Parameter-parallel distributed variational quantum algorithm

Yun-Fei Niu[1◦], **Shuo Zhang**[1⋆◦], **Chen Ding**[1], **Wan-Su Bao**[1†] **and He-Liang Huang**[1,2,3,4‡]

1 Henan Key Laboratory of Quantum Information and Cryptography, Zhengzhou, Henan 450000, China
2 Hefei National Research Center for Physical Sciences at the Microscale and School of Physical Sciences, University of Science and Technology of China, Hefei 230026, China
3 Shanghai Research Center for Quantum Science and CAS Center for Excellence in Quantum Information and Quantum Physics, University of Science and Technology of China, Shanghai 201315, China
4 Hefei National Laboratory, University of Science and Technology of China, Hefei 230088, China

⋆ shuoshuo19851115@163.com , † bws@qiclab.cn ,
‡ quanhhl@ustc.edu.cn

Abstract

Variational quantum algorithms (VQAs) have emerged as a promising near-term technique to explore practical quantum advantage on noisy intermediate-scale quantum (NISQ) devices. However, the inefficient parameter training process due to the incompatibility with backpropagation and the cost of a large number of measurements, posing a great challenge to the large-scale development of VQAs. Here, we propose a parameter-parallel distributed variational quantum algorithm (PPD-VQA), to accelerate the training process by parameter-parallel training with multiple quantum processors. To maintain the high performance of PPD-VQA in the realistic noise scenarios, a alternate training strategy is proposed to alleviate the acceleration attenuation caused by noise differences among multiple quantum processors, which is an unavoidable common problem of distributed VQA. Besides, the gradient compression is also employed to overcome the potential communication bottlenecks. The achieved results suggest that the PPD-VQA could provide a practical solution for coordinating multiple quantum processors to handle large-scale real-word applications.

Received 02-08-2022
Accepted 30-03-2023
Published 26-05-2023
doi:10.21468/SciPostPhys.14.5.132

Contents

◦ These authors contributed equally to the development of this work.

1 Introduction

Quantum computing holds the promise of solving certain problems that intractable for classical computers, such as factoring large numbers [1–3], database search [4, 5], solving linear systems of equations [6–8]. However, a universal fault-tolerant quantum computer that can solve efficiently the above problems would require millions of qubits with low error rates [9, 10], which is still a long way from current techniques and may take decades. Thus, we will be in the noisy intermediate-scale quantum (NISQ) era for a long time [11–16]. Variational quantum algorithms (VQAs) leverage a quantum device to minimize a specific cost function [17, 18], by employing a classical optimizer (e.g., Adam optimizer [19]) to train parameter quantum circuits (PQCs). Such algorithms were shown to have natural noise resilience [20] and even benefit from noise, making it particularly suitable for near-term quantum devices, and thus be considered the most promising path to quantum advantage on practical problems in NISQ era [18]. Previous studies have exhibited the application of VQAs on a variety of problems, including classification task [21–24] and generative task [25–27], combinatorial optimization [28–32], quantum many-body problem [33] and quantum chemistry [34–39].

The training process of VQAs is actually not very efficient compared to the classical neural network, due to the following two main reasons: 1) The quantum state of the intermediate process of the quantum circuit cannot be stored, making VQAs impossible to use the back-propagation to update the parameters as efficiently as the classical neural network; 2) A large number of measurements is required for the result readout of the quantum circuit, which is time-consuming. Therefore, the training of VQAs will face significant challenges, as the amount of data and trainable parameters increases.

To address the above issue, a distributed VQA based on data-parallel has been proposed by Du $et.\ al.$ to accelerate the training of VQA [40]. In this work, a parameter-parallel distributed variational quantum algorithm (PPD-VQA) is proposed to further accelerate the training process by parameter-parallel training with multiple quantum processors. Although the idea of parallel training is not difficult to come up with, including data-parallel or parameter-parallel, it is worth investigating whether the approach works in the realistic scenario that the local quantum nodes will inevitably be affected by quantum noise, and the noise intensity of each node is different. We first proof the convergence of the PPD-VQA, even if each local node has different quantum noise. Further, we design an alternate training strategy to alleviate the acceleration attenuation caused by excessive noise differences among multiple quantum

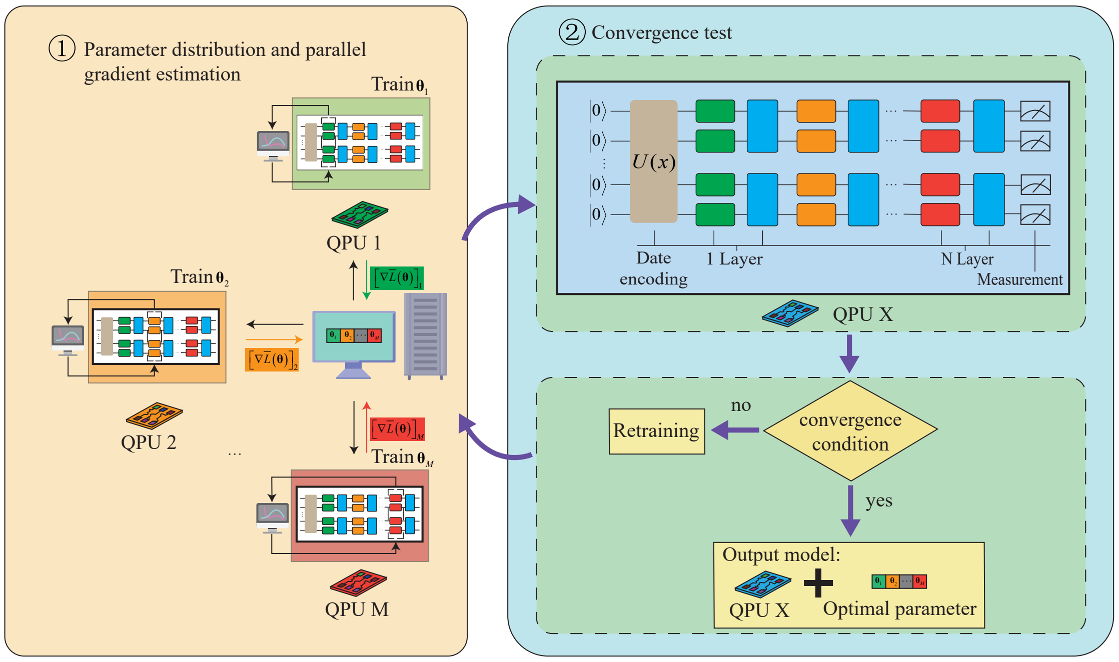

Figure 1: **Schematic diagram of PPD-VQA.** The diagram illustrates two main steps of the PPD-VQA workflow. Firstly, the central parameter server allocates the trainable parameters to M local nodes, consisting of a QPU and a classic computer, for parallel training. Each local node only trains a part of the trainable parameters, and synchronizes the gradient information to the central parameter server. Secondly, a local node, named QPU X, is selected to verify that the convergence condition is met. If it does not converge, repeat Steps 1 and 2, otherwise, output the optimal parameters and selected local node as the trained model.

processors, and adopt the gradient compression to cut a large amount of communication bandwidth, to enhance the practicality and scalability of PPD-VQA.

