ACCELERATING VARIATIONAL QUANTUM ALGORITHMS WITH MULTIPLE QUANTUM PROCESSORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Variational quantum algorithms (VQAs) are prime contenders that exploit near-term quantum machines to gain computational advantages over classical algorithms. As such, how to accelerate the optimization of modern VQAs has attracted great attention in past years. Here we propose a `QU`antum `DI`stributed `O`ptimization scheme (abbreviated as QUDIO) to address this issue. Conceptually, QUDIO utilizes a classical central server to partition the learning problem into multiple subproblems and allocate them to a set of quantum local nodes. In the training stage, all local nodes proceed with parallel training and the classical server synchronizes optimization information among local nodes timely. In doing so, we prove a sublinear convergence rate of QUDIO in the number of global iterations under the ideal scenario. Moreover, when the imperfection of the quantum system is considered, we prove that an increased synchronization time leads to a degraded convergence rate or even incurs divergent optimization. Numerical results on standard benchmarks illustrate that QUDIO can surprisingly reach a superlinear clock-time speedup in terms of the number of local nodes. Our proposal can be readily mixed with other advanced VQAs-based techniques to narrow the gap between the state of the art and applications with the quantum advantage [1].

1 INTRODUCTION

Deep learning techniques have penetrated the world during past decades Goodfellow et al. (2016). Prototypical applications comprise using deep neural networks (DNNs) to facilitate online shopping Zhang et al. (2019), to accelerate molecule design Senior et al. (2020), and to enhance language translation Devlin et al. (2019). Notably, the success of deep learning heavily relies on distributed hardware and distributed optimization techniques Dean et al. (2012) in the sense that multiple GPU cards are employed to collaboratively process the same learning task. For instance, to ensure that the training of deep bidirectional transformers can be completed within a reasonable time (e.g., 4 days), in total 64 GPU cards are used to conduct the distributed optimization Devlin et al. (2019). However, the price to pay is ten thousands dollars and emitting 1438 lbs of carbon dioxide Strubell et al. (2019). This considerable resource-consumption signifies that deep learning models will become hard to develop and optimize when the problem size is continuously enlarged. Therefore, it is highly demanded to seek novel techniques to resolve complicated problems in a fast, economic, and environmentally friendly way.

The oath of quantum computing is to accomplish certain tasks beyond the reach of classical computers Harrow & Montanaro (2017). During past years, big breakthrough has been achieved towards this goal, e.g., a demonstration of the quantum supremacy experiment in the task of sampling the output of a pseudo-random quantum circuit Arute et al. (2019). Among various quantum learning models, variational quantum algorithms (VQAs) Bharti et al. (2021); Cerezo et al. (2020), which are formed by parameterized quantum circuits (PQCs) and a classical optimizer, have attracted great attention from industry and academia. The popularity of VQAs origins from the versatility of PQCs, which guarantees their efficient implementations on the noisy intermediate-scale quantum (NISQ) machines Preskill (2018), as well as theoretical evidence of quantum superiority Abbas et al. (2020); Huang et al. (2021b;a). Moreover, prior studies have exhibited the progress of VQAs of accomplishing diverse learning tasks, e.g., machine learning issues such as data classification Havlíček et al.

[1]The source code is available at `https://anonymous.4open.science/r/QUDIO-1076/`.

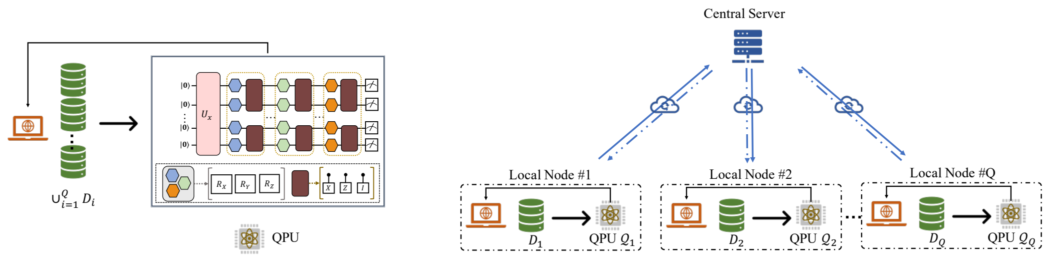


Figure 1: **The scheme of VQAs and quantum distributed optimization scheme (QUDIO).** The left panel illustrates the workflow of conventional VQAs. The right panel presents the workflow of QUDIO, which consists of multiple local nodes and a central server. Each local node only manipulates a subgroup of the given problem in parallel, while the central server communicates with all local nodes and synchronizes their results. In this way, QUDIO can accelerate various VQAs.

(2019); Mitarai et al. (2018); Schuld & Killoran (2019) and image generation Huang et al. (2020); Rudolph et al. (2020), combinatorial optimization Farhi et al. (2014); Harrigan et al. (2021), and finance Alcazar et al. (2020); Coyle et al. (2021).

Despite the tantalizing achievements, modern VQAs are generally plagued by expensive or even unaffordable *wall-clock* time for large-volume data, hampered by the regulation such that only a *single* quantum chip is employed in optimization. For concreteness, we exemplify the machinery of quantum neural networks (QNNs), as a crucial subclass of VQAs, when dealing with classification tasks Havlíček et al. (2019). As shown in the left panel of Fig. 1, to correctly classify n training examples, the classical optimizer iteratively feeds each example to PQCs and then leverages the discrepancy between the obtained n predictions of QNN and n labels to conduct the gradient-based optimization. In this way, the classical optimizer needs to query a single quantum processor at least $O(n)$ times to complete one iteration. Such overhead prohibits the applicability of QNNs for large n Qian et al. (2021). With this regard, it is of great importance to enhance the computational efficiency of VQAs, as the necessary condition to pursue quantum advantages.

