Quantum autoencoders for communication-efficient cloud computing

Citation: Quantum Machine Intelligence. 2023 Jul 10;5(2):27

Published Version: https://doi.org/10.1007/s42484-023-00112-5

Publisher: Springer International Publishing

Permanent Link: https://hdl.handle.net/1721.1/151086

Version: Author's final manuscript: final author’s manuscript post peer review, without publisher’s formatting or copy editing

Quantum autoencoders for communication-efficient cloud computing

Quantum autoencoders for communication-efficient quantum cloud computing

Yan Zhu[1], Ge Bai[1], Yuexuan Wang[2,9], Tongyang Li[3,4,5*] and Giulio Chiribella[1,6,7,8*]

[1]QICI Quantum Information and Computation Initiative, Department of Computer Science, The University of Hong Kong, Pokfulam Road, Hong Kong.
[2]AI Technology Lab Department of Computer Science, The University of Hong Kong, Pokfulam Road, Hong Kong.
[3]Center on Frontiers of Computing Studies, Peking University.
[4]School of Computer Science, Peking University.
[5]Center for Theoretical Physics, Massachusetts Institute of Technology.
[6]Department of Computer Science, University of Oxford, Parks Road, Oxford OX1 3QD, United Kingdom.
[7] Perimeter Institute For Theoretical Physics, 31 Caroline Street North, Waterloo N2L 2Y5, Ontario, Canada.
[8]The University of Hong Kong Shenzhen Institute of Research and Innovation, Yuexing 2nd Rd Nanshan, Shenzhen, China.
[9]College of Computer Science and Technology, Zhejiang University, China.

*Corresponding author(s). E-mail(s): tongyangli@pku.edu.cn; giulio@cs.hku.hk;

Abstract

In the model of quantum cloud computing, the server executes a computation on the quantum data provided by the client. In this scenario, it is important to reduce the amount of quantum communication between the client and the server. A possible approach is to transform the desired

computation into a compressed version that acts on a smaller number of qubits, thereby reducing the amount of data exchanged between the client and the server. Here we propose quantum autoencoders for quantum gates (QAEGate) as a method for compressing quantum computations. We illustrate it in concrete scenarios of single-round and multi-round communication and validate it through numerical experiments. A bonus of our method is it does not reveal any information about the server's computation other than the information present in the output.

Keywords: Quantum autoencoders, Quantum cloud computing, Quantum gate

1 Introduction

Over the past decade, quantum computing technology underwent a rapid series of advances. Both the size and power of quantum computers have been steadily increasing over the years, recently entering a new regime of "quantum supremacy" [1], in which the output of the quantum computations can be barely reproduced by the world's best supercomputers. Milestone achievements are the "quantum supremacy" demonstrations by Google [2] and USTC [3], using superconducting qubits and photonic qubits, respectively.

A promising direction in quantum computing is the study of cloud computing scenarios, where a client requests a remote server to perform some desired quantum operations [4–6]. However, limits on the amount of quantum communication between client and server constraint the size of the computations that can be effectively implemented in quantum cloud computing.

In this paper, we address the problem of reducing the amount of communication needed by the server to perform a quantum operation on quantum data provided by the client. The operation belongs to a parametric family of quantum gates known to both parties, but the specifications of the gate are known only to the server. Our objective here is to maximize the accuracy of the executed quantum operation when the capacity of the communication link and the total number of qubits exchanged between the server and the client are limited.

Our main contribution is a method for compressing a parametric family of quantum gates, turning it into another gate family acting on a smaller number of qubits. Our method is based on autoencoders, a type of neural networks that have been very successful in classical machine learning [7], and have been recently used for the compression of quantum states [8]. We introduce a quantum gate autoencoder (QAEGate), providing the first quantum machine learning model that takes gates, rather than states, as inputs.

Our method also addresses the problem of minimizing the amount of information revealed to the client. In many situations, the server would not like

to disclose any more information about the operation other than the unavoidable information revealed by the application of the operation on the client's input. Our method achieves this feature by constructing a blind compression protocol, independent of the specifications of the operation.

Technically, the training of our QAEGate model is based on stochastic gradient descent [9]. We prove that the training is guaranteed to converge in polynomial time in the size of the initial operation. We then conduct numerical experiments that show the effectiveness of QAEGate in various settings including single-round and multi-round communication between the client and the server.

The remainder of the paper is organized as follows. In Section 2, we introduce some related works about quantum cloud computing and quantum autoencoders. Some preliminaries on autoencoders and quantum supermaps are provided in Section 3. Section 4 is devoted to introduce the quantum cloud computing task discussed in this paper. The structure and training details of our proposed QAEGate model are covered in Section 5. Then, we explain how to apply our method to concrete scenarios of quantum cloud computing in Section 6 and conduct some numerical experiments in Section 7. We conclude this paper in Section 8, with discussions on future directions for our method.