2 PPD-VQA

The conventional VQAs employ PQCs and update their parameters $\boldsymbol{\theta}$ via a classical optimization training procedure, to find the global minimum of the given loss functions L. Usually, in the training procedure, the gradients of each parameter is evaluated by the parameter-shift rule [41, 42]. The PPD-VQA leverages the fact that the partial derivatives of the observable with respect to each parameter are genuinely independent of one another at each iteration to accelerate the training of conventional VQA, by parallelizing the gradient estimation across multiple quantum processing unit (QPU) nodes. Conceptually, a classical central parameter server and M local nodes constitute the framework of PPD-VQA, where each local node consists of a QPU and a classical optimizer. As shown in Fig. 1 and Algorithm 1, at each iteration, the central parameter server divides the trainable parameters $\boldsymbol{\theta}$ into M parts, each of the M local nodes is tasked with computing the gradient of the parameters for a given component. Then, the complete gradient information is obtained through information sharing between local nodes and central parameter sever, which is used to update the trainable parameters as the initial parameters of next iteration. This process is repeated until the optimal parameters are found. The specific process can be divided into the following two steps:

Step 1: Parameter distribution and parallel gradient estimation. At the beginning of

t-th iteration, the classical central server distribute the complete parameter $\boldsymbol{\theta}^{(t)}$ of PQC to each local node as the initial parameters, as well as instructions on which parameters the i-th local node is assigned for training. The default instruction is to divide the trainable parameters $\boldsymbol{\theta}^{(t)}$ into M equal parts

$$\boldsymbol{\theta}^{(t)} = \left[\boldsymbol{\theta}_1^{(t)}, \cdots, \boldsymbol{\theta}_M^{(t)}\right], \quad \boldsymbol{\theta}_i^{(t)} = \left(\theta_{i,1}^{(t)}, \theta_{i,2}^{(t)}, \cdots, \theta_{i,n}^{(t)}\right), \quad n = \frac{d}{M},$$

and the i-th local node is responsible for estimating the corresponding component of gradient $\left[\nabla \bar{L}(\boldsymbol{\theta})\right]_i$. After the training on each node, the local node synchronizes $\left\{\left[\nabla \bar{L}(\boldsymbol{\theta})\right]_i\right\}_{i=1}^{M}$ to the central parameter server, and the central parameter server combines the information from each local node into a completed gradient $\nabla \bar{L}(\boldsymbol{\theta}^{(t)})$ used to update $\boldsymbol{\theta}^{(t)}$ to $\boldsymbol{\theta}^{(t+1)}$.

Step 2: Convergence test. Choose a fixed local node from the M local nodes, and substitute the parameters $\boldsymbol{\theta}^{(t+1)}$ into this local node. After that, employ this model to the test dataset to to determine whether the convergence condition has been met. The setting of the convergence condition depends on the machine learning task. For example, for the classification task, the convergence condition might be set as a certain classification accuracy threshold. The convergence test refers to selecting a QPU and getting the accuracy score on the classification task. If the convergence condition has not been met, return to Step 1 for the subsequent training iteration; otherwise, output the final parameters and chosen local node as the trained model and terminate the training procedure. By implementing the convergence test, we can monitor the performance of the trained PQC on the chosen QPU to ensure that the final PQC will perform well on the chosen QPU.

The core idea of PPD-VQA is simple and natural. However, distributed quantum machine learning faces different challenges than its classical counterpart, the main one being that the quantum processors on different local nodes are not identical, due to the inevitable quantum noise. In general, the error rate ε_i of each qubit on a quantum processor is different, and we let the average error rate of the processor be $\bar{\varepsilon} = \frac{1}{N}\sum_i \varepsilon_i$, where N is number of qubits. Thus, the non-uniformity mentioned above manifests itself in two ways: 1) The average error rates of each quantum processors are different. For example, some processors have lower noise and some have higher noise; 2) Even if the average error rate of each quantum processor is the same, the error rate of each qubit in these processors is unlikely to be consistent. In such a realistic scenario, it remains to be verified whether the parameter-parallel training is still effective, and whether the convergence conditions can be achieved. This important issue is directly related to the practical utility of our scheme and will be discussed in the next section.

3 Performance Analysis and error mitigation strategy in the realistic Noise Scenario

Gradient represents the optimization direction during the training procedure of VQA, which plays an decisive role in the process of finding the global minima of loss function. Thus, by examining the gradient, we analyze how noise affects convergence of PPD-VQA in the realistic scenario that noise varies for each quantum processor. Furthermore, we will propose a strategy to mitigate the negative consequences that maybe caused by this realistic scenario.

3.1 Convergence and acceleration

We apply the "worst-case" noise channel–the depolarizing channels [43] for the following research. According to the Lemma 6 in Ref. [44] all noisy channels $\varepsilon(\cdot)$, which are separately

Algorithm 1 The pseudocode of PPD-VQA.

```
Require: $\boldsymbol{\theta} \in [0, 2\pi)^d$: the parameters of ansatz; $L$: loss function; $M$: the number of local
    nodes, and we donate $M_i$ as the $i$-th local node;
Ensure: optimal parameters $\boldsymbol{\theta}*$
 1: while convergence condition is not satisfied do
 2:     The central parameter server divides the parameter $\boldsymbol{\theta}$ into $M$
        parts and allocates $\boldsymbol{\theta}$ to $M$ local nodes
 3:     for Local nodes $M_i$, $\forall i \in \{1, \cdots, M\}$ in parallel do
 4:         Calculate the estimated gradient component $\left[\nabla \bar{L}(\boldsymbol{\theta})\right]_i$
 5:     end for
 6:     Synchronize $\nabla \bar{L}(\boldsymbol{\theta})$ by merging $\{\left[\nabla \bar{L}(\boldsymbol{\theta})\right]_i\}_{i=1}^M$
 7:     Update $\boldsymbol{\theta}$ with a classical optimizer, such as ADAM
 8:     Choose a local node from $\{M\}_{i=1}^M$ for convergence test
 9:     if convergence condition is satisfied then
10:         break
11:     end if
12: end while
```

applied to each layer of the ansatz, can be merged together and represented by a new depolarizing channel acting on the whole ansatz, i.e.,

$$\tilde{\varepsilon}(\rho) = (1 - \tilde{p})\rho + \tilde{p}\frac{\mathbb{I}}{2^n}, \tag{1}$$

where $\tilde{p} = 1 - (1 - p)^N$, p is the depolarizing probability in $\varepsilon(\cdot)$, and N refers to depth of ansatz. To facilitate understanding, we denote $\tilde{p}_i$ as the noise level of the i-th QPU. Obviously, the depolarizing noise turns the quantum state into a maximally mixed state with a certain probability, which could make the gradients obtained by parameter-shift-rule in the experiment deviate from that of the ideal environment without noise.

We firstly simplify some notations and introduce basic concepts in optimization theory for ease of subsequent discussion. Denote $D = \{x_k, y_k\}$ as the training dataset, where $x_k \in R^{2^n}$ and $y_k \in [0, 1]$ refer to example and the corresponding label respectively. We define L as the loss function, $\nabla L(\boldsymbol{\theta}^{(t)})$ as the gradient of loss function L. Here we employ the mean square error (MSE) loss function, *i.e.*,

$$L = \frac{1}{2N_D}\sum_k (\hat{y}_k - y_k)^2 + \frac{\lambda}{2} \parallel \theta^{(t)} \parallel^2, \tag{2}$$

where the predicted label $\hat{y}_k{}^{(t)} = Tr[U(\boldsymbol{\theta}^{(t)})\rho_k U^\dagger(\boldsymbol{\theta}^{(t)})O]$ is defined by the expected output of noiseless PQC $U(\boldsymbol{\theta}^{(t)})$ with the observable O and input state ρ_k, N_D is the number of the data, and $\lambda \geq 0$ is the regularizer coefficient.

According to parameter-shift-rule, the j-th component of the analytical gradient evaluated on i-th QPU $[\nabla L(\boldsymbol{\theta}^{(t)})]_{i,j}$ at t-th iteration satisfies

$$[\nabla L(\boldsymbol{\theta}^{(t)})]_{i,j} = \frac{1}{N_D}\left[\sum_k (\hat{y}_k{}^{(t)} - y_k)\frac{\hat{y}_k{}^{(t,+j)} - \hat{y}_k{}^{(t,-j)}}{2} + \lambda\theta_{i,j}^{(t)}\right].$$

Here $\hat{y}_k{}^{(t,\pm j)} = Tr[U(\boldsymbol{\theta}^{(t)} \pm \frac{\pi}{2}\boldsymbol{e}_j)\rho_k U^\dagger(\boldsymbol{\theta}^{(t)} \pm \frac{\pi}{2}\boldsymbol{e}_j)O]$ denote the output of PQC with shifted parameter $\boldsymbol{\theta}^{(t)} \pm \frac{\pi}{2}\boldsymbol{e}_j$ where $\boldsymbol{e}_j$ is the unit vector with its j-th component equals to one. Thus, for each data, the local node should implement $1 + 2d/M$ quantum circuits for the gradient estimation, where d/M is the number of parameter in each local node.

Now we quantify the convergence of PPD-VQA with multiple local nodes that have different performance, by using the following utility metric [44]:

$$R_1(\boldsymbol{\theta}^{(T)}) = \mathbb{E}[\|\nabla L(\boldsymbol{\theta}^{(T)})\|^2], \tag{3}$$

where T is the number of iterations and the expectation $\mathbb{E}[\bullet]$ is taken over the random variables associated with depolarizing noise. This metric evaluates how far the result is away from the stationary point. The upper bounds of $R_1(\boldsymbol{\theta}^{(T)})$ when implementing PPD-VQA with multiple non-identical processors are summarized in the following theorem.

