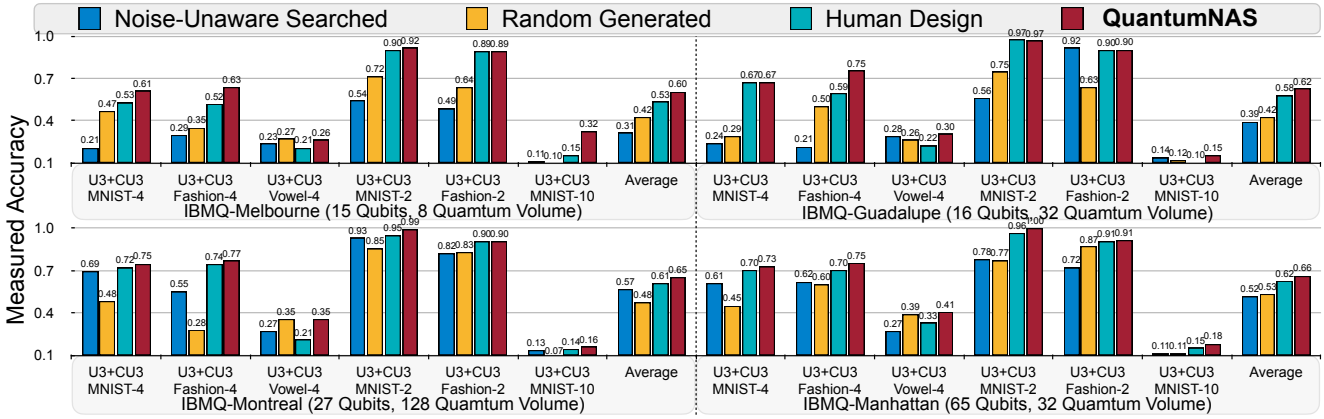

Figure 15. QuantumNAS has high scalability and can maintain the performance advantage with large models on large devices (in terms of #qubits).

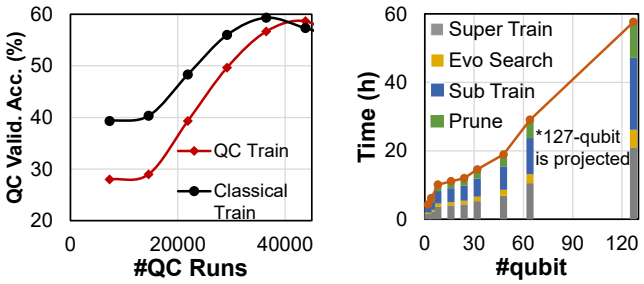

Figure 16. (a) Scalable QNN training for MNIST-4 on real quantum machines with parameter shift. (b) Runtime breakdown with different numbers of qubits.

Table IV
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

Table V
CIRCUIT TRAINING ON REAL QC WITH PARAMETER SHIFT IS FEASIBLE.

Task	MNIST-4	MNIST-2	Fashion-4	Fashion-2	Vowel-4	Fashion-2	Fashion-2
Machine	Jarkata	Jarkata	Manila	Santiago	Lima	Guadalupe	Montreal
Qubit Usage	4	4	4	4	4	16	16
Classical	0.59	0.79	0.54	0.89	0.31	0.70	0.74
QC Train	0.59	0.83	0.49	0.84	0.34	0.71	0.74

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 demonstrate the high feasibility of training circuits on quantum machines using parameter shift. We train SubCircuits for different tasks on different machines as in Table V. Results show comparable real QC test accuracy of training on real QC and classical simulators. We also show the training curve of MNIST-4 in Figure 16 left.