2 Related work

Classical cloud computing involves the execution of a computation on a remote server [10]. In the quantum version of this scenario [11], a client provides an input state and asks a remote server to perform a sequence of quantum gates on it. Designing practical quantum cloud computing protocols involves addressing two important issues. The first issue is how to cope with the limited amount of quantum communication available in realistic scenarios. Yang et al. [12] addressed this by deriving a general lower bound on the minimum amount of communication needed to specify the desired computation using theoretical techniques from quantum Shannon theory. However, their results are limited to scenarios where the server approximately implements the desired process for $n \geq 1$ times in parallel on n identical systems, rather than the general case. The second issue is how to protect data privacy throughout the protocol. Sheng et al. [13] proposed a model to prevent interception and disturbance in distributed quantum machine learning. Related issues have also been considered in blind delegated quantum computation [4], where the authors introduced a protocol to allow a client to safely have a server execute a desired quantum computation. However, previous works have mainly focused on guaranteeing data confidentiality on the client side rather than the server side. This leaves room for further research on how to ensure data privacy and security from the server's perspective.

Quantum autoencoders for quantum states were proposed by [8] and has been applied to quantum state compression and denoising quantum data [14, 15]. The quantum model used in these works has a similar structure

to classical autoencoders, taking as input a quantum state represented by a fixed-length vector. This model, however, cannot be extended from quantum states to quantum gates, which in general are provided as black boxes. To convert a gate into a quantum state, one would have to apply it on a fixed input state. However, such conversion is not reversible due to the quantum No-programming theorem [16, 17].

3 Preliminaries

Autoencoders & Quantum autoencoders.

An autoencoder is composed of an encoder and a decoder. One of its main application is dimensionality reduction, where the encoder maps a high-dimensional vector input x to a low-dimensional representation h, and the decoder maps h back to a reconstructed high-dimensional vector x'. In the training process, the autoencoder is optimized so that the high-dimensional vector x' is as close as possible to the original high-dimensional vector x. If x' and x are close enough, the overall protocol provides a faithful compression of the original input x into the low dimensional vector h. The structure of a typical autoencoder is depicted in Figure 1.

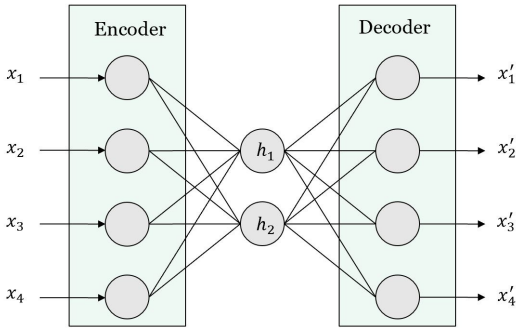

Fig. 1 Structure of a typical autoencoder. In this specific example, the autoencoder compresses a 4-dimensional vector into a 2-dimensional vector.

Building upon the success of classical autoencoders, researchers in [8] introduced the concept of quantum autoencoders. A quantum autoencoder also consists of an encoder and a decoder, both typically implemented using parameterized quantum circuits. The typical structure of a quantum autoencoder is shown in Figure 2, where the encoder converts an n-qubit input quantum state ρ into an $(n-a)$-qubit quantum state, while the decoder converts the encoded state back to an n-qubit quantum state ρ'. The training objective of the quantum autoencoder is to minimize the difference between the original and reconstructed quantum states, typically measured using a quantum fidelity or distance metric. To train a quantum autoencoder, a quantum-classical hybrid

scheme is typically used. In this scheme, the variational quantum circuits and measurements are performed on the quantum computer, while optimization is carried out via a classical optimization algorithm.

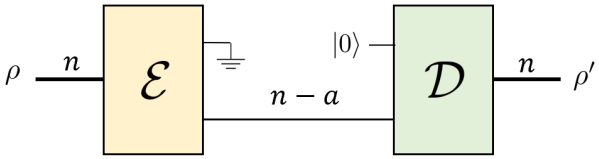

Fig. 2 Structure of a typical quantum autoencoder. In this specific example, the quantum autoencoder compresses a n-qubit quantum state into a $(n-a)$-qubit state.

The Choi state & Overlap

A $d \times d$ matrix M can be viewed as a column vector in a d^2-dimensional space, using the correspondence $M \mapsto |M\rangle\!\rangle := \sum_{i,j} M_{i,j}|i\rangle|j\rangle$. This correspondence can be lifted to a one-to-one correspondence between quantum gates in dimension d, and quantum states in dimension d^2 [18, 19]. Explicitly, a quantum gate U is associated to the Choi state

$$C_U := \frac{|U\rangle\!\rangle \langle\!\langle U|}{d}. \tag{1}$$

A convenient measure of closeness for two quantum states ρ and σ is their overlap

$$f(\rho, \sigma) = \mathrm{Tr}(\rho\sigma)\,.$$

Therefore, using the overlap between the Choi states of two quantum gates as a similarity measure is a natural choice.

Quantum supermaps.

A general quantum operation is represented by a quantum channel $\mathcal{C}$, which is a map from density operators to density operators [20]. A quantum supermap [21, 22] $\tilde{\mathcal{S}}$ maps a quantum channel $\mathcal{C}$ into a quantum channel $\mathcal{C}'$ as $\mathcal{C}' = \tilde{\mathcal{S}}(\mathcal{C})$. Every quantum supermap can be represented by a quantum circuit in Figure 3 [21], where the input quantum operation $\mathcal{C}$ sends states in $\mathcal{H}_{in}$ to states in $\mathcal{H}_{out}$ and the output quantum operation $\mathcal{C}'$ sends states on $\mathcal{K}_{in}$ to states in $\mathcal{K}_{out}$. The supermap is realized by two maps $\mathcal{V}$ and $\mathcal{W}$ located at the input and at the output ports of the quantum operation $\mathcal{C}$ respectively. Quantum supermaps are closely related to quantum cloud computing, as they can facilitate efficient implementation of quantum computations on remote quantum servers. Specifically, a quantum supermap can be used to map the quantum computation required by the client to the action of a quantum computation

carried out by the server. This allows for delegated quantum computing, where a client can delegate a quantum computation to a remote server. Therefore, quantum supermaps have great potential to enable practical applications of quantum cloud computing, including quantum machine learning and quantum cryptography.