Theorem 1 *Suppose that M noisy local nodes of PPD-VQA have different depolarizing noise with depolarizing probability $\{\tilde{p}_i\}_{i=1}^M$, the metric $R_1(\boldsymbol{\theta}^{(T)})$ has following upper bound*

$$R_1 \leq \frac{1+9\pi^2\lambda d}{2T(1-\tilde{p}_{max})^2} + \frac{2G+d}{(1-\tilde{p}_{max})^2}(2-\tilde{p}_{max})\tilde{p}_{max}(1+10\lambda)^2 + \frac{2dK+d}{2N_D K^2}\frac{1}{(1-\tilde{p}_{max})^2},$$

where loss function L is S-smooth with $S=(3/2+\lambda)d^2$, G-Lipschitz with $G=d(1+3\pi\lambda)$, and $\tilde{p}_{max} = \max\{\tilde{p}_i\}_{i=1}^M$.

The proof of Theorem 1 is essentially similar with conventional VQA, for both of them acquire the complete gradient information only once in an iteration. Therefore, one can obtain the upper bound of $R_1(\boldsymbol{\theta}^{(T)})$ of PPD-VQA in noise scenario by following a similar proof procedure of Theorem 1 in Ref. [44]. We briefly sketch our proof as follows.

The first step is to establish the relation between true gradient component $\left[\nabla L(\boldsymbol{\theta}^{(t)})\right]_{i,j}$ (unbiased) and that in the estimated gradient $\left[\nabla \bar{L}(\boldsymbol{\theta}^{(t)})\right]_{i,j}$ (biased) that is evaluated from QPU i (see Appendix A for the detailed derivation),

$$[\nabla \bar{L}(\boldsymbol{\theta}^{(t)})]_{i,j} = (1-\tilde{p}_i)^2[\nabla L(\boldsymbol{\theta}^{(t)})]_{i,j} + C_{i,j}^{(t)} + \varsigma_{i,j}^{(t)}, \tag{4}$$

where $C_{i,j}^{(t)}$ originates from the depolarizing noise, and $\varsigma_{i,j}^{(t)}$ is a item related to random variables, which has zero mean.

Then one can further utilize the S-smooth and G-Lipschitz of the L to calculate the loss difference, i.e.,

$$L(\boldsymbol{\theta}^{(t+1)}) - L(\boldsymbol{\theta}^{(t)}) \leq \langle \nabla \bar{L}(\boldsymbol{\theta}^{(t)}), \boldsymbol{\theta}^{(t+1)} - \boldsymbol{\theta}^{(t)}\rangle + \frac{S}{2}\|\boldsymbol{\theta}^{(t+1)} - \boldsymbol{\theta}^{(t)}\|_2^2. \tag{5}$$

Substitute Eq.(4) and $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta\nabla\bar{L}(\boldsymbol{\theta}^{(t)})$ (we set the learning rate $\eta = 1/S$) into Eq.(5) and take the expectation over the random variable $\varsigma_{i,j}^{(t)}$, one have

$$\begin{aligned}\mathbb{E}_{\varsigma_{i,j}^{(t)}}[L(\boldsymbol{\theta}^{(t+1)}) - L(\boldsymbol{\theta}^{(t)})] \leq & \sum_{i,j}\Bigg[-\frac{1}{2S}(1-\tilde{p}_i)^2\left(\left[\nabla L(\boldsymbol{\theta}^{(t)})\right]_{i,j}\right)^2 \\ & + \frac{2G/d+1}{2S}(2-\tilde{p}_i)\tilde{p}_i(1+10\lambda)^2\Bigg] + \frac{2dK+d}{4SN_DK^2}.\end{aligned} \tag{6}$$

Note that $-\sum_{i,j}(1-\tilde{p}_i)^2\left(\left[\nabla L(\boldsymbol{\theta}^{(t)})\right]_{i,j}\right)^2 \leq -(1-\tilde{p}_{max})^2\|\nabla L(\boldsymbol{\theta}^{(t)})\|^2$, and $(2-\tilde{p}_i)\tilde{p}_i \leq (2-\tilde{p}_{max})\tilde{p}_{max}$, we obtain

$$\begin{aligned}\|\nabla L(\boldsymbol{\theta}^{(t)})\|^2 \leq & 2S\frac{L(\boldsymbol{\theta}^{(t)}) - \mathbb{E}_{\varsigma_{i,j}^{(t)}}L(\boldsymbol{\theta}^{(t+1)})}{(1-\tilde{p}_{max})^2} \\ & + \frac{2G+d}{(1-\tilde{p}_{max})^2}(2-\tilde{p}_{max})\tilde{p}_{max}(1+10\lambda)^2 + \frac{2dK+d}{4SN_DK^2}\frac{1}{(1-\tilde{p}_{max})^2}.\end{aligned} \tag{7}$$

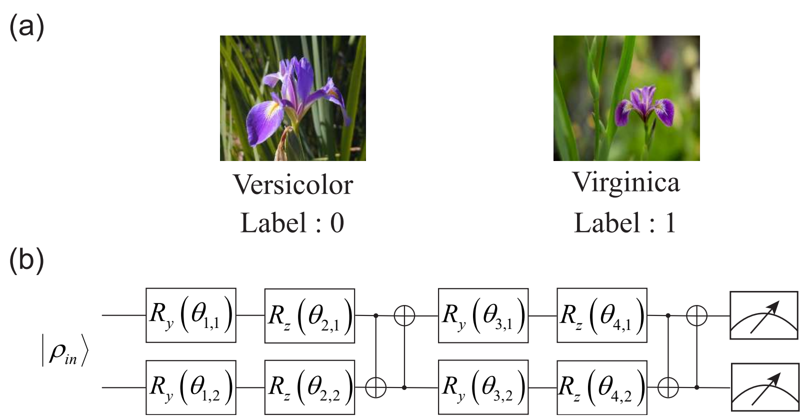

Figure 2: **Classification task on Iris dataset and the ansatz in numerical simulation.** (a) A visualization of training examples sampled from Iris dataset. We choose the data of iris versicolor (label 0) and iris virginica (label 1) for binary classification. (b) Ansatz of PPD-VQA for the classification.

Finally, by summing over $t = 0, 1, \cdots, T$, the upper bound of $R_1(\boldsymbol{\theta}^{(T)})$ is achieved.

From Theorem 1 above and Theorem 1 in Ref. [44], we can observe that the convergence rate between conventional VQA and PPD-VQA is similar, i.e., both of them scale with $O(1/\sqrt{T})$ [44], since the second term and the third term are constant in above inequality when $\{\tilde{p}\}_{i=1}^{M}$ is fixed. The similar convergence rate guarantees that PPD-VQA promises a intuitive linear runtime speedup of the computation of the gradient with respect to the increased number of local nodes M.

Next, we perform numerical experiment to study the performance of PPD-VQA in the realistic noise scenario.

In our simulations, we apply PPD-VQA to the binary classification task, by employing the Iris dataset and ansatz shown in Fig. 2. We choose 100 examples from Iris dataset with 50 versicolors (label 0) and 50 virgunicas (label 1), where 75% examples are randomly selected as the training set and the remaining 25% as the test set. We encode the classical example x_k in Iris dataset into the a two-qubit quantum state ρ_k by amplitude encoding, *i.e.*,

$$| \psi_k \rangle = \frac{\alpha_{k1} \,|\, 00\rangle + \alpha_{k2} \,|\, 01\rangle + \alpha_{k3} \,|\, 10\rangle + \alpha_{k4} \,|\, 11\rangle}{\sqrt{|\,\alpha_{k1}\,|^2 + |\,\alpha_{k2}\,|^2 + |\,\alpha_{k3}\,|^2 + |\,\alpha_{k4}\,|^2}}\,, \tag{8}$$

where $(\alpha_{k1}, \alpha_{k2}, \alpha_{k3}, \alpha_{k4})$ is the feature of x_k. Then a hardware-efficient PQC with 8 trainable single-qubit gates, as shown in Fig. 2(b), is employed for the training. After the quantum state ρ_k has evolved, we perform K global measurements on the final quantum state. We then derive the expectation of observable and map it to $[0,1]$ by linear mapping, where the observable is set as $(Z^{\otimes 2} + I)/2$.