To conquer the above issues, here we devise an efficient `QU`antum `DI`stributed `O`ptimization scheme (abbreviated as QUDIO) to accelerate the optimization of VQAs. Two key ingredients of QUDIO are *a classical central server* and *Q local nodes*, i.e., each of them consists of a quantum processor and a classical optimizer. Conceptually, the classical server partitions the learning problem into Q subproblems and allocates them to Q local nodes. During the training procedure, all local nodes proceed optimization in parallel and the classical server synchronizes optimization information among local nodes timely. An attractive property of QUDIO is *adequately* utilizing the accessible quantum resources to accelerate VQAs, owing to its compatibility. Namely, the deployed quantum processors are allowed to be any type of quantum hardware such as linear optical, ion-trap, and superconducting quantum chips. Such compatibility contributes to apply a wide class of VQAs to manipulate varied large-scale computational problems and seek potential quantum advantages by unifying quantum powers in a maximum extent. To our best knowledge, QUDIO is *the first distributed scheme* oriented to the acceleration of VQAs with *theoretical guarantee*. Moreover, the achieved empirical results in this study indicate that:

QUDIO can linearly accelerate the optimization of VQAs when Q is linearly increased.

Contributions. We summarize our main results below.

1. We analyze the convergence rate of QUDIO under the ideal scenario. Particularly, we prove an asymptotic convergence between QUDIO and conventional VQAs with a single quantum processor. Moreover, unlike distributed methods in DNNs, QUDIO are immune to the communication bottleneck, due to the small number of trainable parameters in VQAs. These results can not only be employed as theoretical guidance to assure good performance of QUDIO, but also motive us to devise more advanced distributed-VQAs schemes.

2. We further derive the convergence rate of QUDIO in the NISQ case, where the depolarization noise is employed to simulate the imperfection of the quantum system. The achieved result implies that the convergence rate of QUDIO could become degraded with respect to the amplified system noise and the decreased number of quantum measurements. Remark-

ably, the convergence analysis of QUDIO *sharply contrasts with classical distributed optimization methods* Boyd et al. (2011); Dean et al. (2012), due to the following fact. That is, the gradients information operated in VQAs is biased, induced by the system imperfection such as sample error and gate noise, whereas most classical distributed methods assume the unbiased gradients information. Such discrepancy means that naively imitating classical distributed algorithms to design distributed VQAs is suboptimal, since the inevitable biased gradient information in the quantum scenario may incur a deficient convergence.

3. We conduct extensive numerical simulations to validate the computational efficiency of our scheme. In particular, QUDIO is exploited to accomplish the image classification task (see Section 4) and the ground energy of hydrogen molecule estimation task (see Appendix D) under both the ideal and noisy scenarios. The achieved simulation results validate that QUDIO realizes sublinear and even superlinear speedups compared with conventional QNNs and VQEs. The source code of QUDIO is available at the Github repository `xxx`.

2 BACKGROUNDS AND RELATED WORK

Basic notations. The fundamental unit in quantum computation is quantum bit (*qubit*), which refers to a two-dimensional vector. Under Dirac notation, a qubit state is defined as $|\boldsymbol{\alpha}\rangle = a_0 |0\rangle + a_1 |1\rangle \in \mathbb{C}^2$, where $|0\rangle = [1,0]^\top$ and $|1\rangle = [0,1]^\top$ specify two unit bases, and the coefficients $a_0, a_1 \in \mathbb{C}$ satisfy $|a_0|^2 + |a_1|^2 = 1$. An N-qubit state is denoted by $|\Psi\rangle = \sum_{i=1}^{2^N} a_i |i\rangle \in \mathbb{C}^{2^N}$, where $|i\rangle \in \mathbb{R}^{2^N}$ is the unit vector whose i-th entry being 1 and other entries are 0, and $\sum_{i=0}^{2^N-1} |a_i|^2 = 1$ with $a_i \in \mathbb{C}$. Besides Dirac notation, the density matrix can be used to describe more general qubit states. For example, the density matrix of the state $|\Psi\rangle$ is $\rho = |\Psi\rangle\langle\Psi| \in \mathbb{C}^{2^N \times 2^N}$. For a set of qubit states $\{p_i, |\Psi_i\rangle\}_{i=1}^m$ with $p_i > 0$, $\sum_{i=1}^m p_i = 1$, and $|\Psi_i\rangle \in \mathbb{C}^{2^N}$ for $\forall i \in m$, its density matrix is $\rho = \sum_{i=1}^m p_i \rho_i$ with $\rho_i = |\psi_i\rangle\langle\psi_i|$ and $\mathrm{Tr}(\rho) = 1$.

There are three types of quantum operations using to manipulate qubit states, which are *quantum gates*, *quantum channels*, and *quantum measurements*. Specifically, quantum gates, as unitary transformations, can be treated as the computational toolkit for quantum circuit models, i.e., an N-qubit gate $U \in \mathcal{U}(2^N)$ obeys $UU^\dagger = \mathbb{I}_{2^N}$, where $\mathcal{U}(\cdot)$ stands for the unitary group. In the open system, quantum channels are applied to formalize the evolving of qubits states instead of unitary. Mathematically, every quantum channel $\mathcal{E}(\cdot)$ is a linear, completely positive, and trace-preserving map Nielsen & Chuang (2010). A special quantum channel is called the *depolarization channel*, which is defined as $\mathcal{E}_p(\rho) = (1-p)\rho + p\frac{\mathbb{I}_{2^N}}{2^N}$. Quantum measurements is extracting quantum information of the evolved state, which contains the computation result, into the classical form. Specifically, applying the measurement $\{\Pi_i\}$ to the state ρ, the probability of outcome i is given by $\Pr(i) = \mathrm{Tr}(\rho\Pi_i)$.

Quantum neural networks. Denote the given dataset as $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$, where $\boldsymbol{x}_i \in \mathbb{R}^{D_c}$ and $y_i \in \mathbb{R}$ refer to the features and label for the i-th example. The generic form of the output (prediction) of quantum neural networks (QNNs) Du et al. (2020a); Havlíček et al. (2019) is

$$\hat{y}_i \equiv h(\boldsymbol{\theta}, O, \rho_i) = \mathrm{Tr}(OU(\boldsymbol{\theta})\rho_i U(\boldsymbol{\theta})^\dagger),\ \forall i \in [n], \tag{1}$$

where ρ_i, $U(\boldsymbol{\theta})$, and $O \in \mathbb{C}^{2^N \times 2^N}$ are the encoded N-qubit state corresponding to $\boldsymbol{x}_i$, the ansatz with d_Q parameters (i.e., $\boldsymbol{\theta} \in [0, 2\pi)^{d_Q}$), and the fixed quantum observable, respectively. Notably, the versatility of QNNs arises from diverse data embedding strategies (e.g., preparing ρ_i by basis, amplitude, and qubit encoding methods LaRose & Coyle (2020)), flexible architectures of the ansatz $U(\boldsymbol{\theta})$ (e.g., hardware-efficient and tensor-network based ansatzes), and the agile choice of O. The aim of QNNs is to seek the optimal parameters $\boldsymbol{\theta}^*$ that minimize a predefined loss $\mathcal{L}$. Throughout the whole work, we specify $\mathcal{L}$ as the mean square error with l_2-norm regularizer, i.e.,