We add experiments on using real QC devices to evaluate SubCircuits in search as in Table IV 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. The observation aligns with recent work on QC characterization [53]. One reason is that QuantumNAS has already found a good mapping that is hard to optimize further. We further show the runtime of QuantumNAS training a VQE model fully on real QC with different qubit numbers in Figure 16 right. The SuperCircuit contains 32 parameters in ZZ+RY space. The runtime is an *upper bound* as we assume the largest SubCircuit is used. The pruning ratio is set as 50%. The runtime increases approximately **linearly** as the qubit number increases. The projected 127 qubit runtime is around 57 hours. Thus QuantumNAS has **high scalability.**

Results on VQE Tasks. Figure 17 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 18 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*. For CH_4-10Q, we also

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 17:47:05 UTC from IEEE Xplore. Restrictions apply.

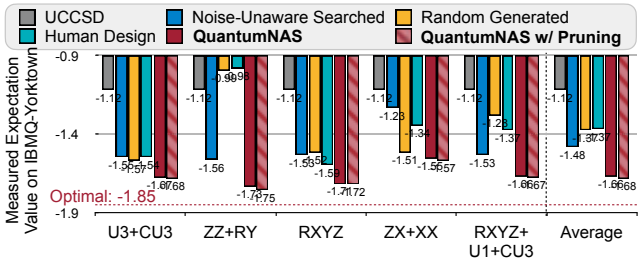

Figure 17. H_2 VQE expectation value in different spaces. QuantumNAS consistently obtains the lowest expectation value.

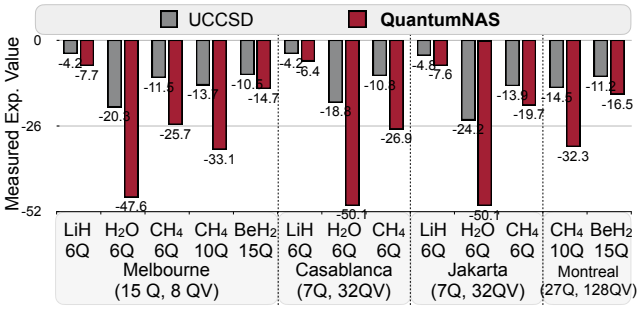

Figure 18. VQE expectation value comparisons on various molecules.

use the IBMQ-Montreal noisy simulator to simulate the full original circuit (7164 gates) and obtains expectation value as -12.86; We also get that of BeH_2-15Q (30851 gates) as -9.81. Despite the much larger circuit, the full circuit results are **worse** than shallower ones because the larger number of gates introduce more significant noise.

C. Performance Analysis

Accuracy Improvement Breakdown. We select five tasks and five design spaces to show the breakdown of accuracy improvements in Figure 19. 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.* As already mentioned in Section I, Figure 3 further shows the #parameters vs. accuracy curves. With more parameters, the accuracy of baseline designs quickly saturates and drops due to gate errors. In contrast, QuantumNAS can mitigate the negative effect of gate errors, and *delays the accuracy peak*, enabling more effective circuit parameters and higher accuracy.

Effect of Front and Restricted Sampling. Figure 20 shows the measured performance of SubCircuits on five tasks in `ZX+XX` and `RXYZ+U1+CU3` spaces. The baseline method is random sampling. Since the front and restricted sampling controls the difference between the consecutive samples, the SuperCircuit training is more stable. Thus it improves the *reliability* of estimated relative performance,

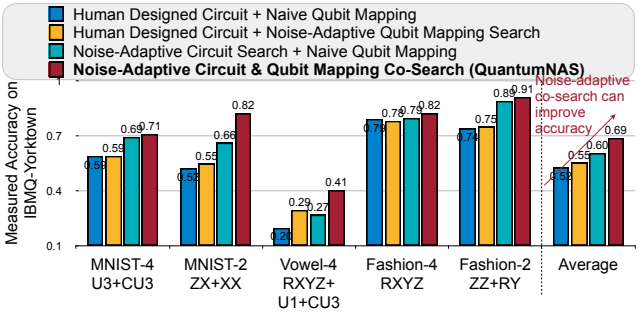

Figure 19. Proposed circuit & qubit mapping co-search improves accuracy.