We implement the task using the PPD-VQA with M =1 (conventional VQA), 2, 4, 8 local nodes, respectively. For each type of PPD-VQA, we also set different noise parameters separately. Specifically, for each node the PPD-VQA, the depolarizing probability p_i for single-qubit gate is set by sampling from a Gaussian distribution i.e., $p_i \sim N(\mu, \sigma^2)$, where the mean μ varies from 0.01 to 0.05 with step 0.02 and $\sigma = \mu/9$. The depolarizing probability of two-qubit gate is set as $4p_i$ refer to the performance of SOTA quantum processor *Zuchongzhi* [15]. Each local node's noise will be somewhat different as a result of such random sampling. A total of 100 independent experiments were run for each setting, and in each experiment, the measurement shots is set to 8192, batchsize is set to 5, and the convergence condition is that the classification accuracy on the training set exceeds 96%.

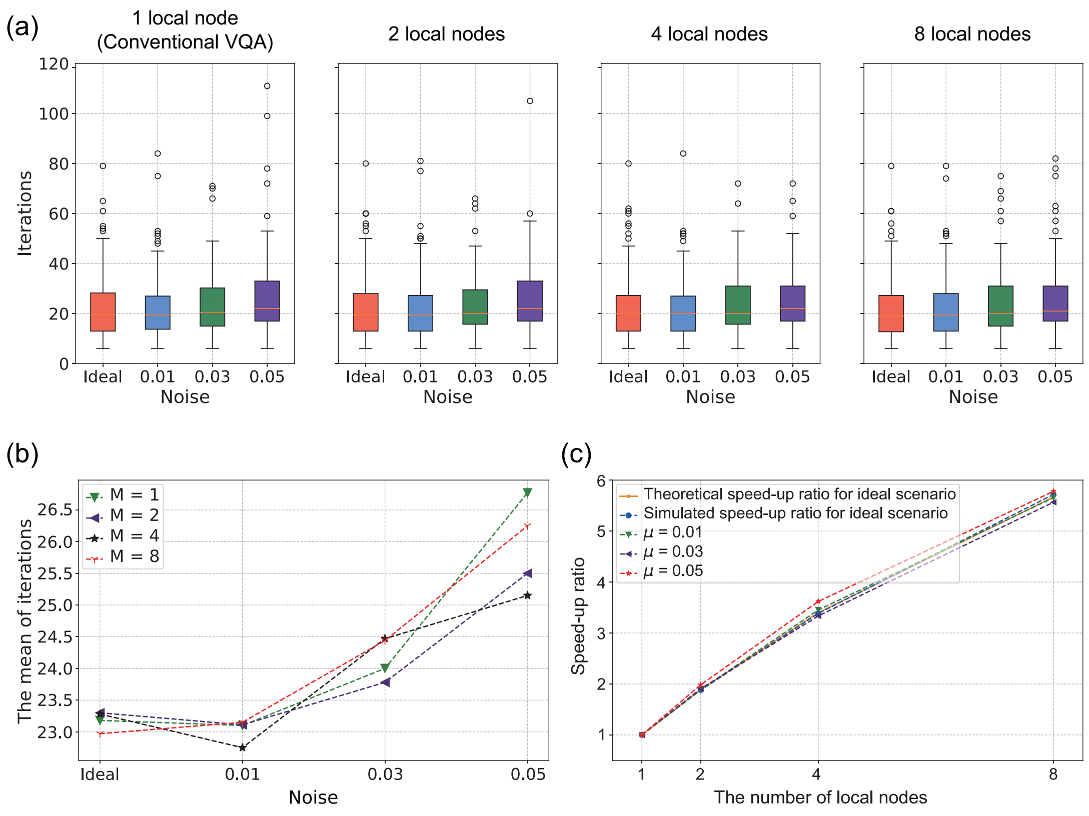

Figure 3: **Simulation results of PPD-VQA with M local nodes under noise scenario for Iris dataset classification.** (a) Boxplots count the iterations of PPD-VQA with M local nodes, where $M = 1,2,4,8$ from left to right, when achieving a predefined training accuracy. The depolarizing probability p_i for single-qubit gate is set by sampling from a Gaussian distribution i.e., $p_i \sim N(\mu,\sigma^2)$, where the mean $\mu = 0$ (ideal case), $0.01, 0.03, 0.05$, and $\sigma = \mu/9$. The depolarizing probability can be converted to Pauli errors e_p for single-qubit gate by using $e_p = \frac{3}{4}p_i$ [13]. As a reference, the single-qubit gate Pauli error of the *Zuchongzhi* processor is 0.14% [15]. The middle line of the boxes, which is the median of the data, represents the average of the number of iterations. The upper and lower limits of the box, which are the upper and lower quartiles of that, respectively, which means that the box contains 50% examples. Above and below the box, there are other lines each representing the maximum and minimum values and the circles represent outliers. (b) Scaling behavior of the mean of the iterations in (a) for increasing noise (μ). The results of PPD-VQA with $M = 1,2,4,8$ local nodes are shown. (c) Scaling behavior of speed-up ratio in clock-time for increasing number of local nodes M. The results of different depolarizing probabilities are shown.

As shown in Fig. 3(a, b), for both conventional VQA and PPD-VQAs with 2, 4, and 8 local nodes, the number of iterations required to achieve a preset training accuracy increases with the mean of noise μ, and the PPD-VQA with multiple local nodes has a similar convergence speed as conventional VQA (see Fig. 3(b)), which is consistent with Theorem 1. Meanwhile, the number of iterations is not sensitive to changes of the number of local nodes, which may be caused by the proximity of noise levels of M local nodes.

We further introduce a metric, i.e. $R_S = T_1/T_m$ to evaluate the speed-up ratio of the PPD-VQA with $M = m > 1$ local nodes compared to the conventional VQA with just $M = 1$ local node, where T_1 and T_m are the time consuming of conventional VQA and PPD-VQA from the start of training to meeting the convergence conditions, respectively. It should be noted that T_1 and T_m are obtained in the same noise scenario. Assuming that the time consumption of implementing each quantum circuit is the same (since the number of measurements is the same, and only the rotation angle of the single-qubit gate will be changed each time the circuit is executed), the formula of the speedup ratio R_S can be further rewritten as

$$R_S = \frac{(1+2d)\times N_D \times N_I^1}{(1+\frac{2d}{M})\times N_D \times N_I^M} = \frac{(1+2d)\times N_I^1}{(1+\frac{2d}{M})\times N_I^M}\,, \tag{9}$$

where d is the number of parameters, N_I^1 and N_I^M are the total number of iterations for the conventional VQA and PPD-VQA, respectively. In the ideal scenario of noiseless, $N_I^1 = N_I^M$, thus $R_S = \frac{1+2d}{1+\frac{2d}{M}}$ in ideal scenario.

Figure 3(c) shows that speed-up ratio for the PPD-VQA with 1 (conventional VQA), 2, 4, 8 local nodes under under a variety of noise scenarios. No matter how the mean of noise μ changes, the speed-up ratio of PPD-VQA is almost only related to the number of local nodes and is extremely close to the ideal case. The reason that the speed-up ratio is almost independent of noise is that the training speed of 1 nodes and M nodes slows down at the same time as the noise grows. This result strongly supports that PPD-VQA can achieve a very good acceleration in realistic scenarios.

3.2 Alternate training strategy for mitigating the negative effects of large noise differences between different local nodes

In the previous subsection, the difference in the noise of the quantum processors of each node is not particularly large, because the noise is set by sampling from a Gaussian distribution $N(\mu,\sigma^2)$, where $\sigma = \mu/9$. In this subsection, we will study the performance of PPD-VQA in cases where the noise difference is more pronounced.