$$\mathcal{L}(\boldsymbol{\theta}, \mathcal{D}) = \frac{1}{2n}\sum_{i=1}^n (h(\boldsymbol{\theta}, O, \rho_i) - y_i)^2 + \lambda\|\boldsymbol{\theta}\|_2^2, \tag{2}$$

where $\lambda \geq 0$ refers to the regularizer coefficient. Moreover, we set $\{O, \mathbb{I} - O\}$ in Eq. (1) as a two-outcome positive operator valued measure (POVM) Nielsen & Chuang (2010). The updating rule of $\boldsymbol{\theta}$ is specified to the *stochastic gradient* method, i.e., in the *ideal scenario*, we have

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta\nabla\mathcal{L}(\boldsymbol{\theta}^{(t)}, \boldsymbol{x}^{(t)}), \tag{3}$$

where η is the learning rate, $\boldsymbol{x}^{(t)}$ is randomly sampled from $\mathcal{D}$, and $\boldsymbol{\theta}^{(t)}$ refers to the trainable parameters at the t-th step. The evaluation of $\nabla\mathcal{L}(\boldsymbol{\theta}^{(t)}, \boldsymbol{x}^{(t)})$ is typically achieved via the parameter shift rule Mitarai et al. (2018), i.e., its j-th component for $\forall j \in [d_Q]$ yields

$$\nabla\mathcal{L}(\boldsymbol{\theta}^{(t)}, \boldsymbol{x}^{(t)}) = (\hat{y}^{(t)} - y^{(t)})\frac{\hat{y}^{(t,+_j)} - \hat{y}^{(t,-_j)}}{2} + \lambda\boldsymbol{\theta}_j^{(t)}, \tag{4}$$

where $\hat{y}^{(t,\pm_j)} = h(\boldsymbol{\theta}^{(t,\pm)}, O, \rho^{(t)})$ denote the outputs of QNNs with shifted parameters $\boldsymbol{\theta}^{(t,\pm)} = \boldsymbol{\theta}^{(t)} \pm \frac{\pi}{2}\boldsymbol{e}_j$, $\boldsymbol{e}_j$ is the unit vector whose j-th entry equals to 1. After T iterations, the updated parameters $\boldsymbol{\theta}^{(T)}$ serve as the approximation of the optimal solution $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \mathcal{D})$.

Related work. Prior literature related to our work can be divided into two categories. The first category is distributed deep learning algorithms and the second category is quantum distributed computation. Particularly, in the first category, the acceleration of training DNNs is achieved by involving multiple GPUs to fulfill the joint optimization Dean et al. (2012); Bottou et al. (2018). Despite of a similar manner, our proposal is sharply inconsistent with classical distributed optimization methods from the following two perspectives. First, the gradients information operated in VQAs is biased, induced by the system imperfection such as sample error and gate noise, whereas most classical distributed methods assume the unbiased gradients information. Next, unlike distributed methods in DNNs, VQAs are immune to the communication bottleneck, due to the small number of trainable parameters. In the second category, quantum distributed computation Beals et al. (2013) focuses on using multiple less powerful quantum circuits to simulate a standard quantum circuit, which differs from our aim. Although some quantum software Bergholm et al. (2018); et. al. (2019) showcases parallelling variational quantum eigen-solvers (VQEs) Peruzzo et al. (2014), how to devise distributed-VQAs optimization schemes with both runtime boost and convergence guarantee remains largely unknown.

3 ACCELERATE QNN BY QUANTUM DISTRIBUTED OPTIMIZATION SCHEME

We depict the paradigm of QUDIO in the right panel of Fig. 1 and present the corresponding Pseudocode in Alg. 1. The algorithmic implementation of QUDIO consists of three components.

1. At the preprocessing stage, the central server partitions the given problem into Q subproblems and allocates these subproblems to Q local node $\{\mathcal{Q}_i\}_{i=1}^Q$ (see Sections 4 for elaboration of the problem partition).
2. The training procedure of QUDIO obeys an iterative manner. Define the total number of global and local steps as T and W, respectively. At the t-th global step with $t \in [T]$, the central server first dispatches the synchronized parameters $\boldsymbol{\theta}^{(t)}$ to Q local nodes, which correspond to initial parameters for local updates (Line 5). *With a slight abuse of notation*, we denote $\boldsymbol{\theta}_i^{(t,w)}$ to describe trainable parameters for the i-th local node at the w-th *local step*. As such, the parameters for all local nodes at the t-th global step satisfy $\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}_i^{(t,w=0)}$, $\forall i \in [Q]$. After initialization, all local nodes $\{\mathcal{Q}_i\}$ proceed W local iterations independently to update $\boldsymbol{\theta}_i^{(t,w)}$, $\forall i \in [Q]$ (highlighted by the pink region in Lines 7-8). Once all local updates are fulfilled, the central server collects parameters $\{\boldsymbol{\theta}_i^{(t,W)}\}_{i=1}^Q$ from all local nodes to execute synchronization of trainable parameters (Line 11), i.e., $\boldsymbol{\theta}^{(t+1)} = \frac{1}{Q}\sum_{i=1}^Q \boldsymbol{\theta}_i^{(t,W)}$. This completes the t-th global step.
3. Through repeating the above procedure with T global steps, the classical central server outputs the synchronized $\boldsymbol{\theta}^{(T)} = \frac{1}{Q}\sum_{i=1}^Q \boldsymbol{\theta}_i^{(T,W)}$ to estimate the optimal solution $\boldsymbol{\theta}^*$.

Wall-clock time analysis. In principle, compared with conventional VQAs with a single quantum processor, the parallel optimization mechanism presented above enables QUDIO to *reduce the wall-clock time of training VQAs by a constant factor that in terms of the number of local nodes Q*. This linear speedup is warranted by the small amount of trainable parameters for most VQAs, which permits a low communication overhead.