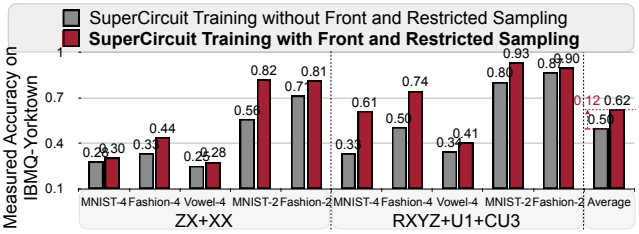

Figure 20. Proposed front and restricted sampling improve final accuracy.

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 21 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. Yorktown has the highest error rate so the performance is worse than others. (2) Comparing Rome ('–') and 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 '–'.

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 VI. 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. This can only be achieved when QuantumNAS has access to a relatively large design space. So we cannot only search shallow circuits.

Random Search vs. Evolutionary Search. Multiple candidate algorithms are applicable for the search stage. We compare the evolutionary with random search in Figure 22.

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 17:47:05 UTC from IEEE Xplore. Restrictions apply.

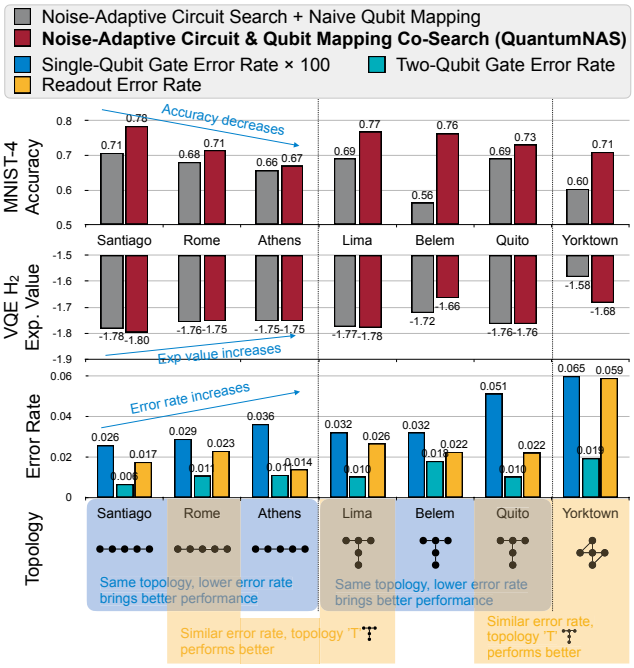

Figure 21. Effect of qubit topology/error rate/qubit mapping on performance.

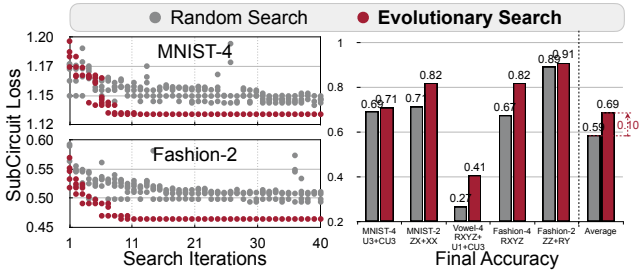

Figure 22. Optimization curves and final accuracy of random/evo search.

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 23 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. In general, circuits with more parameters can afford a larger pruning ratio; those with fewer parameters has a smaller ratio. In Table VII, we further show the acceleration from pruning a circuit with 180 parameters using the Pennylane classical simulation framework.

V. Related Work

Quantum Machine Learning. Quantum machine learning (QML) [6], [35], [36], [49], [54]–[57] 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

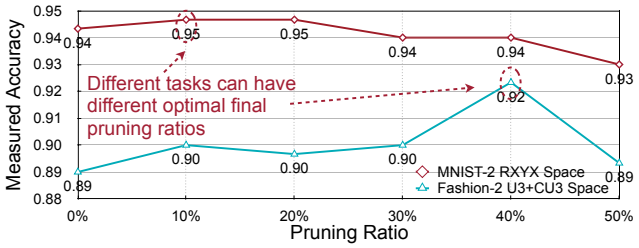

Figure 23. Measured accuracy with different final pruning ratios.