We first monitor the performance of PPD-VQA when the noise difference of different local nodes changes from small to large. To quantify the noise differences of local nodes, we introduce a metric, named $Differ$, which is defined as

$$Differ = D_{KL}(P(p)\,\|\,P_{\text{Uniform}})\,,$$

where D_{KL} is Kullback-Leibler (K-L) divergence [45], P_{Uniform} refers to the uniform distribution, $P(p)$ is the normalized distribution of depolarization probability of each local node, where $P(p)_k = p_k/\sum_{i=1}^{M} p_i$, and p_k is the depolarization probability of the k-th local node. With this metric, a noise setting with a resulting distribution that corresponds to a higher K-L divergence with respect to uniform distribution would mean greater noise variance between local nodes. Besides, we set another constraint that the mean of $\{p_i\}_{i=1}^{M}$ is 0.04. For each PPD-VQA with $M \in [1,2,4,8]$ local nodes, $Differ$ varies from 0 to 0.625, we generate 10 instances of noise setting for each $Differ$, and for each instance 50 experiments with different initial parameters are implemented. As shown in Fig. 4, the speed-up ratio tends to become smaller as the $Differ$

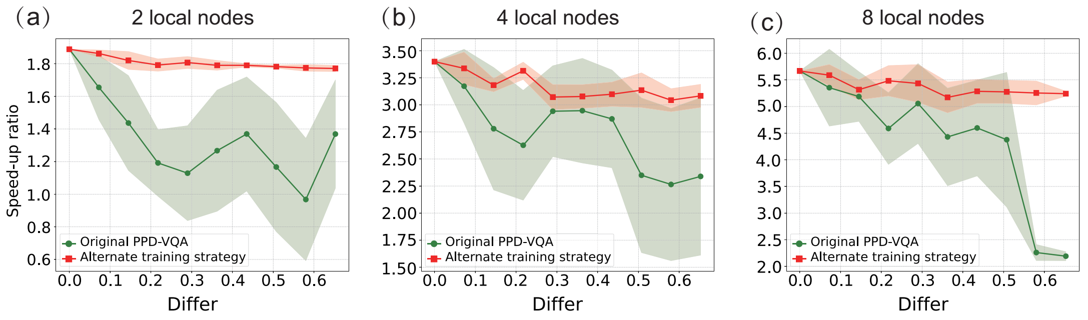

Figure 4: **Simulation results of PPD-VQA with M local nodes in cases where the noise difference is more pronounced.** (a) The average speed-up ratio as a function of $Differ$ (a metric for quantifying the noise differences of local nodes) for the PPD-VQA with $M = 2$(a), 4(b), 8(c) local nodes. 100 independent experiments are implemented for each setting. The green and red solid lines present the results for original PPD-VQA and PPD-VQA with alternate training strategy, respectively. The shaded area represents statistical error caused by 100 independent experiments. It is obvious that the red curves have noticeable larger values and smaller variances than the green curves in all three cases.

increases, indicating that the advantage of PPD-VQA in terms of speedup is diminished in extreme cases where the noise difference between local nodes is significant.

To suppress acceleration decay of PPD-VQA caused by excessive noise difference between local nodes, we propose a simple but effective approach named as alternate training strategy, whose core idea is decoupling the trainable parameter groups and corresponding quantum processors. The process of this alternate training strategy is as follows: Suppose that at the first iteration, the i-th local node is scheduled to train the parameters $\boldsymbol{\theta_i}$. We denote this process as $\{\boldsymbol{\theta_i} : \mathrm{QPU}_i\}_{i=1}^{M}$. Then in the next iteration, The corresponding relationship between trainable parameters and local nodes becomes $\{\boldsymbol{\theta_M} : \mathrm{QPU}_1\} \cup \{\boldsymbol{\theta_i} : \mathrm{QPU}_{i+1}\}_{i=1}^{M-1}$, that is, we perform a cyclic shift on the correspondence between the trainable parameters and local nodes. The alternate training strategy is repeated with the training process, which makes each parameter group $\boldsymbol{\theta_i}$ be trained in turn by all quantum processors throughout the whole training process.

The numerical simulation results of PPD-VQA with alternate training strategy are shown in Fig. 4. An immediate observation is that when the noise difference between local nodes increases, PPD-VQA performance degrades relatively little thanks to the alternate training strategy. Besides, the performance of PPD-VQA becomes more stable as the variance of the mean of different experiments is significantly smaller. These two benefits suggest that this strategy can be effectively employed for mitigating the negative effects of large noise differences between different local nodes.

4 Gradient compression

Another challenge of distributed machine learning is the large amount of communication bandwidth for gradient exchange [46]. With the development of quantum computing hardware, this problem may also arise in large-scale distributed quantum machine learning. To overcome this potential problem, we adopt the technique of gradient compression [47] widely used in the classical community to PPD-VQA, to reduce the communication bandwidth for distributed training. The pseudocode of PPD-VQA with gradient compression for local node i in PPD-VQA is as follows.

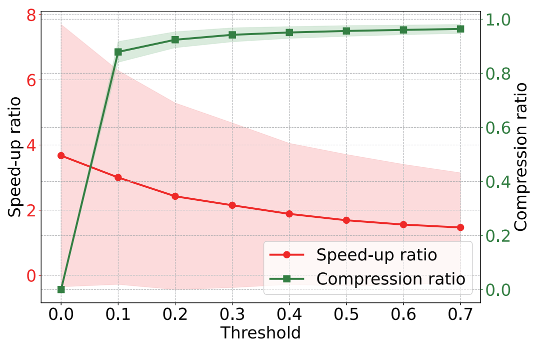

Figure 5: **The speed-up ratio and compression ratio as a function of threshold value for the PPD-VQA with $M = 4$ local nodes.** The depolarizing noise p_i in each node is set by sampling from the gaussian distribution $N(\mu, \sigma^2)$ with $\mu = 0.016$ and $\sigma = \mu/9$.

Algorithm 2 The pseudocode of PPD-VQA with gradient compression.

Require: $\boldsymbol{\theta} \in [0, 2\pi)^d$: the parameters of ansatz; L: loss function; M: the number of local nodes, and we donate M_i as the i-th local node; $Mask = (0, \cdots, 0)$ has the same dimension with $\boldsymbol{\theta_i}$ defined in section *PPD-VQA*, and $\odot$ is Hardamard product, i.e., $\boldsymbol{a} \odot \boldsymbol{b} = (a_1 b_1, \cdots, a_n b_n)$.

Ensure: optimal parameters $\boldsymbol{\theta}*$

```
 1: Calibrate threshold $thr$
 2: $\left[\nabla \bar{L}(\boldsymbol{\theta})\right]_i = \mathbf{0}$ for $i \in [1, 2, \cdots, M]$
 3: while convergence condition is not satisfied do
 4:     The central parameter server divides the parameter $\boldsymbol{\theta}$ into $M$
        parts and allocates $\boldsymbol{\theta}$ to $M$ local nodes
 5:     for Local nodes $M_i$, $\forall i \in [M]$ in parallel do
 6:         Calculate gradient component $G_i(\boldsymbol{\theta})$
 7:         $\left[\nabla \bar{L}(\boldsymbol{\theta})\right]_i = \left[\nabla \bar{L}(\boldsymbol{\theta})\right]_i + G_i(\boldsymbol{\theta})$
 8:         for $j = 1 + (i-1)\frac{d}{M}, \cdots, 1 + i\frac{d}{M}$ do
 9:             if $|\left[\nabla \bar{L}(\boldsymbol{\theta})\right]_{i,j}| > thr$ then
10:                 $Mask[j] = 1$
11:             end if
12:         end for
13:         $g_i(\boldsymbol{\theta}) = \left[\nabla \bar{L}(\boldsymbol{\theta})\right]_i \odot Mask$
14:         $\left[\nabla \bar{L}(\boldsymbol{\theta})\right]_i = \left[\nabla \bar{L}(\boldsymbol{\theta})\right]_i \odot \neg Mask$
15:     end for
16:     Synchronize the compressed gradient $g(\boldsymbol{\theta})$ by merging $\{g_i(\boldsymbol{\theta})\}_{i=1}^{M}$;
17:     Update $\boldsymbol{\theta}$ with a classical optimizer, such as ADAM;
18:     Choose a local node from $\{M\}_{i=1}^{M}$ for convergence test.
19:     if convergence condition is satisfied then
20:         break
21:     end if
22: end while
```

The idea of gradient compression is gradient clipping, which makes the gradient sparse by comparing its individual components with a threshold thr. Only the components of the gradient with larger absolute values compared with thr can be synchronized to the central

Table 1: **The comparison of the performance between PPD-VQA without gradient compression and with gradient compression.** The results of PPD-VQA under different number of local nodes ($M = 2$, $M = 4$ and $M = 8$) and noise settings ($\mu = 0.016$ and $\mu = 0.064$) are presented. In the table we count total number of iterations for all 100 instances in each setting. Communication volume CV is defined as the total number of gradient components after clipping transmitting between the central parameter server and multiple local nodes, and compression ratio is $1-CV_{\text{with}}/CV_{\text{without}}$, where CV_{with} (CV_{without}) is the communication volume for PPD-VQA with (without) gradient compression. The symbol $\rightarrow$ indicates the change of speed-up ratio from left (PPD-VQA without gradient compression) to right (PPD-VQA with gradient compression). Two typical results help us explore the relationship between acceleration of PPD-VQA and compression ratio: (top) The acceleration of PPD-VQA with gradient compression has only a slight decay when the gradient compression ratio is over 60%. (bottom) The acceleration of PPD-VQA with gradient compression decreases significantly when the gradient compression ratio is too high (over 96%).