Applications of QUDIO. A core component of QUDIO is the approach of decomposing the given problem into Q parts, which in turn results in the varied forms of the estimated gradients $\{g_i^{(t,w)}\}$ in

Algorithm 1 QUDIO The yellow and pink boxes refer to the local nodes and the central server, respectively.

```
1: Input: The initialized parameters $\boldsymbol{\theta}^{(0)} \in [0, 2\pi)^{d_Q}$, the employed loss function $\mathcal{L}$, the hyper-parameters $\{Q, \eta, W, T\}$
2: The central server partitions the given problem into $Q$ parts and allocates them to $Q$ local nodes
3: for $t = 0, \cdots, T-1$ do
4:     for Quantum processor $\mathcal{Q}_i$, $\forall i \in [Q]$ in parallel do
5:         $\boldsymbol{\theta}_i^{(t,0)} = \boldsymbol{\theta}^{(t)}$
6:         for $w = 0, \cdots, W-1$ do
7:             Compute the estimate stochastic gradient $g_i^{(t,w)}$
8:             Update $\boldsymbol{\theta}_i^{(t,w+1)} = \boldsymbol{\theta}_i^{(t,w)} - \eta g_i^{(t,w)}$
9:         end for
10:    end for
11:    Synchronize $\boldsymbol{\theta}^{(t+1)} = \frac{1}{Q}\sum_{i=1}^{Q} \boldsymbol{\theta}_i^{(t,W)}$
12: end for
13: Output: $\boldsymbol{\theta}^{(T)}$
```

Line 7 of Alg. 1. For the purpose of elucidating, in the rest of the main text, we focus on elaborating the way of problem partition and the gradients estimation when applying QUDIO to speed up the training of QNNs with both empirical and theoretical evidence. In Appendix D, we exhibit how to utilize QUDIO to accelerate **variational quantum eigensolvers** (VQEs), as a crucial paradigm of VQAs in the regime of quantum chemistry. Furthermore, although our work mainly focuses on accelerating QNNs and VQEs, QUDIO can be *effectively extended to accelerate* other VQA-based paradigms such as quantum approximate optimization algorithms Farhi et al. (2014); Hadfield et al. (2019); Zhou et al. (2020).

4 Accelerating the optimization of QNN by QUDIO

We now elaborate on how to exploit QUDIO in Alg. 1 to accelerate the training of QNN formulated in Eq. (2). Concretely, at the preprocessing stage, the central server splits the dataset $\mathcal{D}$ into Q subgroups $\{\mathcal{D}_i\}_{i=1}^{Q}$ and assigns them into Q local nodes $\{\mathcal{Q}_i\}_{i=1}^{Q}$. In the training procedure, QUDIO adopts an iterative strategy presented in Lines 3-11 of Alg. 1 to optimize the trainable parameters. At the t-th global step, when Q local nodes receive the synchronized parameters $\boldsymbol{\theta}^{(t)}$ sent by the central server, they proceed W local updates independently. Define $\boldsymbol{\theta}_i^{(t,w=0)} = \boldsymbol{\theta}^{(t)}$ for $\forall i \in [Q]$ and $\forall w \in [W]$. The updating rule of $\mathcal{Q}_i$ associated with the *stochastic gradient descent optimizer* satisfies

$$\boldsymbol{\theta}_i^{(t,w+1)} = \boldsymbol{\theta}_i^{(t,w)} - \eta g_i(\boldsymbol{\theta}_i^{(t,w)}, \boldsymbol{x}_i^{(t,w)}) \in [0, 2\pi)^{d_Q}, \tag{5}$$

where η is the learning rate, $(\boldsymbol{x}_i^{(t,w)}, y_i^{(t)})$ refers to the i-th example uniformly sampled from $\mathcal{D}_i$, and $g_i(\boldsymbol{\theta}_i^{(t,w)}, \boldsymbol{x}_i^{(t,w)})$ denotes the estimation of $\nabla\mathcal{L}(\boldsymbol{\theta}_i^{(t,w)}, \boldsymbol{x}_i^{(t,w)})$ induced by the system noise and sample error. Once all local updates are finished, the central server receives $\{\boldsymbol{\theta}_i^{(t,W)}\}_{i=1}^{Q}$ and synchronizes them to update the global trainable parameters $\boldsymbol{\theta}^{(t)}$ in Line 11 of Alg. 1. Through repeating the above process with T times, QUDIO outputs $\boldsymbol{\theta}^{(T)} = \frac{1}{Q}\sum_{i=1}^{Q}\boldsymbol{\theta}_i^{(t,W)}$ to estimate the optimal solution $\boldsymbol{\theta}^*$.

In the rest of this section, we analyze the convergence rate of QUDIO. The crux to achieve this goal is establishing the mathematical expression of the estimated gradient g_i in Eq. (5). More precisely, suppose that the depolarization noise channel $\mathcal{N}_p(\cdot)$ is injected to each quantum circuit depth, i.e., $\mathcal{N}_p(\rho) = (1-p)\rho + p\frac{\mathbb{I}}{2^N}$. The quantum state before measurements yields

$$\gamma_i^{(t,w)} = (1-\tilde{p})U(\boldsymbol{\theta}^{(t,w)})\rho_i^{(t)}U(\boldsymbol{\theta}^{(t,w)})^\dagger + \tilde{p}\frac{\mathbb{I}}{2^N},$$

where $\tilde{p} = 1 - (1-p)^{L_Q}$ and L_Q refers to the total circuit depth Du et al. (2020a). Then applying quantum measurement to the state $\gamma_i^{(t,w)}$ produces the outcome that can be viewed as a binary random variable with the Bernoulli distribution, i.e., $V_k^{(t,w)} \sim \mathrm{Ber}(\mathrm{Tr}(O\gamma_i^{(t,w)}))$. In this way, the sample mean $\bar{y}_i^{(t,w)} = \sum_{k=1}^{K} V_k^{(t,w)}/K$ corresponding to $\mathrm{Tr}(O\gamma_i^{(t,w)})$ is obtained after K measurements. In conjunction with above observations and the analytic gradients in Eq. (4), the explicit form of the estimated gradients for the j-th component with $\forall j \in [d_Q]$ satisfies

$$g_{i,j}(\boldsymbol{\theta}_i^{(t,w)}, \boldsymbol{x}_i^{(t,w)}) = (\bar{y}_i^{(t,w)} - y_i^{(t)})\frac{\bar{y}_i^{(t,w,+_j)} - \bar{y}_i^{(t,w,-_j)}}{2} + \lambda\boldsymbol{\theta}_{i,j}^{(t,w)}, \tag{6}$$

where $\bar{y}_i^{(t,w,\pm_j)} = \frac{1}{K}\sum_{k=1}^{K} V_k^{(\pm_j)}$ are estimated outputs with shifted parameters, i.e., $V_k^{(\pm_j)} \sim \mathrm{Ber}(\mathrm{Tr}(O\gamma_i^{(\pm_j)}))$ and $\gamma_i^{(\pm_j)} = (1-\tilde{p})U(\boldsymbol{\theta}^{(t,w,\pm_j)})\rho_i U(\boldsymbol{\theta}^{(t,w,\pm_j)})^\dagger + \tilde{p}\frac{\mathbb{I}}{2^N}$.