Table VI
Ours shows higher acc. than shallow circuits.

Device	Space	MNIST-4 *D*	MNIST-4 Acc.	Fashion-4 *D*	Fashion-4 Acc.	Vowel-4 *D*	Vowel-4 Acc.	MNIST-2 *D*	MNIST-2 Acc	Fashion-2 *D*	Fashion-2 Acc
Santiago	Shallow	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	Shallow	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	Shallow	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**

[49], data analysis [36], and principal component analysis [35]. 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 [50], quantum convolution [51], and quantum Boltzmann machine [58], etc. Most prior works are exploratory and rely on classical simulation of small quantum systems [50]. Several works also propose to search circuits [59]–[62] 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.

Noise-Adaptive Quantum Compiling. A quantum compiler translates a quantum program written in high-level programming languages to hardware instructions. For NISQ systems [11], 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 [63]–[66], composite pulses [67]–[69], randomized compiling [70], hidden inverses [71], qubit mapping [14]–[16], instruction scheduling [18], [72], and frequency tuning [17], [73], [74]. 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.

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) [3], [4], [38] for quantum simulation of physical systems.

Prior work showed that finding such an ansatz and learning their parameters is challenging. Different classes of

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 17:47:05 UTC from IEEE Xplore. Restrictions apply.

Table VII
PRUNING CAN SPEEDUP CIRCUITS IN PENNYLANE.

Pruning Ratio	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Time (s)	3.46	3.19	2.93	2.63	2.32	2.11	1.80	1.52	1.28	0.95
Speedup	0	7.9%	15.4%	24.0%	33.0%	39.1%	48.0%	55.9%	63.0%	72.4%

ansatz designs have been proposed: (1) *Problem ansatz* [4], [39] is adapted to a target problem. E.g., UCCSD ansatz [52] is a design based on the structures in a quantum system using computational chemistry models. (2) *Hardware ansatz* [3] is adapted to the properties of the computing hardware. Problem ansatz is shown to be typically more resilient to barren plateau than hardware ansatz [41]. In this work, our QuantumNAS aims to find a balanced and robust ansatz design via SuperCircuit-based search.

Quantum Error Mitigation. As the error forms the bottleneck of the quantum area [53]. Various error mitigation methods are proposed. Extrapolation methods [75], [76] perform multiple measurements under different error rates and extrapolate ideal measurement outcomes when no noise. Quasi-probability [75], [77] probabilistically inserts X/Y/Z gates then sum results together to cancel out noise. Ensemble of diverse mapping [78] runs different mappings of the same circuit on different machines and combines the results. Other methods such as quantum subspace expansion [79] and learning-based mitigation [80], [81] are also proposed. QuantumNAS is fundamentally different. Prior work focuses on the low-level numerical correction of trained circuits; QuantumNAS embraces much larger optimization freedom by searching ansatz with intrinsic robustness and performs pruning during training. The existing noise mitigation can be combined with QuantumNAS as they are orthogonal.

VI. CONCLUSION

We propose QuantumNAS, a noise-adaptive co-search framework for robust variational circuit and qubit mapping. We leverage the SuperCircuit-based search to explore an 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 `TorchQuantum`, serving as a convenient infrastructure for future research of variational quantum algorithms.

Outlook. Our results suggest a variety of new avenues for further theoretical and experimental explorations in the domain of variational quantum algorithms. For example: (1) *Machine Learning:* How to deploy a noise-adaptive search strategy to automate the design of a quantum feature map (i.e., embedding data in high-dimensional Hilbert space)? (2) *Optimization:* Can a searched variational ansatz be designed to alleviate the barren plateau issue as seen in many existing shallow ansatz? (3) *Chemistry:* What are the applications amenable to the ansatz search strategy and how to use the searched ansatz to efficiently prepare low-energy eigenstates of a many-body quantum system?