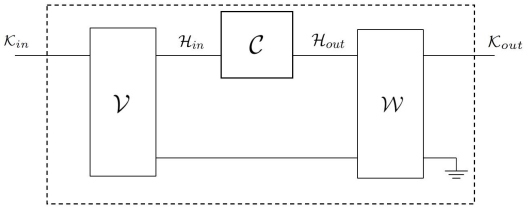

Fig. 3 Quantum supermap.

4 The Quantum Cloud Computing Task

In this paper we will focus on a basic scenario of quantum cloud computing, depicted in Figure 4. A parametric family of n-qubit quantum gates $\{U_x\}_{x\in\mathsf{X}}$ with parameter x represents a set of possible quantum computations. A server is able to implement a gate U_x, unknown to the client. The client chooses an n-qubit input state ρ and asks the server to apply the gate U_x on it, thus obtaining the state $\mathcal{U}_x(\rho) := U_x\rho U_x^\dagger$. At the same time, the server wants to keep x confidential, avoiding unnecessary information leaked to the client during the communication.

A trivial way to achieve the above task is to have the sender send the quantum state ρ to the server, who performs the required gate, and sends back the output state $\mathcal{U}_x(\rho)$. However, the quantum communication is an expensive resource, and it is often limited in realistic applications, making it difficult to transmit a full n-qubit quantum state back and forth between the client and the server. Here we consider the scenario where the capacity of the quantum communication link is bounded by $a \leq n$ qubits.

In order to cope with the bottleneck on the amount of quantum communication, we design a pair of quantum circuits, including an encoder $\mathcal{E}$ implemented by the server and a decoder $\mathcal{D}$ implemented by the client. The encoder $\mathcal{E}$ serves as a quantum supermap transforming an n-qubit quantum channel to an a-qubit quantum channel, while the decoder $\mathcal{D}$ does the opposite, recovering the n-qubit quantum channel from the output of the encoder. Since the decoder is implemented by the client, who has no knowledge of the parameter x, the supermap $\mathcal{D}$ must be independent of x. On the other hand, the encoder is implemented by the server, and could in principle depend on x. However, a dependence on x may result into a leakage of information to the client. For this reason, we also require the encoder $\mathcal{E}$ to be independent of x.

We will later show that this choice reaches the best possible confidentiality in the ideal situation where U_x is accurately implemented.

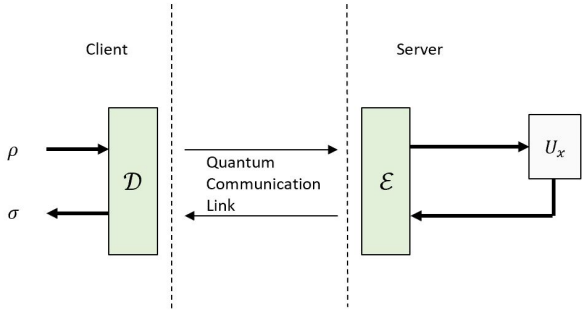

Fig. 4 The basic scenario of quantum cloud computing. The client expects to apply a quantum gate U_x on its own state ρ by communicating with a server over a communication link of limited capacity.

5 Quantum Autoencoders for Quantum Gates

In this section we consider a basic scenario of quantum cloud computing with one round of communication between client and server. In this scenario, we develop a model of quantum autoencoders for quantum gates (QAEGate). In the following, we will introduce the structure of the proposed model and demonstrate its implementation and training through numerical experiments. Furthermore, we prove that stochastic gradient descent has convergence guarantee in the training of our QAEGate model. We will extend our model to more scenarios in Section 6.

5.1 Structure

Our model consists of an encoder and a decoder. The encoder produces an encoded quantum gate by inserting the gate U_x into a suitable quantum circuit, initializing $n-a$ input qubits to a fixed state $|0\rangle$, and discarding $n-a$ output qubits. The result is a (generally noisy) quantum channel acting on a qubits. In turn, the decoder converts the a-qubit channel back into an n-qubit channel, which aims to approximate the initial gate. The structure of the encoder and the decoder are shown in Figure 5.