M	**noise setting**	**without gradient compression**		**with gradient compression**		**result**	
		iterations	communication volume	iterations	communication volume	compression ratio	speed-up ratio
2	$\mu = 0.016$	2324	2324×8	2259	7016	62.3%	$1.87 \rightarrow 1.92$
	$\mu = 0.064$	2680	2680×8	2780	8565	60.1%	$2.04 \rightarrow 1.97$
4	$\mu = 0.016$	2324	2324×8	2444	3630	80.0%	$3.36 \rightarrow 3.20$
	$\mu = 0.064$	2712	2712×8	3295	2862	86.9%	$3.64 \rightarrow 2.99$
8	$\mu = 0.016$	2282	2282×8	2463	3634	80.0%	$5.71 \rightarrow 5.29$
	$\mu = 0.064$	2702	2702×8	3200	2818	87%	$6.09 \rightarrow 5.14$
2	$\mu = 0.016$	2324	2324×8	5659	701	96.3%	$1.87 \rightarrow 0.77$
	$\mu = 0.064$	2680	2680×8	6463	787	96.3%	$2.04 \rightarrow 0.84$
4	$\mu = 0.016$	2324	2324×8	6338	623	96.7%	$3.36 \rightarrow 1.23$
	$\mu = 0.064$	2712	2712×8	6410	791	96.4%	$3.64 \rightarrow 1.54$
8	$\mu = 0.016$	2282	2282×8	6043	607	96.7%	$5.71 \rightarrow 2.15$
	$\mu = 0.064$	2702	2702×8	6322	781	96.4%	$6.09 \rightarrow 2.60$

parameter server, thus ensuring that the general direction of the parameter update remains correct. The remaining components smaller than thr are still retained in corresponding local node and counted as a part of new gradient in next iteration. Thus we obtain the uncropped original $[\nabla \bar{L}(\boldsymbol{\theta})]_i$ in local node i. This method greatly reduces the actual communication bandwidth required in PPD-VQA. However, due to the existence of quantum noise, it is also unknown whether gradient compression works on PPD-VQA, so next we will perform numerical simulations to address this concern.

We test the gradient compression on a PPD-VQA with $M = 4$ local nodes, where the noise p_i in each node is set by sampling from the Gaussian distribution $N(\mu, \sigma^2)$ with $\mu = 0.016$ and $\sigma = \mu/9$. In our simulation, the threshold value thr varies from 0 to 0.7 with step 0.1, and we still implement 100 independent experiments for each setting. As shown in Fig. 5, by setting a reasonable compression threshold, we can greatly reduce the communication cost. It can be also observed that the increase of the gradient compression ratio leads to the decay of the acceleration of PPD-VQA. When $thr > 0.1$, the growth of gradient compression ratio becomes very slow, while speed-up ratio is still decreasing rapidly. Thus, we need to find a balance between the decay of acceleration advantage and reducing the communication volume. When the threshold value is 0.1, we can achieve a relatively high gradient compression ratio ($> 80\%$) without losing too much acceleration advantage ($R_S > 2.7$).

In Table 1, we further show two types of typical results for the PPD-VQA with $M = 2, 4, 8$ local nodes, and noise level $\mu = 0.016, 0.064$. For each setting, 100 independent experiments are implemented. In the first typical result, we set a reasonable compression ratio, so that the speed-up ratio is almost not lost compared to the uncompressed case. However, in this scenario, the compression ratio can still be higher than 60%, or even up to 87%, indicating that we can solve the problem on communication bottleneck without losing too much acceleration advantage of PPD-VQA. In the second typical result, to achieve a more aggressive compression ratio (above 96%), the speedup of PPD-VQA is significantly reduced. Possible reason is that fewer trainable parameters make the gradient not have the sparsity compared with deep neural networks, which leads to a significant increase in the number of iteration when we apply the gradient compression algorithm to PPD-VQA. Anyway, our experiments demonstrate that gradient compression is very suitable for PPD-VQA, even in the realistic noise scenario.

In addition to reducing communication bandwidth, gradient compression may help to mitigate errors in the experimental implementation of the PPD-VQA, due to the following two reasons:1) For most quantum computing systems, it is not easy to implement the tiny angular rotations of single-qubit quantum operations with high precision. Gradient compression can avoid updates of tiny angles and thus potentially improve experimental accuracy. 2) As the frequency of updating parameters (especially those with small gradient changes) is reduced, the number of gate operations that need to be changed by the quantum device is consequently reduced and the accumulation of quantum operation errors is naturally suppressed.

5 Conclusion

Our results show that PPD-VQA is highly promising as it achieves approximately linear acceleration over the training process of conventional VQA, both in theory and simulation results. The PPD-VQA exhibits good resilience to the excessive noise differences among local nodes, by employing the alternate training strategy. Furthermore, by adopting the gradient compression strategy, potential communication bottlenecks can also be addressed to support the future scalability of PPD-VQA.

The PPD-VQA is naturally compatible with the data-parallel distributed VQA proposed in [40], so the combination of the two approaches could enable a stronger acceleration for the training of VQA. When doing such a combination, some methods [48–50] can be employed to enhance the generalization ability of data-parallel training [49, 50]. Besides, error mitigation techniques [51, 52] have the potential to further improve the capability of PPD-VQA on near-term quantum devices. Some more complex application scenarios, such as privacy-preserving distributed VQA, requires more in-depth discussions in the future works.

Acknowledgments

H.-L. H. acknowledges support from the Youth Talent Lifting Project (Grant No. 2020-JCJQ-QT-030), National Natural Science Foundation of China (Grants No. 12274464,11905294), China Postdoctoral Science Foundation, and the Open Research Fund from State Key Laboratory of High Performance Computing of China (Grant No. 201901-01).

A The derivation of estimated gradient

As it is shown in III.A, the j-th component of the analytical gradient evaluate from QPU i at t-th iteration $\left[\nabla L(\boldsymbol{\theta}^{(t)})\right]_{i,j}$ is

$$\left[\nabla L(\boldsymbol{\theta}^{(t)})\right]_{i,j} = \frac{1}{N_D}\left[\sum_k (\hat{y_k}^{(t)} - y_k)\frac{\hat{y_k}^{(t,+j)} - \hat{y_k}^{(t,-j)}}{2} + \lambda\theta_{i,j}^{(t)}\right].$$

However, in the realistic scenario, the j-th component of estimated gradient calculated on i-th QPU $[\nabla\bar{L}(\boldsymbol{\theta}^{(t)})]_{i,j}$ is obtained by the estimated values of $\bar{y}_{k,i}^{(t)}$ and $\bar{y}_{k,i}^{(t,\pm j)}$, *i.e.*,

$$\left[\nabla\bar{L}(\boldsymbol{\theta}^{(t)})\right]_{i,j} = \frac{1}{N_D}\sum_k\left[(\bar{y_{k,i}}^{(t)} - y_k)\frac{\bar{y}_{k,i}^{(t,+j)} - \bar{y}_{k,i}^{(t,-j)}}{2} + \lambda\theta_{i,j}^{(t)}\right].$$

According to the definition and notation of the depolarizing noise model in III.A, the estimated label $\bar{y}_{k,i}^{(t)}$ has the mean value $v_{k,i}^{(t)}$ and variance $(\sigma_{k,i}^{(t)})^2$ after K measurements:

$$v_{k,i}^{(t)} = (1-\tilde{p}_i)\hat{y}_k^{(t)} + \tilde{p}_i\frac{Tr[O]}{2^n},$$
$$(\sigma_{k,i}^{(t)})^2 = \frac{1}{K}(1-\tilde{p}_i)\tilde{p}_i\left(\hat{y}_k^{(t)} - \frac{Tr[O]}{2^n}\right)^2.$$

We further introduce the random variable $\xi_{k,i}^{(t)}$ with zero mean and variance $(\sigma_{k,i}^{(t)})^2$ to describe the output of PQC on QPU i, *i.e.*,

$$\bar{y}_{k,i}^{(t)} = v_{k,i}^{(t)} + \xi_{k,i}^{(t)}.$$

Similarly, we can define $v_{k,i}^{(t,\pm j)}$ and the random variable $\xi_{k,i}^{(t,\pm j)}$ to describe the output of QPU i with shifted-parameter in the same way.