ACKNOWLEDGMENT

We thank National Science Foundation, MIT-IBM Watson AI Lab, and Qualcomm Innovation Fellowship for supporting this research. We acknowledge the use of IBM Quantum services for this work.

APPENDIX

A. Abstract

Our paper introduces a pipeline to search for the most robust architecture (ansatz) and qubit mapping of parameterized quantum circuits with noise information in the loop.

The artifact contains two parts. The first is a general parameterized quantum circuit training framework called `TorchQuantum`. It supports the contribution of our paper on a convenient framework for parameterized quantum circuit research. It can be validated by running the training of an example circuit such as a quantum neural network (QNN) to perform image classification. The second part is the QuantumNAS pipeline. It supports our contribution on a method to find the most robust circuit ansatz and corresponding qubit mapping on the target quantum device. We provide scripts and Google Colab notebooks to reproduce each step of the pipeline, from SuperCircuit training, noise-aware search, SubCircuit training, to pruning, and finally evaluation on real quantum hardware (IBMQ quantum machines). The results can be validated by the better performance of the searched circuits, such as the higher classification accuracy of QNN and the lower expectation value of VQE. The minimal hardware requirements will be Intel CPU, one Nvidia GPU, and remote access to IBMQ quantum machines (using our provided token). The minimal software requirements will be python libraries such as PyTorch, Qiskit, etc.

B. Artifact check-list (meta-information)

- **Algorithm:** SuperCircuit-based quantum circuit search. It contains:
  - SuperCircuit training.
  - SubCircuit and qubit mapping co-search.
  - SubCircuit training.
  - pruning.
- **Program:** Benchmarks:
  - Image classification with quantum neural networks.
  - Vowel recognition with quantum neural networks.
  - Variational quantum eigensolver (VQE) task with parameterized quantum circuits.
- **Compilation:** Qiskit Compiler. Public available. Version 0.31.0.
- **Binary:** PyTorch binary checkpoint files are included.
- **Model:** Automatically searched and human designed quantum neural networks, variational quantum circuits.
- **Data set:**
  - Vowel recognition. Public available: https://www.openml.org/d/58. Approximate size: 100KB.

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 17:47:05 UTC from IEEE Xplore. Restrictions apply.

  - MNIST image classification. Public available: http://yann.lecun.com/exdb/mnist/. Approximate size: 15MB.
  - Fashion MNIST image classification. Public available: https://github.com/zalandoresearch/fashion-mnist. Approximate size: 15MB.

  The dataset can also be automatically accessed through our `TorchQuantum` library with no need for explicit downloading and installing.
- **Run-time environment:**
  - Not OS-specific.
  - Main software dependencies: Python, PyTorch, Qiskit
  - No need for root access.
- **Hardware:**
  - Intel CPUs.
  - Remote access to IBMQ quantum machines. Public available: https://quantum-computing.ibm.com/
  - Nvidia GPUs for faster training.
- **Run-time state:** Sensitive to the noise characteristic of quantum machines. The noise on real quantum machines changes constantly, so the searched noise-aware circuit architecture and the measured performance may change.
- **Execution:** Specific conditions during experiments: the searched circuits need to be evaluated on the real machine right after searching, otherwise the noise characteristics will drift, and performance will be degraded. The runtime depends on the available CPU/GPU machines and the size of the quantum circuits. The approximate runtime on CPU machines for searching and testing one 4-qubit quantum circuit will be around 5 to 10 hours. The approximate runtime on one Nvidia GPU will be about 5 to 10 times faster.
- **Metrics:**
  - Classification task: top1 accuracy, the higher the better.
  - VQE task: expectation value of molecule ground state energy, the lower the better.
- **Output:**
  - Classification task: classification labels.
  - VQE task: expectation value of molecule ground state energy.
- **Experiments:** We provide shell scripts for each step of the QuantumNAS pipeline. You may find the detailed instructions in https://github.com/mit-han-lab/torchquantum/artifact/README.md. The maximum allowable variation of classification accuracy should be smaller than 10%.
- **How much disk space required (approximately)?:** $<$ 10 GB
- **How much time is needed to prepare workflow (approximately)?:** 1 hour.
- **How much time is needed to complete experiments (approximately)?:** 24 hours.
- **Publicly available?:** Yes. https://github.com/mit-han-lab/torchquantum and https://zenodo.org/record/5787244#.YbunmBPMJhE.
- **Code licenses (if publicly available)?:** MIT License.
- **Data licenses (if publicly available)?:**
  - Vowel: Creative Commons Public available.
  - MNIST: Creative Commons Attribution-Share Alike 3.0.
  - Fashion MNIST: MIT License.
- **Workflow framework used?:** PyTorch, Qiskit.
- **Archived (provide DOI)?:** https://github.com/mit-han-lab/torchquantum. We will also provide a Zenodo link and DOI at the end of the evaluation.