We quantify the convergence of QUDIO by the utility $R_1 = \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla\mathcal{L}(\boldsymbol{\theta}^{(t)})\|^2]$, where the expectation is taken over the randomness of data sample, the imperfection of quantum system, and the finite quantum measurements. Intuitively, R_1 evaluates how far QNN is away to a stationary point, which is a standard measure in non-convex optimization theory Jain & Kar (2017); Sun (2019). The following theorem summarizes the convergence of QUDIO whose proof is provided in Appendix B.

Theorem 1. *Assume that the discrepancy between the collected local gradients and the analytic gradients is bounded, i.e.,* $\forall i \in [Q]$ *and* $\forall\boldsymbol{\theta} \in [0, 2\pi)^{d_Q}$*, there exists*

$$\mathbb{E}_{\boldsymbol{x}_i^{(t,w)} \sim \mathcal{D}_i}\left[\left\|g_i^{(t,w)} - \nabla\bar{\mathcal{L}}\left(\boldsymbol{\theta}_i^{(t,w)}, \boldsymbol{x}_i^{(t,w)}\right)\right\|^2\right] \leq \sigma. \tag{7}$$

Following notations above, when the system noise is modeled by the depolarization channel $\mathcal{N}_p$ *and the measurement shot is* K*, the convergence rate of QUDIO yields*

$$R_1 \leq O\left(\lambda d_Q\sqrt{\frac{S}{T}} + \sqrt{\frac{S}{T}}\left(4W^2\sigma^2 + 2W^2G^2\right) + C_1\right),$$

where $C_1 \sim O(W^2(\tilde{p}d_Q + \tilde{p}d_Q/K))$*,* $\tilde{p} = (1-(1-p)^{L_Q})$*,* L_Q *is the total circuit depth, and* S (G) *is the smooth (Lipschitz) constant of* $\mathcal{L}$*.*

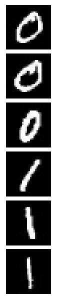
The results of Theorem 1 deliver three-fold implications. First, large system noise and few number of measurements may induce the optimization of QUDIO to be divergent, since the term C_1 is independent with T and is amplified by p and $1/K$. This observation hints the importance of integrating error mitigation techniques Cai et al. (2020); Du et al. (2020b); McClean et al. (2020); Strikis et al. (2020) into QUDIO to enhance its trainability. Second, in the NISQ scenario, reducing the iteration number of local updating W suggests a better performance, because C_1 is proportional to W. This phenomenon is starkly contrast with classical distributed optimization methods, which adopt large W to alleviate the communication overhead. Last but not least, under the ideal setting, the convergence rate between conventional QNNs and QUDIO is identical, i.e., both of them scale with $O(1/\sqrt{T})$ with respect to the step number T Du et al. (2020a). Celebrated by the joint optimization strategy, the similar convergence rate warrants that QUDIO promises a linear runtime speedup with respect to the increased number of local nodes Q.

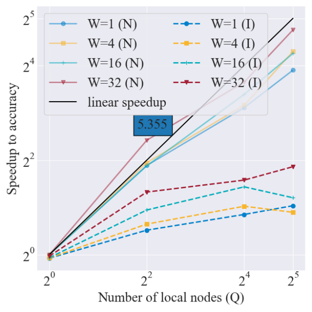
Remark. We emphasize that the developed tools in the proof of Theorem 1 can be easily extended to analyze the convergence of QUDIO-based QNNs with other loss functions, quantum noisy models, and optimizers. Moreover, the result of Theorem 1 indicates that naively imitating classical distributed algorithms to design distributed VQAs is *suboptimal*, since the inevitable biased gradient information in the quantum scenario may incur a deficient convergence. This suggests us to devise novel quantum distributed algorithms with an improved convergence rate.

5 Numerical simulations

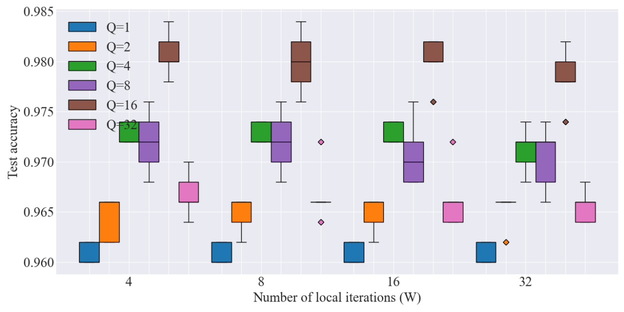
5.1 Experimental setup

Dataset. We carry out numerical simulations to exhibit how QUDIO accelerates QNNs when dealing with a standard binary classification task with a large size of training examples. More precisely,


(a)



(b)



(c)

Figure 2: **Simulation results of QUDIO towards hand-written digits image classification.** (a) A visualization of some training examples sampled from the MNIST dataset. (b) Scaling behavior of QUDIO in clock-time for increasing number of local nodes Q for varied number of local steps W. The labels '$W = a$ (I)' and '$W = a$ (N)' refer that the total number of local iterations is $W = a$ under the ideal and NISQ scenarios respectively. The hyper-parameters settings for the NISQ case are $p = 10^{-5}$ and $K = 100$. (c) The achieved test accuracy of QUDIO with varied W and Q in the NISQ scenario, where the hyper-parameters settings are same with those described in (b).

we sample 756 digit images labeled with '0' and '1' from the MNIST handwritten digit database LeCun (1998), where 256 images (examples) compose the training set and the rest 500 images (examples) form the test dataset. Fig. 2(a) visualizes some examples in the distilled dataset. Once the distillation is completed, the data preprocessing is applied, i.e., all examples are down-sampled to 8×8 pixels followed by the vectorization and l_2 normalization. In other words, each example corresponds to a 64-dimensional vector.