Therefore, a formulaic description of the relation between the estimated partial derivative $[\nabla\bar{L}(\boldsymbol{\theta}^{(t)})]_{i,j}$ and the analytic gradients $\left[\nabla L(\boldsymbol{\theta}^{(t)})\right]_{i,j}$ is as follows,

$$\begin{aligned}
\left[\nabla\bar{L}(\boldsymbol{\theta}^{(t)})\right]_{i,j} &= \frac{1}{N_D}\sum_k\left[(v_{k,i}^{(t)} + \xi_{k,i}^{(t)} - y_k)(\frac{v_{k,i}^{(t,+j)} - v_{k,i}^{(t,-j)} + \xi_{k,i}^{(t,+j)} - \xi_{k,i}^{(t,-j)}}{2} + \lambda\theta_{i,j}^{(t)}\right] \\
=&(1-\tilde{p}_i)^2\left[\nabla L(\boldsymbol{\theta}^{(t)})\right]_{i,j} \\
&+\frac{1}{N_D}\sum_k\left[(1-\tilde{p}_i)\tilde{p}_i(\frac{Tr[O]}{2^n} - y_k)\frac{\hat{y}_{k,i}^{(t,+j)} - \hat{y}_{k,i}^{(t,-j)}}{2} + (2\tilde{p}_i - \tilde{p}_i^2)\lambda\theta_{i,j}^{(t)}\right] \\
&+\frac{1}{N_D}\sum_k\left[(v_{k,i}^{(t)} - y_k)(\xi_{k,i}^{(t,+j)} - \xi_{k,i}^{(t,-j)}) + \frac{\hat{y}_{k,i}^{(t,+j)} - \hat{y}_{k,i}^{(t,-j)}}{2}\xi_{k,i}^{(t)} + \xi_{k,i}^{(t)}(\xi_{k,i}^{(t,+j)} - \xi_{k,i}^{(t,-j)})\right] \\
=&(1-\tilde{p}_i)^2\left[\nabla L(\boldsymbol{\theta}^{(t)})\right]_{i,j} + C_{i,j}^{(t)} + \varsigma_{i,j}^{(t)},
\end{aligned}$$

where

$$C_{i,j}^{(t)}=\frac{1}{N_D}\sum_k\left[(1-\tilde{p}_i)\tilde{p}_i(\frac{Tr[O]}{2^n}-y_k)\frac{\hat{y}_{k,i}^{(t,+j)}-\hat{y}_{k,i}^{(t,-j)}}{2}+(2\tilde{p}_i-\tilde{p}_i^2)\lambda\theta_{i,j}^{(t)}\right],$$

$$\varsigma_{i,j}^{(t)}=\frac{1}{N_D}\sum_k\left[(v_{k,i}^{(t)}-y_k)(\xi_{k,i}^{(t,+j)}-\xi_{k,i}^{(t,-j)})+\frac{\hat{y}_{k,i}^{(t,+j)}-\hat{y}_{k,i}^{(t,-j)}}{2}\xi_{k,i}^{(t)}+\xi_{k,i}^{(t)}(\xi_{k,i}^{(t,+j)}-\xi_{k,i}^{(t,-j)})\right].$$

Obviously, $\varsigma_{i,j}^{(t)}$ has the mean zero, where the random variables $\xi_{k,i}^{(t)}$, $\xi_{k,i}^{(t,\pm j)}$ in above formula are independent, because these random variables arise when we separately calculate the expectation of the observable with different shifted parameters.

B Bias term analysis

In this subsection, we give an error analysis on estimated gradient. By leveraging the explicit form of estimated gradient in Appendix A, estimated gradient has following average bias term compared to the ideal gradient,

$$\begin{aligned}bias\ \ term&=|E_{\xi_{k,i}^{(t)},\xi_{k,i}^{(t,\pm j)}}(\left[\nabla\bar{L}(\boldsymbol{\theta}^{(t)})\right]_{i,j}-\left[\nabla L(\boldsymbol{\theta}^{(t)})\right]_{i,j})|\\&=|(\tilde{p}_i^2-2\tilde{p}_i)\left[\nabla L(\boldsymbol{\theta}^{(t)})\right]_{i,j}+C_{i,j}^{(t)}|\\&\leqslant|(\tilde{p}_i^2-2\tilde{p}_i)\left[\nabla L(\boldsymbol{\theta}^{(t)})\right]_{i,j}|+|C_{i,j}^{(t)}|\\&\leqslant 2\tilde{p}_i(\frac{1}{2}+3\lambda\pi)+(1+6\lambda\pi)\tilde{p}_i\\&=(2+9\lambda\pi)\tilde{p}_i\,,\end{aligned}$$

where the inequality uses the upper bound of $\left[\nabla L(\boldsymbol{\theta}^{(t)})\right]_{i,j}$, *i.e.*, $\left[\nabla L(\boldsymbol{\theta}^{(t)})\right]_{i,j}\leqslant 1\times\frac{1}{2}+3\lambda\pi$ when $\theta\in[\pi,3\pi]$ and $\hat{y}_k^{(t)},\hat{y}_k^{(t,\pm j)}\in[0,1]$, and the upper bound of $C_{i,j}^{(t)}$, i.e., $C_{i,j}^{(t)}\leqslant\tilde{p}_i+2\tilde{p}_i\cdot\lambda\cdot 3\pi$.

Therefore, the larger the noise, the larger the upper bound of the bias term, which visually demonstrates the effect of noise on the gradient calculation. However, in actual training process, we pay more attention to the estimated gradient $[\nabla\bar{L}(\boldsymbol{\theta}^{(t)})]_{i,j}$ containing the bias term, because VQAs are typical adaptive noise approaches.

References

[1] P. W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM J. Comput. **26**, 1484 (1997), doi:10.1137/S0097539795293172.

[2] C.-Y. Lu, D. E. Browne, T. Yang and J.-W. Pan, *Demonstration of a compiled version of Shor's quantum factoring algorithm using photonic qubits*, Phys. Rev. Lett. **99**, 250504 (2007), doi:10.1103/PhysRevLett.99.250504.

[3] H.-L. Huang et al., *Experimental blind quantum computing for a classical client*, Phys. Rev. Lett. **119**, 050503 (2017), doi:10.1103/PhysRevLett.119.050503.

[4] L. K. Grover, *A fast quantum mechanical algorithm for database search*, in *Proceedings of the twenty-eighth annual ACM symposium on theory of computing*, Association for Computing Machinery, New York, USA, ISBN 9780897917858 (1996), doi:10.1145/237814.237866.

[5] G. L. Long, Y. Song Li, W. L. Zhang and L. Niu, *Phase matching in quantum searching*, Phys. Lett. A **262**, 27 (1999), doi:10.1016/S0375-9601(99)00631-3.

[6] A. W. Harrow, A. Hassidim and S. Lloyd, *Quantum algorithm for linear systems of equations*, Phys. Rev. Lett. **103**, 150502 (2009), doi:10.1103/PhysRevLett.103.150502.

[7] X.-D. Cai et al., *Experimental quantum computing to solve systems of linear equations*, Phys. Rev. Lett. **110**, 230501 (2013), doi:10.1103/PhysRevLett.110.230501.

[8] H.-L. Huang et al., *Homomorphic encryption experiments on IBM's cloud quantum computing platform*, Front. Phys. **12**, 120305 (2016), doi:10.1007/s11467-016-0643-9.

[9] C. Gidney and M. Ekerå, *How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits*, Quantum **5**, 433 (2021), doi:10.22331/q-2021-04-15-433.

[10] Y. Zhao et al., *Realization of an error-correcting surface code with superconducting qubits*, Phys. Rev. Lett. **129**, 030501 (2022), doi:10.1103/PhysRevLett.129.030501.

[11] J. Preskill, *Quantum computing in the NISQ era and beyond*, Quantum **2**, 79 (2018), doi:10.22331/q-2018-08-06-79.