C. Description

1) How to access

The artifact is available at the following link:

- https://github.com/mit-han-lab/torchquantum

2) Hardware dependencies

In order to complete the QuantumNAS pipeline experiments in a reasonable amount of time, a CPU with at least 16 GB memory is necessary. Remote access to IBMQ quantum machines is necessary. We provide tokens in the artifact to access the IBMQ machine so no additional effort is required to get access. One GPU machine is highly recommended as it will significantly accelerate the training process.

3) Software dependencies

The artifact is implemented in Python and requires several packages such as PyTorch and Qiskit. The detailed full list of the required packages is in `requirements.txt` file and can be installed automatically when installing the `TorchQuantum` library.

4) Data sets

Machine learning datasets include Vowel recognition, MNIST, and FashionMNIST. VQE benchmarks include various molecules. Details can be found in Table I of the paper.

5) Models

The QNN models are searched SubCircuits with our noise-aware search pipeline. We also have human design, random search, no-unaware searched models as baselines.

D. Installation

You can first download the repo to local machine. Then enter the folder and run:

```
pip install --editable .
```

See our README.md for detailed installation instructions.

E. Experiment workflow

We provide multiple examples to run our artifact.

1) Running the `TorchQuantum` library

We provide the script to construct, train and deploy a simple QNN for the MNIST task in the `artifact/example1` folder. The command is:

```
./example1/1_train_qnn.sh
```

2) Running the QuantumNAS pipeline

(i) Run the SuperCircuit training step:

```
./example2/quantumnas/1_train_supercircuit.sh
```

(ii) Run the SubCircuit and qubit mapping co-search step:

```
./example2/quantumnas/2_search.sh
```

(iii) Run the SubCircuit training step:

```
./example2/quantumnas/3_train_subcircuit.sh
```

(iv) Run the pruning of searched SubCircuit:

```
./example2/quantumnas/4_prune.sh
```

(v) Run the SubCircuit evaluation step:

```
./example2/quantumnas/5_eval.sh
```

Furthermore, we provide scripts to train and evaluate the human baseline designs:

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 17:47:05 UTC from IEEE Xplore. Restrictions apply.

(i) Run the human baseline training step: `./example2/human/1_train.sh`

(ii) Run the human baseline evaluation step: `./example2/human/2_eval.sh`

We also provide other scripts for training QuantumNAS in different datasets, design spaces, and quantum machines and other baselines. Please check the README.md file for the details.

F. Evaluation and expected results

For the evaluation of the functionality of QuantumNAS, the quantum circuit should be successfully constructed and trained. The expected results will be decreasing training loss and increasing training accuracy on the classification task.

For the evaluation of QuantumNAS, the evaluated accuracy on real quantum machines will be obtained after running the scripts. The expected results will be (1) QuantumNAS searched model has better accuracy than baseline models for classification tasks. (2) QuantumNAS searched model achieves lower expectation value of molecule ground state energy for the VQE tasks. This can verify the critical results of the paper in Figure 14, as the main contribution is finding a robust quantum circuit and its qubit mapping for the real quantum devices.