To construct the encoding and decoding circuits we use parameterized unitary operators, which have been successfully employed to build variational quantum circuits [23, 24]. Our construction is depicted in Figure 5. The encoder consists of two parameterized unitary operators, denoted by $U^{\boldsymbol{\theta}_{\mathbf{le}}}$ and $U^{\boldsymbol{\theta}_{\mathbf{re}}}$, depending on parameters $\boldsymbol{\theta}_{\mathbf{le}} = (\theta_{le_1}, \theta_{le_2}, ..., \theta_{le_m})$ and $\boldsymbol{\theta}_{\mathbf{re}} = (\theta_{re_1}, \theta_{re_2}, ..., \theta_{re_m})$. The gates $U^{\boldsymbol{\theta}_{\mathbf{le}}}$ and $U^{\boldsymbol{\theta}_{\mathbf{re}}}$ are placed on the left and right of the original quantum gate, respectively. The decoder consists of two unitary operators $U^{\boldsymbol{\theta}_{\mathbf{ld}}}$ and $U^{\boldsymbol{\theta}_{\mathbf{rd}}}$, depending on parameters $\boldsymbol{\theta}_{\mathbf{ld}} = (\theta_{ld_1}, \theta_{ld_2}, ..., \theta_{ld_m})$

and $\boldsymbol{\theta_{rd}} = (\theta_{rd_1}, \theta_{rd_2}, ..., \theta_{rd_m})$. The gates $U^{\boldsymbol{\theta_{ld}}}$ and $U^{\boldsymbol{\theta_{rd}}}$ are placed on the left and right of the encoded gate, combined with the identity gate on the remaining $n - a$ qubits. Note that these parameterized unitary operators are independent of the parameter x of the original quantum gate here.

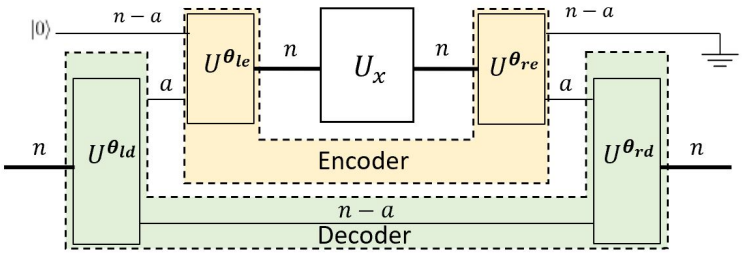

Fig. 5 Implementation details of QAEGate. The parameterized unitary operators in yellow constitute the encoder while the parameterized unitary operators in green constitute the decoder. In the encoding phase, we insert the gate U_x in the middle of the encoder, initialize $n - a$ input qubits to a fixed state $|0\rangle$, and discard $n - a$ output qubits to obtain a quantum channel acting on a qubits. In the decoding phase, the decoder maps the channel back to an approximation of the initial n-qubit quantum gate.

The choice of the parameters $\boldsymbol{\theta_{le}}$, $\boldsymbol{\theta_{re}}$, $\boldsymbol{\theta_{ld}}$, and $\boldsymbol{\theta_{rd}}$ is optimized in order to maximize the similarity between the decoded quantum gate and the original quantum gate. As a similarity measure, we use the overlap between the Choi operators [18, 19], due to the relative ease of evaluating this quantity in numerical experiments. As the optimization procedure, we will use a stochastic gradient descent method, described in Subsection 5.2.

To run the optimization, one needs first to fix the parametrization of the gates. The naive choice would be to pick a parametrization that can represent arbitrary quantum gates. However, this approach has obvious difficulties:

- Describing an arbitrary n-qubit unitary transformation requires an exponential number of parameters, and therefore a full optimization of the parameters in the autoencoder is only feasible for small values of n.
- Due to hardware restrictions of the current quantum computers, it is difficult to implement arbitrary n-qubit unitary operators. This is especially true for the client, who may only have the ability to implement a small number of basic quantum gates.

To address these problems, we propose an implementation of QAEGate based on an approximation of generic n-qubit unitary transformations that uses a polynomial number of parameters. We implement the parameterized unitary operators for n qubits by decomposing them into a sequence of parameterized two-qubit unitary operators, where each pair of qubits corresponds to one such operator. Figure 6 presents the case when $n = 4$. For each two-qubit

parameterized unitary operator, it is further decomposed into a sequence of basic quantum gates, as depicted in Figure 7.

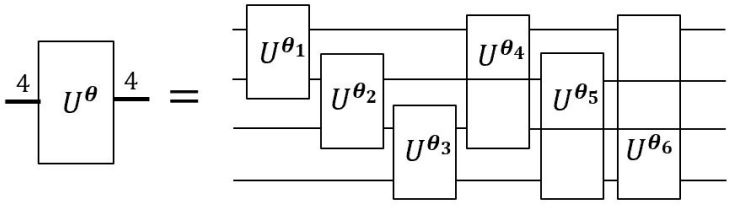

Fig. 6 Decomposition of a parameterized 4-qubit unitary gate into 6 parameterized two-qubit gates.

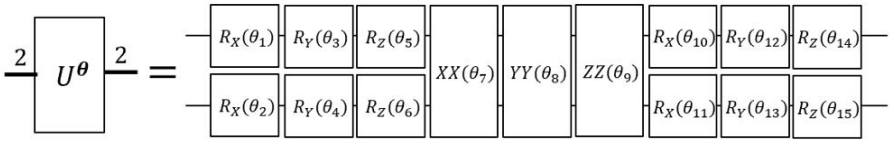

Fig. 7 Decomposition of a two-qubit parameterized unitary gate.

Here $R_X(\theta)$ is the quantum gate defined as follows.

$$R_X(\theta) := \exp(i\theta\sigma_x) = \begin{pmatrix} \cos\theta & -i\sin\theta \\ -i\sin\theta & \cos\theta \end{pmatrix}$$

The other two gates $R_Y(\theta)$ and $R_Z(\theta)$ are similar to $R_X(\theta)$ but for Pauli matrices σ_y and σ_z.