[12] H.-L. Huang, D. Wu, D. Fan and X. Zhu, *Superconducting quantum computing: A review*, Sci. China Inf. Sci. **63**, 180501 (2020), doi:10.1007/s11432-020-2881-9.

[13] F. Arute et al., *Quantum supremacy using a programmable superconducting processor*, Nature **574**, 505 (2019), doi:10.1038/s41586-019-1666-5.

[14] H.-S. Zhong et al., *Quantum computational advantage using photons*, Science **370**, 1460 (2020), doi:10.1126/science.abe8770.

[15] Y. Wu et al., *Strong quantum computational advantage using a superconducting quantum processor*, Phys. Rev. Lett. **127**, 180501 (2021), doi:10.1103/PhysRevLett.127.180501.

[16] Q. Zhu et al., *Quantum computational advantage via 60-qubit 24-cycle random circuit sampling*, Sci. Bull. **67**, 240 (2022), doi:10.1016/j.scib.2021.10.017.

[17] K. Bharti et al., *Noisy intermediate-scale quantum algorithms*, Rev. Mod. Phys. **94**, 015004 (2022), doi:10.1103/RevModPhys.94.015004.

[18] M. Cerezo et al., *Variational quantum algorithms*, Nat. Rev. Phys. **3**, 625 (2021), doi:10.1038/s42254-021-00348-9.

[19] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, (arXiv preprint) doi:10.48550/arXiv.1412.6980.

[20] J. R. McClean, J. Romero, R. Babbush and A. Aspuru-Guzik, *The theory of variational hybrid quantum-classical algorithms*, New J. Phys. **18**, 023023 (2016), doi:10.1088/1367-2630/18/2/023023.

[21] I. Cong, S. Choi and M. D. Lukin, *Quantum convolutional neural networks*, Nat. Phys. **15**, 1273 (2019), doi:10.1038/s41567-019-0648-8.

[22] V. Havlíček et al., *Supervised learning with quantum-enhanced feature spaces*, Nature **567**, 209 (2019), doi:10.1038/s41586-019-0980-2.

[23] J. Liu et al., *Hybrid quantum-classical convolutional neural networks*, Sci. China Phys. Mech. Astron. **64**, 290311 (2021), doi:10.1007/s11433-021-1734-3.

[24] H.-L. Huang et al., *Demonstration of topological data analysis on a quantum processor*, Optica **5**, 193 (2018), doi:10.1364/OPTICA.5.000193.

[25] S. Lloyd and C. Weedbrook, *Quantum generative adversarial learning*, Phys. Rev. Lett. **121**, 040502 (2018), doi:10.1103/PhysRevLett.121.040502.

[26] H.-L. Huang et al., *Experimental quantum generative adversarial networks for image generation*, Phys. Rev. Appl. **16**, 024051 (2021), doi:10.1103/PhysRevApplied.16.024051.

[27] M. S. Rudolph, N. B. Toussaint, A. Katabarwa, S. Johri, B. Peropadre and A. Perdomo-Ortiz, *Generation of high-resolution handwritten digits with an ion-trap quantum computer*, Phys. Rev. X **12**, 031010 (2022), doi:10.1103/PhysRevX.12.031010.

[28] S. Ebadi et al., *Quantum optimization of maximum independent set using Rydberg atom arrays*, Science **376**, 1209 (2022), doi:10.1126/science.abo6587.

[29] M. P. Harrigan et al., *Quantum approximate optimization of non-planar graph problems on a planar superconducting processor*, Nat. Phys. **17**, 332 (2021), doi:10.1038/s41567-020-01105-y.

[30] E. Farhi, J. Goldstone, S. Gutmann and L. Zhou, *The quantum approximate optimization algorithm and the Sherrington-Kirkpatrick model at infinite size*, Quantum **6**, 759 (2022), doi:10.22331/q-2022-07-07-759.

[31] E. Farhi and A. W. Harrow, *Quantum supremacy through the quantum approximate optimization algorithm*, (arXiv preprint) doi:10.48550/arXiv.1602.07674.

[32] L. Zhou, S.-T. Wang, S. Choi, H. Pichler and M. D. Lukin, *Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices*, Phys. Rev. X **10**, 021067 (2020), doi:10.1103/PhysRevX.10.021067.

[33] M. Gong et al., *Quantum neuronal sensing of quantum many-body states on a 61-qubit programmable superconducting processor*, Sci. Bull. (2023), doi:10.1016/j.scib.2023.04.003.

[34] A. Kandala et al., *Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets*, Nature **549**, 242 (2017), doi:10.1038/nature23879.

[35] F. Arute et al., *Hartree-Fock on a superconducting qubit quantum computer*, Science **369**, 1084 (2020), doi:10.1126/science.abb9811.

[36] J. Romero et al., *Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz*, Quantum Sci. Technol. **4**, 014008 (2018), doi:10.1088/2058-9565/aad3e4.

[37] C. Cade, L. Mineh, A. Montanaro and S. Stanisic, *Strategies for solving the Fermi-Hubbard model on near-term quantum computers*, Phys. Rev. B **102**, 235122 (2020), doi:10.1103/PhysRevB.102.235122.

[38] C. Hempel et al., *Quantum chemistry calculations on a trapped-ion quantum simulator*, Phys. Rev. X **8**, 031022 (2018), doi:10.1103/PhysRevX.8.031022.

[39] Y. Nam et al., *Ground-state energy estimation of the water molecule on a trapped-ion quantum computer*, npj Quantum Inf. **6**, 33 (2020), doi:10.1038/s41534-020-0259-3.

[40] Y. Du, Y. Qian, X. Wu and D. Tao, *A distributed learning scheme for variational quantum algorithms*, IEEE Trans. Quantum Eng. **3**, 1 (2022), doi:10.1109/TQE.2022.3175267.

[41] K. Mitarai, M. Negoro, M. Kitagawa and K. Fujii, *Quantum circuit learning*, Phys. Rev. A **98**, 032309 (2018), doi:10.1103/PhysRevA.98.032309.

[42] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac and N. Killoran, *Evaluating analytic gradients on quantum hardware*, Phys. Rev. A **99**, 032331 (2019), doi:10.1103/PhysRevA.99.032331.

[43] M. M. Wilde, *Quantum information theory*, Cambridge University Press, Cambridge, UK, ISBN 9781107034259 (2013), doi:10.1017/CBO9781139525343.

[44] Y. Du, M.-H. Hsieh, T. Liu, S. You and D. Tao, *Learnability of quantum neural networks*, PRX Quantum **2**, 040337 (2021), doi:10.1103/PRXQuantum.2.040337.

[45] D. A. Meyer and N. R. Wallach, *Global entanglement in multiparticle systems*, J. Math. Phys. **43**, 4273 (2002), doi:10.1063/1.1497700.

[46] J. Verbraeken et al., *A survey on distributed machine learning*, ACM Comput. Surv. **53**, 1 (2020), doi:10.1145/3377454.

[47] Y. Lin, S. Han, H. Mao, Y. Wang and W. J. Dally, *Deep gradient compression: Reducing the communication bandwidth for distributed training*, (arXiv preprint) doi:10.48550/arXiv.1712.01887.

[48] W. Zhu, X. Chen and W. B. Wu, *Online covariance matrix estimation in stochastic gradient descent*, J. Am. Stat. Assoc. **118**, 393 (2021), doi:10.1080/01621459.2021.1933498.

[49] M. H. Zhu, L. N. Ezzine, D. Liu and Y. Bengio, *Fedilc: Weighted geometric mean and invariant gradient covariance for federated learning on non-iid data*, (arXiv preprint) doi:10.48550/arXiv.2205.09305.

[50] A. Rame, C. Dancette and M. Cord, *Fishr: Invariant gradient variances for out-of-distribution generalization*, in *Proceedings of the 39th international conference on machine learning*, Proc. Mach. Learn. Res. **162**, 18347 (2022).

[51] H.-L. Huang et al., *Near-term quantum computing techniques: Variational quantum algorithms, error mitigation, circuit compilation, benchmarking and classical simulation*, Sci. China Phys. Mech. Astron. **66**, 250302 (2023), doi:10.1007/s11433-022-2057-y.

[52] Z. Cai et al., *Quantum error mitigation*, (arXiv preprint) doi:10.48550/arXiv.2210.00921.