Implementation of QUDIO-based QNNs. When applying QUDIO-based QNNs to learn the classification rule toward the training set established above, i.e., $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$ with $n = 256$ and $\boldsymbol{x}_i \in \mathbb{R}^{64}$, the central server partitions it into $\{\mathcal{D}_i\}_{i=1}^Q$ and allocates them to Q local nodes. For example, when $Q = 32$, we have $|\mathcal{D}_i| = 8$ for $\forall i \in [32]$. Due to the identical implementation of all local nodes, here we only state the setup of the i-th local node $\mathcal{Q}_i$ for $\forall i \in [Q]$. The QNN relation to $\mathcal{Q}_i$ can be formulated as $h(\boldsymbol{\theta}, O, \rho_i) = \mathrm{Tr}(OU(\boldsymbol{\theta})\rho_i U(\boldsymbol{\theta})^\dagger)$ defined in Eq. (1). Particularly, the amplitude encoding method is adopted to encode $\boldsymbol{x}_i \in \mathcal{D}_i$ to the quantum state ρ_i, i.e., $\rho_i = |\boldsymbol{x}_i\rangle\langle\boldsymbol{x}_i|$, and $|\boldsymbol{x}_i\rangle = \sum_{j=1}^{64} \boldsymbol{x}_{i,j}|j\rangle$. In this way, the required number of qubits to implement QNN is $N = 8$. The prepared state ρ_i is interacted with the hardware-efficient ansatz $U(\boldsymbol{\theta}) = \prod_{l=1}^L U_l(\boldsymbol{\theta})$ Kandala et al. (2017). The block number is set as $L = 4$. The operator O is set as $O = \mathbb{I}_{2^5} \otimes |0\rangle\langle 0|$. Refer to Appendix C for implementation details.

Hyper-parameters setting. The number of local nodes Q ranges from 1 to 32. The number of total local iterations has six settings, i.e., $W \in [1, 2, 4, 8, 16, 32]$. In the NISQ case, we set $K \in \{5, 20, 50, 100\}$ and $p \in [10^{-4}, 10^{-1}]$. Each setting is repeated with 5 times to collect the statistical results. For the stochastic gradient descent optimizer, its initial learning rate is set as 0.01 and the decay-rate is set as 0.1 every 40 global iterations.

The source code of QUDIO. The realization of QUDIO is based on Pytorch Paszke et al. (2019) and its distributed communication package. To be more concrete, we select the GLOO backend and ring all-reduce operations to achieve the communication protocol between processes on CPU. Note that this distributed optimization framework can be easily extended to coordinate multiple quantum processors. All simulation results in this study are completed by the classical device with Intel(R) Xeon(R) Gold 6267C CPU @ 2.60GHz and 128 GB memory. We release our code to the repository `https://anonymous.4open.science/r/QUDIO-1076/`.

Metric. To better quantify the performance of QUDIO from different angles, we introduce two metrics, i.e., the *speedup to accuracy* and the *test accuracy*, to evaluate the achieved results. Namely, the former considers the speedup ratio of QUDIO compared with the setting of $Q = 1$ (original QNN), i.e., supposes that the train accuracy reaches a threshold (e.g., 95%) in T_1 (T_2) clock-time for $Q = 1$ ($Q = a$), the speedup to accuracy is evaluated by $T1/T2$. The latter intends to compare the top test accuracy of QUDIO within a fixed number of global steps T with varied Q and W.

5.2 Experimental results

Our main experimental results are exhibited in Fig. 2. As shown in Fig. 2 (b), under the measure of speedup to accuracy, QUDIO gains the acceleration for both ideal and NISQ scenarios by increasing the number of local nodes Q. Strikingly, QUDIO can even reach a superlinear speedup in the NISQ scenario when $W = 32$, e.g., it achieves 5.355 times speedup when $Q = 4$. This phenomenon accords with our theoretical claim such that QUDIO is insensitive to the communication bottleneck, which differs from distributed DNNs Dean et al. (2012). Moreover, the distinct scaling behavior of QUDIO between the ideal and the NISQ cases is mainly caused by the fact that the evaluation of the analytic gradients in the ideal case is extremely fast and the communication cost gradually dominates the clock time when Q becomes large. The box plot in Fig. 2(c) pictures the achieved results of QUDIO in the measure of test accuracy. For all settings of Q, an increased W generally degrades the performance of QUDIO from a statistical view. These results partially echo with Theorem 1 such that larger W suggests worse performance. An evidence is when $Q = 16$, QUDIO achieves the best test accuracy when $W = 4$. Envisioned by the above observations, we next comprehensively investigate what factors dominate the capability of QUDIO.

Speedup analysis. We adopt the following settings to explore the speedup ratio of QUDIO. Concretely, both the number of local nodes and the number of local iterations have six varied settings, i.e., $Q, W \in \{1, 2, 4, 8, 16, 32\}$. In the NISQ scenario, the depolarization rate and the number of measurements are set as $p = 10^{-4}$ and $K = 100$, respectively. Fig. 3 depicts the speedup ratio of QUDIO in the measure of time to accuracy. In particular, QUDIO attains a sublinear acceleration in terms of Q. Besides, a larger number of local iterations W generally promises a higher speedup ratio. We also notice that there exists a manifest margin between the noiseless and NISQ settings. This phenomenon is mainly caused by the opposite role of the communication overhead, i.e., the communication overhead occupies a large portion of the computational runtime in the noiseless case, while it becomes negligible in the NISQ case. Tab. 1 records the concrete clock time under each setting.

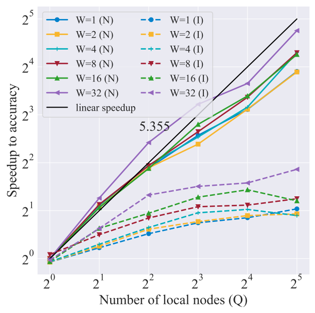


Figure 3: **The speedup ratio of QUDIO in the measure of the time to accuracy.** The labels 'N' and 'I' represent the noisy and ideal scenarios, respectively.

Table 1: The runtime of QUDIO with respect to the varied number of Q and W.