G. Experiment customization

The general `TorchQuantum` can be used to construct, train and deploy different architectures of the quantum circuits. The users can build different customized circuits from scratch.

H. Notes

Since most of the results in our paper are evaluated on real quantum machines provided by IBMQ, some of the machines such as `IBMQ_Yorktown`, `IBMQ_Athens`, `IBMQ_Melbourne` have already retired, so we are only able to reproduce results on the remaining accessible quantum machines.

I. Methodology

Submission, reviewing and badging methodology:

- https://www.acm.org/publications/policies/artifact-review-badging
- http://cTuning.org/ae/submission-20201122.html
- http://cTuning.org/ae/reviewing-20201122.html

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] Y. Cao, J. Romero, J. P. Olson, M. Degroote *et al.*, "Quantum chemistry in the age of quantum computing," *Chemical reviews*, vol. 119, no. 19, pp. 10 856–10 915, 2019.

[6] 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.

[7] 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.

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

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

[10] 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.

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

[12] 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.

[13] 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.

[14] G. Li, Y. Ding, and Y. Xie, "Tackling the qubit mapping problem for nisq-era quantum devices," in *ASPLOS*, 2019, pp. 1001–1014.

[15] 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 *ASPLOS*, 2019, pp. 1015–1029.

[16] 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 *ASPLOS*, 2019, pp. 987–999.

[17] 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 *MICRO*. IEEE, 2020, pp. 201–214.

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 17:47:05 UTC from IEEE Xplore. Restrictions apply.

[18] P. Murali, D. C. McKay, M. Martonosi, and A. Javadi-Abhari, "Software mitigation of crosstalk on noisy intermediate-scale quantum computers," in *ASPLOS*, 2020, pp. 1001–1016.

[19] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *ICML*. PMLR, 2018, pp. 4095–4104.

[20] 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 *ECCV*. Springer, 2020, pp. 544–560.

[21] 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.

[22] H. Wang, Z. Wu, Z. Liu, H. Cai, L. Zhu, C. Gan, and S. Han, "Hat: Hardware-aware transformers for efficient natural language processing," *arXiv preprint arXiv:2005.14187*, 2020.

[23] H. Wang *et al.*, "Efficient algorithms and hardware for natural language processing," Ph.D. dissertation, Massachusetts Institute of Technology, 2020.

[24] H. Tang, Z. Liu, S. Zhao, Y. Lin, J. Lin, H. Wang, and S. Han, "Searching efficient 3d architectures with sparse point-voxel convolution," in *ECCV*. Springer, 2020, pp. 685–702.

[25] 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.

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

[27] 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.

[28] 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.

[29] 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.

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

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

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

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

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

[35] S. Lloyd, M. Mohseni, and P. Rebentrost, "Quantum principal component analysis," *Nature Physics*, vol. 10, no. 9, pp. 631–633, 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] C. Kokail, C. Maier, R. van Bijnen, T. Brydges *et al.*, "Self-verifying variational quantum simulation of lattice models," *Nature*, vol. 569, no. 7756, pp. 355–360, 2019.

[38] J. R. McClean, J. Romero *et al.*, "The theory of variational hybrid quantum-classical algorithms," *New Journal of Physics*, vol. 18, no. 2, p. 023023, 2016.

[39] 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.

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

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

[42] 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.

[43] S. Han, H. Mao *et al.*, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[44] H. Wang, Z. Zhang, and S. Han, "Spatten: Efficient sparse attention architecture with cascade token and head pruning," in *HPCA*. IEEE, 2021, pp. 97–110.

[45] Z. Zhang, H. Wang, S. Han, and W. J. Dally, "Sparch: Efficient architecture for sparse matrix multiplication," in *HPCA*. IEEE, 2020, pp. 261–274.

[46] 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.