The two-qubit gate $XX(\theta)$ is the Ising coupling gate that commonly appears in real-world quantum computers with trapped-ion architecture [25]. It is defined as $XX(\theta) := \exp(i\theta\sigma_x \otimes \sigma_x)$ for the Pauli matrix σ_x, and written in matrix form as:

$$XX(\theta) = \begin{bmatrix} \cos\theta & 0 & 0 & -i\sin\theta \\ 0 & \cos\theta & -i\sin\theta & 0 \\ 0 & -i\sin\theta & \cos\theta & 0 \\ -i\sin\theta & 0 & 0 & \cos\theta \end{bmatrix}.$$

$YY(\theta)$ and $ZZ(\theta)$ are Ising coupling gates corresponding to Pauli matrices σ_y and σ_z respectively.

5.2 Training

To find the optimal parameters of QAEGate, we adopt classical gradient-based methods such as stochastic gradient descent [26].

We denote the parameters of QAEGate to be trained as $\boldsymbol{\theta}$. The training set is composed of k quantum gates sampled randomly from $\{U_x\}$. In every iteration, we select a quantum gate U from these k gates randomly and insert it into the QAEGate. Then we calculate the overlap f between the Choi state C of U and the Choi state C' of the decoded gate from the QAEGate. We define the loss function as $\mathcal{L} = 1 - f$ here and update the parameters of QAEGate according to the gradients $\nabla_{\boldsymbol{\theta}}\mathcal{L}$ until the decoded quantum gate is close enough to the original quantum gate or the maximum number of iterations is reached. We describe the complete training process in Algorithm 1.

Algorithm 1 Quantum autoencoders for quantum gates (QAEGate).

Require: k quantum gates sampled randomly from $\{U_x\}$, maximum number of iterations K, learning rate η, threshold δ.

Initialize QAEGate parameters $\boldsymbol{\theta}$ randomly, $i = 0$, $f = 1$
while $i < K$ or $f > \delta$ **do**
  Select the quantum gate U randomly from k input gates
  Insert U into the QAEGate
  Calculate the Choi state C of U
  Get the decoded gate from the QAEGate
  Calculate the corresponding Choi state C'
  $f = \mathrm{Tr}(CC')$, which is the overlap between C and C'
  $\mathcal{L} = 1 - f$
  Calculate $\nabla_{\boldsymbol{\theta}}\mathcal{L}$ and update $\boldsymbol{\theta}$ as $\boldsymbol{\theta} = \boldsymbol{\theta} - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}$
  $i = i + 1$
end while

5.3 Convergence Analysis

We further prove that our proposed QAEGate can be efficiently trained by the stochastic gradient-based method in Algorithm 1. Specifically, we obtain the following convergence guarantee for n-qubit QAEGate model where the query complexity represents the number of evaluations of the quantum circuit:

Theorem 1 (Convergence guarantee) *If we perform SGD, as in Algorithm 1, to optimize $\boldsymbol{\theta}$ in the training of the QAEGate model, then the convergence rate is $T = \mathcal{O}(\frac{n^2}{\epsilon^4})$ for achieving the condition $\mathbb{E}[\|\nabla_{\boldsymbol{\theta}}\mathcal{L}\|^2] \leq \epsilon^2$. If we consider the complexity of estimating the gradients, then the query complexity is $\mathcal{O}(\frac{n^4}{\epsilon^4})$.*

The proof is provided in the Appendix. While the stochastic gradient-based method used in our proposed model has a convergence guarantee here, we must note that gradient-based optimization for variational quantum circuits can encounter barren plateaus [27] when dealing with large-scale problems. This is a known limitation of our proposed model.

6 Applications

In this section we first apply QAGate to a basic scenario of the quantum cloud computing, in which we just consider single-round communication between the client and the server. Also, there is just one family of n-qubit quantum gates $\{U_x\}_{x\in\mathsf{X}}$ stored in the server. Then, we extend the structure of QAEGate to apply it to two more general scenarios. One allows multiple-round communication between the client and the server. The other enables the server to store different families of n-qubit quantum gates.

6.1 Basic scenario

Let us start from the basic scenario introduced in Figure 4 of Section 4. The client starts from a generic n-qubit input ρ and converts it into an a-qubit state, by applying the gate $U^{\boldsymbol{\theta}_{\mathbf{ld}}}$ and storing aside $n-a$ qubits in a quantum memory. Then, the a qubits are sent to the server through a quantum communication link. The server implements the encoder supermap, converting the original n-qubit gate U_x into a quantum channel acting on a qubits. The encoded channel is applied to an a-qubit state received from the client, and produces an a-qubit state is sent back to the client through the quantum communication link. Finally, the client performs the gate $U^{\boldsymbol{\theta}_{\mathbf{rd}}}$ on the a qubits received from the server and on the $n-a$ qubits previously stored in the quantum memory. Overall, the operations performed at the client's end implement the decoder, transforming the a-qubit channel implemented by the server into an approximation of the target n-qubit gate.