Number of local nodes (Q)	1	2	4	8	16	32
Ideal: Time to accuracy (s)	104.53	66.20	40.81	36.02	34.25	28.14
NISQ: Time to accuracy (s)	2518.46	1049.63	470.25	270.47	200.10	93.16
Number of local steps (W)	1	2	4	8	16	32
Ideal: Time to accuracy (s)	54.37	55.85	52.24	45.67	43.62	28.14
NISQ: Time to accuracy (s)	169.99	155.37	117.18	115.20	117.73	93.16

Accuracy analysis. We next turn to explore how the factors Q and W influence the final test accuracy of QNN. Analogous to the speedup analysis, the number of local nodes and local iterations have six settings, i.e., $Q, W \in \{1, 2, 4, 8, 16, 32\}$. Meanwhile, the depolarizing rate and the number of measurement are set as $p = 0.0001$ and $K = 100$ respectively. As shown in Fig. 4, for all settings, the test accuracy achieved by QUDIO is above 95%. This observation reflects the robustness of QUDIO. Moreover, when the number of local iterations W is kept to be identical, QUDIO gains the highest test accuracy with the setting $Q = 16$. On the contrary, when the number of local nodes Q is identical (box with the same color), increasing the local iterations W results in a slightly deteriorate test accuracy and a large variance. These phenomena compile with the claim of Theorem 1 and suggest that the performance of QUDIO highly depends on the number of local nodes W.

The role of the system noise and the number of measurements. We last examine how the factors of system noise p and the number of measurements K influence the performance of QUDIO. In particular, we evaluate the test accuracy of QUDIO by scaling the depolarization rate p from 0.0001

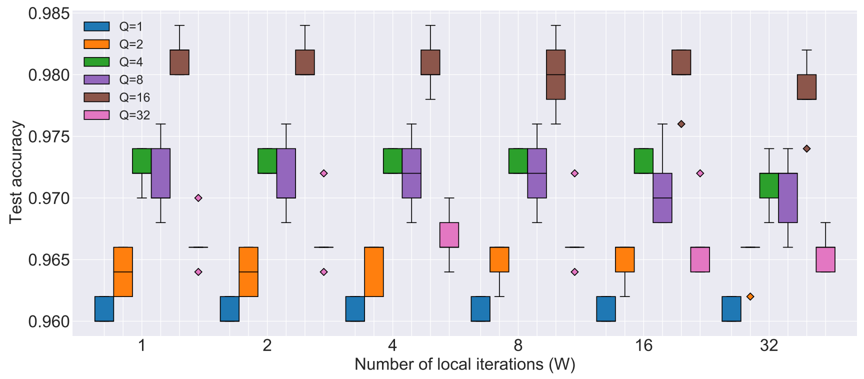


Figure 4: **Role of number of quantum processors and local iterations.**

to 0.0512 and ranging the number of measurements from 5 to 100. The number of local nodes and local iterations is fixed to be $Q = 16$ and $W = 2$, respectively.

Fig. 5 summarizes the exprimental results. In particular, when $p < 0.0064$, the performance of QUDIO heavily depends on the number of measurements. For example, the test accuracy is around 98% with $M = 100$, while it drops to 87% with $M = 5$. When $p > 0.0064$, both p and K determine the performance of QUDIO. For example, for the setting $K = 5$, the test accuracy of QUDIO with $p = 0.0512$ is reduced by 14% than the setting $p = 0.0256$ (i.e., from 77% to 66%). This observation echoes with our theoretical analysis such that integrating error mitigation techniques Cai et al. (2020); Du et al. (2020b); McClean et al. (2020); Strikis et al. (2020) into QUDIO is crucial to enhance its trainability.

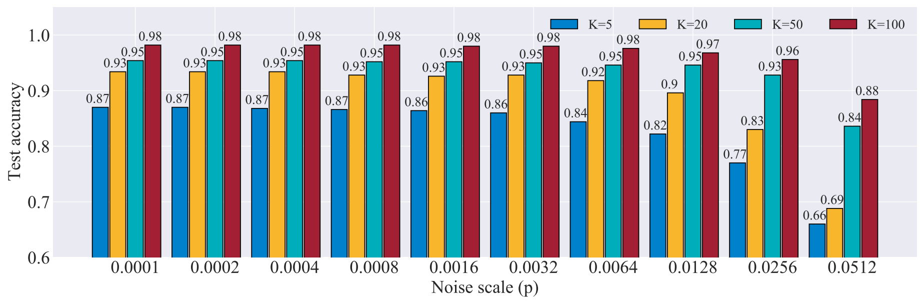


Figure 5: **The test accuracy of QUDIO with varied depolarization rate and the number of measurements.** The label '$K = a$' refers that the number of measurements K is set as a.

6 Discussion and conclusion

In this study, we devise QUDIO to accelerate VQAs with multiple quantum processors. We also provide theoretical analysis about how the system noise and the number of measurements influence the convergence of QUDIO. An attractive feature is that in the ideal setting, QUDIO obeys the asymptotic convergence rate with conventional QNNs, which ensures its runtime speedup with respect to the increased number of local nodes. The achieved numerical simulation results confirm the effectiveness of our proposal. Particularly, in the NISQ scenario, QUDIO can achieve superline speedups in the measure of time-to-accuracy. For these reasons, QUDIO and its variants, which marry the distributed techniques with VQAs, could substantially contribute to use NISQ machines to accomplish real-world problems with quantum advantages. Furthermore, QUDIO is highly compatible and can be seamlessly embedded into the cloud computing, since it supports various types of quantum processors to set up local nodes and its central server is purely classical.

REFERENCES

Amira Abbas, David Sutter, Christa Zoufal, Aurélien Lucchi, Alessio Figalli, and Stefan Woerner. The power of quantum neural networks. *arXiv preprint arXiv:2011.00027*, 2020.

Javier Alcazar, Vicente Leyton-Ortega, and Alejandro Perdomo-Ortiz. Classical versus quantum models in machine learning: insights from a finance application. *Machine Learning: Science and Technology*, 1(3):035003, 2020.

Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

Robert Beals, Stephen Brierley, Oliver Gray, Aram W Harrow, Samuel Kutin, Noah Linden, Dan Shepherd, and Mark Stather. Efficient distributed quantum computing. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 469(2153):20120686, 2013.

Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, M Sohaib Alam, Shahnawaz Ahmed, Juan Miguel Arrazola, Carsten Blank, Alain Delgado, Soran Jahangiri, et al. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint arXiv:1811.04968*, 2018.