[47] S. B. Bravyi *et al.*, "Fermionic quantum computation," *Annals of Physics*, vol. 298, no. 1, pp. 210–226, 2002.

[48] IBM, "Ibm quantum," *IBMQ*, 2021. [Online]. Available: https://quantum-computing.ibm.com/

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

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 17:47:05 UTC from IEEE Xplore. Restrictions apply.

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

[51] M. Henderson, S. Shakya *et al.*, "Quanvolutional neural networks: powering image recognition with quantum circuits," *Quantum Machine Intelligence*, vol. 2, no. 1, pp. 1–9, 2020.

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

[53] T. Patel, A. Potharaju, B. Li, R. B. Roy, and D. Tiwari, "Experimental evaluation of nisq quantum computers: Error measurement, characterization, and implications," in *SC*, ser. SC '20. IEEE Press, 2020.

[54] 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.

[55] Z. Liang, Z. Wang, J. Yang, L. Yang, J. Xiong, Y. Shi, and W. Jiang, "Can noise on qubits be learned in quantum neural network? a case study on quantumflow," *arXiv preprint arXiv:2109.03430*, 2021.

[56] Z. Wang, Z. Liang, S. Zhou, C. Ding, J. Xiong, Y. Shi, and W. Jiang, "Exploration of quantum neural architecture by mixing quantum neuron designs," *arXiv preprint arXiv:2109.03806*, 2021.

[57] H. Wang, J. Gu, Y. Ding, Z. Li, F. T. Chong, D. Z. Pan, and S. Han, "Roqnn: Noise-aware training for robust quantum neural networks," *arXiv preprint arXiv:2110.11331*, 2021.

[58] 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.

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

[60] ——, "Neural predictor based quantum architecture search," *arXiv preprint arXiv:2103.06524*, 2021.

[61] 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.

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

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

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

[65] 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.

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

[67] 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.

[68] 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.

[69] 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.

[70] 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.

[71] B. Zhang, S. Majumder, P. H. Leung, S. Crain, Y. Wang *et al.*, "Hidden inverses: Coherent error cancellation at the circuit level," *arXiv preprint arXiv:2104.01119*, 2021.

[72] X.-C. Wu, D. M. Debroy, Y. Ding *et al.*, "Tilt: Achieving higher fidelity on a trapped-ion linear-tape quantum computing architecture," *arXiv preprint arXiv:2010.15876*, 2020.

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

[74] 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.

[75] K. Temme, S. Bravyi, and J. M. Gambetta, "Error mitigation for short-depth quantum circuits," *Physical review letters*, vol. 119, no. 18, p. 180509, 2017.

[76] Y. Li and S. C. Benjamin, "Efficient variational quantum simulator incorporating active error minimization," *Physical Review X*, vol. 7, no. 2, p. 021050, 2017.

[77] M. Huo and Y. Li, "Self-consistent tomography of temporally correlated errors," *Communications in Theoretical Physics*, vol. 73, no. 7, p. 075101, 2021.

[78] T. Patel and D. Tiwari, "Veritas: Accurately estimating the correct output on noisy intermediate-scale quantum computers," in *SC*, ser. SC '20. IEEE Press, 2020.

[79] J. R. McClean, M. E. Kimchi-Schwartz, J. Carter, and W. A. De Jong, "Hybrid quantum-classical hierarchy for mitigation of decoherence and determination of excited states," *Physical Review A*, vol. 95, no. 4, p. 042308, 2017.

[80] A. Strikis, D. Qin, Y. Chen, S. C. Benjamin, and Y. Li, "Learning-based quantum error mitigation," *arXiv preprint arXiv:2005.07601*, 2020.

[81] P. Czarnik, A. Arrasmith, P. J. Coles, and L. Cincio, "Error mitigation with clifford quantum-circuit data," *arXiv preprint arXiv:2005.10189*, 2020.

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 17:47:05 UTC from IEEE Xplore. Restrictions apply.