An advantage of the structure of QAEGate is that it can fulfill the server's confidentiality. Specifically, the server wants to forbid the client from obtaining the detailed implementation of the gates, which is described by x in this scenario. For sure, some information about x must be leaked to the client since the client will be approximately granted one access to U_x, which could be used to extract information about x, but we claim that an arbitrary malicious client could not obtain more information than this minimal amount. Formally, we obtain the following confidentiality guarantee for our model:

Theorem 2 (Confidentiality guarantee) *In this protocol, the information about x obtained by an arbitrary malicious client is no larger than the information about x obtainable by accessing U_x once.*

Proof We first consider how much information about x can be extracted given one copy of U_x. We assume x follows a certain distribution, and we denote the random variable of x as X. The most general way to extract information from a gate U_x is to insert it into a quantum circuit and make measurements on the output of the circuit, as shown in Figure 8(a). Let the output of the circuit to be Y, which is a random variable correlated with X. We write $Y=\mathcal{F}(\mathcal{U}_X)$ to denote the relationship between X and Y. The information one can know about X, by obtaining the value of Y, is the mutual information $I(X;\ Y)$. The maximum amount of information one can learn

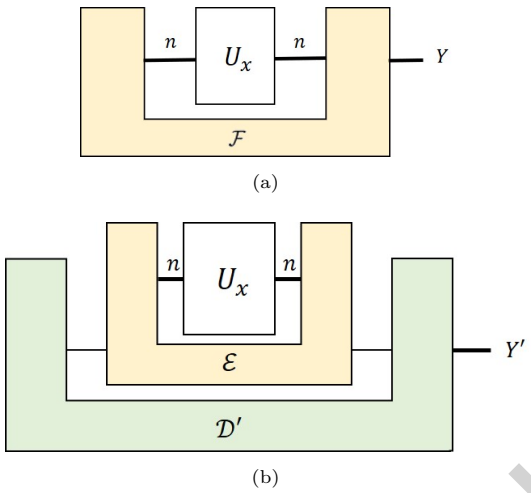

Fig. 8 (a) A general quantum circuit $\mathcal{F}$ for extracting information about x from U_x. (b) A general quantum circuit utilized by a malicious client to extract information about x by communicating with the server.

about X is obtained by optimizing the circuit, which is $I_{\max} := \max_{\mathcal{F}} I(X;\ Y)$. Since any party who is able to access U_x could obtain this amount of information, the server, who grants the client one use of U_x, would inevitably leak $I_{\max}$ amount of information about x to the client.

Next, we consider a malicious client who wants to obtain the most information about x. The server interacts with the client via the encoder, which is equivalent to that the client has access to the encoded gate, which is a channel $\mathcal{E}(\mathcal{U}_x)$. The most general way for the client to extract information about x is to insert $\mathcal{E}(\mathcal{U}_x)$ into a circuit and measure the output of the circuit, as shown in Figure 8(b). Let this circuit be $\mathcal{D}'$ and its output be Y', which is related to X by $Y' = (\mathcal{D}' \circ \mathcal{E})(\mathcal{U}_X)$. The maximum information the client could obtain is $\max_{\mathcal{D}'} I(X;\ Y')$.

Now, since $\mathcal{E}$ is determined by the server, the circuit $\mathcal{D}' \circ \mathcal{E}$ with variable $\mathcal{D}'$ forms a subset of the range of $\mathcal{F}$, which contains all possible circuits taking U_x as input. Therefore, the optimization over $\mathcal{D}'$ is upper bounded by the optimization over $\mathcal{F}$, and thus we have

$$\max_{\mathcal{D}'} I(X;\ Y') \leq I_{\max}\,. \tag{2}$$

The equation above proves Theorem 2, since the left hand side is the maximal information obtainable by an arbitrary malicious client, and the right hand side is the inevitable information leakage given one access to U_x.

□

6.2 Multi-round communication scenario

In this scenario, we relax the assumption of one-round communication and the client can communicate with the server for multiple rounds. In each round, the client can send to and receive from the sever an a-qubit quantum state

through the quantum communication link. For this multi-round communication scenario, the server can exploit U_x more than once. Here we depict the case of two-round communication in Figure 9.

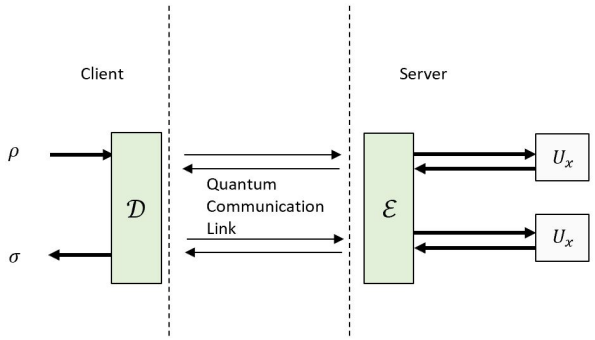

Fig. 9 The multi-round communication scenario. The client can communicate with the server for multiple rounds.

For the optimal performance, a general protocol may utilize different encoders and decoders for each round of communication. Here we keep the remaining $n-a$ qubits of the output of the encoder in the first round and regard it as a part of the input of the encoder in the second round. We depict such a variation of the structure of the QAEGate model in Figure 10. The encoders and decoders of the first and second round are separately parameterized.