Kishor Bharti, Alba Cervera-Lierta, Thi Ha Kyaw, Tobias Haug, Sumner Alperin-Lea, Abhinav Anand, Matthias Degroote, Hermanni Heimonen, Jakob S Kottmann, Tim Menke, et al. Noisy intermediate-scale quantum (nisq) algorithms. *arXiv preprint arXiv:2101.08448*, 2021.

Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

Sergey B Bravyi and Alexei Yu Kitaev. Fermionic quantum computation. *Annals of Physics*, 298 (1):210–226, 2002.

Zhenyu Cai, Xiaosi Xu, and Simon C Benjamin. Mitigating coherent noise using pauli conjugation. *npj Quantum Information*, 6(1):1–9, 2020.

M Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, et al. Variational quantum algorithms. *arXiv preprint arXiv:2012.09265*, 2020.

Alba Cervera-Lierta, Jakob S. Kottmann, and Alán Aspuru-Guzik. Meta-variational quantum eigensolver: Learning energy profiles of parameterized hamiltonians for quantum simulation. *PRX Quantum*, 2:020329, May 2021. doi: 10.1103/PRXQuantum.2.020329. URL `https://link.aps.org/doi/10.1103/PRXQuantum.2.020329`.

Brian Coyle, Maxwell Henderson, Justin Chan Jin Le, Niraj Kumar, Marco Paini, and Elham Kashefi. Quantum versus classical generative modelling in finance. *Quantum Science and Technology*, 6(2):024013, 2021.

Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, and Andrew Y. Ng. Large scale distributed deep networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, pp. 1223–1231, Red Hook, NY, USA, 2012. Curran Associates Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019.

Yuxuan Du, Min-Hsiu Hsieh, Tongliang Liu, Shan You, and Dacheng Tao. On the learnability of quantum neural networks. *arXiv preprint arXiv:2007.12369*, 2020a.

Yuxuan Du, Min-Hsiu Hsieh, Tongliang Liu, Shan You, and Dacheng Tao. Quantum differentially private sparse regression learning. *arXiv preprint arXiv:2007.11921*, 2020b.

Héctor Abraham et. al. Qiskit: An open-source framework for quantum computing, 2019.

Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Stuart Hadfield, Zhihui Wang, Bryan O'Gorman, Eleanor G Rieffel, Davide Venturelli, and Rupak Biswas. From the quantum approximate optimization algorithm to a quantum alternating operator ansatz. *Algorithms*, 12(2):34, 2019.

Matthew P Harrigan, Kevin J Sung, Matthew Neeley, Kevin J Satzinger, Frank Arute, Kunal Arya, Juan Atalaya, Joseph C Bardin, Rami Barends, Sergio Boixo, et al. Quantum approximate optimization of non-planar graph problems on a planar superconducting processor. *Nature Physics*, 17(3):332–336, 2021.

Aram W Harrow and Ashley Montanaro. Quantum computational supremacy. *Nature*, 549(7671): 203, 2017.

Vojtěch Havlíček, Antonio D Córcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow, and Jay M Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209, 2019.

He-Liang Huang, Yuxuan Du, Ming Gong, Youwei Zhao, Yulin Wu, Chaoyue Wang, Shaowei Li, Futian Liang, Jin Lin, Yu Xu, et al. Experimental quantum generative adversarial networks for image generation. *arXiv preprint arXiv:2010.06201*, 2020.

Hsin-Yuan Huang, Michael Broughton, Masoud Mohseni, Ryan Babbush, Sergio Boixo, Hartmut Neven, and Jarrod R McClean. Power of data in quantum machine learning. *Nature communications*, 12(1):1–9, 2021a.

Hsin-Yuan Huang, Richard Kueng, and John Preskill. Information-theoretic bounds on quantum advantage in machine learning. *Phys. Rev. Lett.*, 126:190505, May 2021b. doi: 10.1103/PhysRevLett.126.190505. URL `https://link.aps.org/doi/10.1103/PhysRevLett.126.190505`.

Prateek Jain and Purushottam Kar. Non-convex optimization for machine learning. *Foundations and Trends® in Machine Learning*, 10(3-4):142–336, 2017.

Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M Chow, and Jay M Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549(7671):242–246, 2017.

Ryan LaRose and Brian Coyle. Robust data encodings for quantum classifiers. *Physical Review A*, 102(3):032420, 2020.

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Jarrod R McClean, Zhang Jiang, Nicholas C Rubin, Ryan Babbush, and Hartmut Neven. Decoding quantum errors with subspace expansions. *Nature communications*, 11(1):1–9, 2020.

Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa, and Keisuke Fujii. Quantum circuit learning. *Physical Review A*, 98(3):032309, 2018.

Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*. Cambridge University Press, 2010.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, 5:4213, 2014.

John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

Yang Qian, Xinbiao Wang, Yuxuan Du, Xingyao Wu, and Dacheng Tao. The dilemma of quantum neural networks. *arXiv preprint arXiv:2106.04975*, 2021.

Manuel S Rudolph, Ntwali Toussaint Bashige, Amara Katabarwa, Sonika Johr, Borja Peropadre, and Alejandro Perdomo-Ortiz. Generation of high resolution handwritten digits with an ion-trap quantum computer. *arXiv preprint arXiv:2012.03924*, 2020.

Maria Schuld and Nathan Killoran. Quantum machine learning in feature hilbert spaces. *Physical review letters*, 122(4):040504, 2019.

Andrew W Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander WR Nelson, Alex Bridgland, et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020.

Armands Strikis, Dayue Qin, Yanzhu Chen, Simon C Benjamin, and Ying Li. Learning-based quantum error mitigation. *arXiv preprint arXiv:2005.07601*, 2020.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3645–3650, 2019.

Ruoyu Sun. Optimization for deep learning: theory and algorithms. *arXiv preprint arXiv:1912.08957*, 2019.

Ho Lun Tang, VO Shkolnikov, George S Barron, Harper R Grimsley, Nicholas J Mayhall, Edwin Barnes, and Sophia E Economou. qubit-adapt-vqe: An adaptive algorithm for constructing hardware-efficient ansätze on a quantum processor. *PRX Quantum*, 2(2):020310, 2021.

Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38, 2019.

Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D Lukin. Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Physical Review X*, 10(2):021067, 2020.