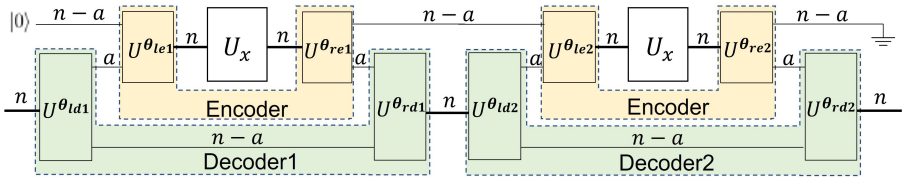

Fig. 10 QAEGate for multi-round communication scenario.

6.3 Sequence of gates scenario

In this scenario, we consider a more general situation that the client expects the server to execute a sequence of gates from different parametric families. In Figure 11, we depict the simplest case that the length of the sequence is 2. Here there are two classes of quantum gates, $\{U_x\}_{x\in\mathsf{X}}$ and $\{U_y\}_{y\in\mathsf{Y}}$, stored in the server. The client wants to obtain $\mathcal{U}_y(\mathcal{U}_x(\rho))$ by communicating with the server.

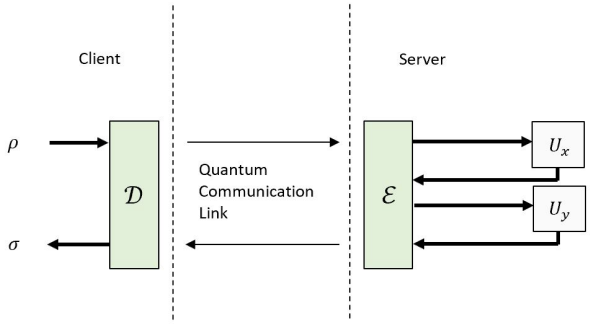

Fig. 11 Sequence of gates scenario. The client expects the server to execute a sequence of gates from different parametric families.

In order to handle this scenario, we design the encoder and the decoder with different structures and depict the modified QAEGate model in Figure 12.

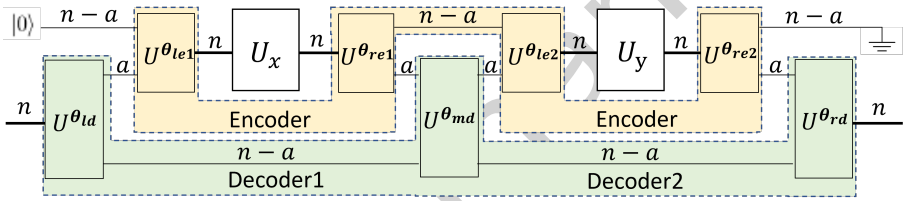

Fig. 12 QAEGate for sequence of gates scenario.

The encoder is composed of four parameterized unitary operators, denoted by $U^{\theta_{le1}}$, $U^{\theta_{re1}}$, $U^{\theta_{le2}}$ and $U^{\theta_{re2}}$. The first two operators are placed on both sides of U_x while the other two are placed on both sides of U_y. The decoder is composed of three parameterized parameterized unitary operators, denoted by $U^{\theta_{ld}}$, $U^{\theta_{md}}$ and $U^{\theta_{rd}}$.

7 Numerical Experiments

In this section, we examine the performance of our proposed models by numerical simulation. We show the effectiveness of our proposed models in different settings for gates based on the Heisenberg model. All simulations are implemented by Tensorflow quantum [28] and performed on a single GPU. We train the models by stochastic gradient descent in all of the experiments. More details are provided in the Appendix.

7.1 Heisenberg model

Heisenberg model [29] is a famous statistical mechanical model, which describes a magnetic system of half spins. It can be defined by the Hamiltonian

$$\tilde{H} = -\frac{1}{2}\sum_{j=1}^{n-1}(J_{\mathrm{x}}\sigma_j^x\sigma_{j+1}^x + J_{\mathrm{y}}\sigma_j^y\sigma_{j+1}^y + J_{\mathrm{z}}\sigma_j^z\sigma_{j+1}^z) - \frac{1}{2}\sum_{j=1}^{n} h\sigma_j^z,$$

where J_{x}, J_{y}, J_{z} and h are constants representing the strength of the coupling and the external magnetic field, respectively. σ^x, σ^y, σ^z are the Pauli matrices. n is the number of qubits in the quantum system. We define Heisenberg gates as the quantum gates which represent evolution defined by a Hamiltonian operator of Heisenberg model. These Heisenberg gates can be represented by

$$U(t) = \exp(-i\tilde{H}t),$$

where $\tilde{H}$ is the Hamiltonian of a Heisenberg model and t is the evolution time.

7.2 Simulation results

Basic Scenario.

In this scenario, we set $\{U_x\}$ stored in the sever as a class of Heisenberg gates with $J_{\mathrm{x}} = J_{\mathrm{y}} = J_{\mathrm{z}} = 0.1$ and $h = 0.5$.

Here we define the evolution time t as the parameter x, unknown to the client. The client can only transmit $a = 1$ qubit or $a = 2$ qubits through the quantum communication link. We consider three cases, $n = 2, 3, 4$, in the experiment. We exploit a training set composed of 50 gates to optimize the compression model and validate it on the test set composed of other 10 gates in each experiment.

We show the training curves of the QAEGate model in Figure 13. Here the x-axis represents the epoch number of the training, and the y-axis represents the overlap between the Choi states of the original gate and the decoded gate. We can see that all of the training processes converge finally, which conforms to our convergence analysis in Theorem 1.

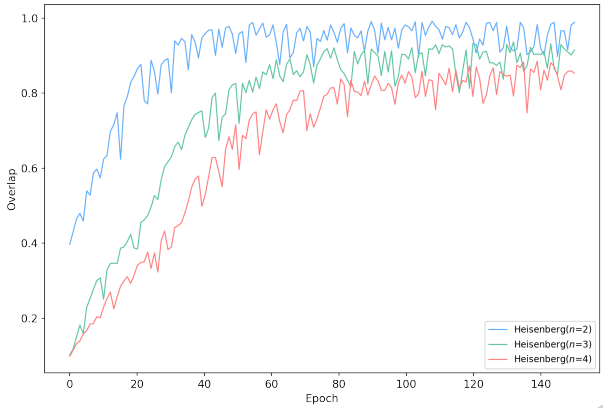

Fig. 13 Training curves of the QAEGate for the basic scenario. We train three QAEGate models for three classes of Heisenberg gates, $n = 2, 3, 4$, respectively. All of training curves converge finally, which conforms to our convergence analysis.

Furthermore, we present the performance of our QAEGate model on test sets in Figure 14. The y-axis represents the average overlap values between the Choi states of the decoded quantum gate and the original quantum gate. We can see that our QAEGate model achieves satisfactory performance in all of the experiments, which confirms the effectiveness of our proposed communication model empirically. Nevertheless, it is worth noting that QAEGate exhibits better and more consistent performance when the dimension of the class of Heisenberg gates $\{U_x\}$, denoted by n, is small. This could be attributed to the greater difficulty in achieving good generalization as the complexity of the gate class increases.

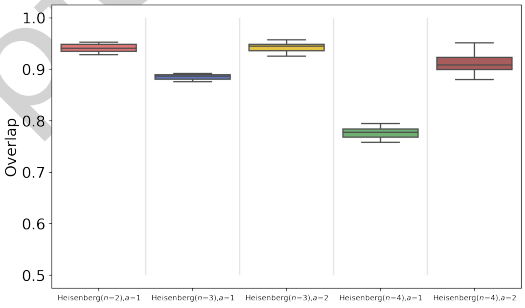

Fig. 14 The performance of our QAEGate model on test sets of the basic scenario visualized by boxplots [30].

Multi-round communication scenario.

In this scenario, we allow two-round communication between the server and the client while they can only transmit $a = 1$ qubit through the quantum communication link at a time.

We present the performance of our QAEGate model on test sets of the two-round communication scenario in Figure 15. The results indicate that our proposed model performs better when additional round of communication is available, compared with the basic scenario with $a = 1$. Especially for the case of $n = 4$, the average overlap value increases by 0.12 compared with the basic single-round scenario. We also compare the case of $a = 2$ in the basic scenario and the case of $a = 1$ in the two-rounds communication scenario. In both cases, two qubits are sent to and received from the server in total. We find that our proposed models have similar performance.

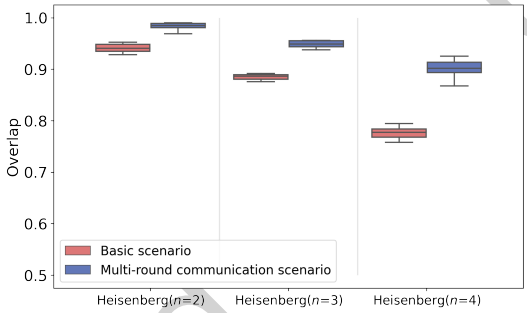

Fig. 15 The performance of our QAEGate model on test sets of the multi-round communication scenario visualized by boxplots [30].

Sequence of gates scenario.

In this scenario, we set $\{U_x\}$ stored in the sever as a class of Heisenberg gates with $J_x = J_y = J_z = 0.1$ and $h = 0.5$ and $\{U_y\}$ as a class of Heisenberg gates with $J_x = J_y = 0.1$ and $J_z = h = 0.5$. Here we define the evolution times of them as x and y whose values can be different. The client still can only transmit $a = 1$ qubit through the quantum communication link. We also consider three cases, $n = 2, 3, 4$, in the experiment. The simulation results on test sets of this scenario are shown in Figure 16 and our proposed model achieved satisfactory performance in all three cases.

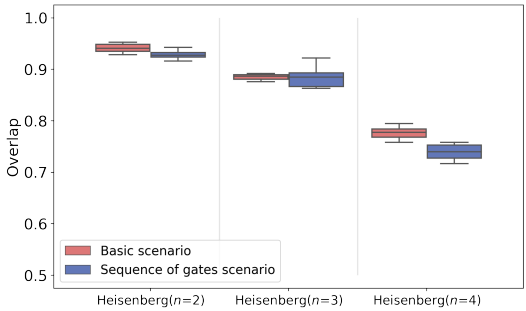

Fig. 16 The performance of our QAEGate model on test sets of the sequence of gates scenario visualized by boxplots [30].

8 Discussion and conclusions

We proposed a quantum machine learning model, QAEGate, for communication-efficient quantum cloud computing, and applied it to single-round and multi-round communication scenarios. In the single-round scenario, the protocol avoids leakage of information about the server's computation and achieves a satisfactory performance in terms of overlap with the target gate versus the number of qubits in its compressed version. By adjusting the structure of QAEGate, we applied our method to other two scenarios, involving multiple gates and multiple rounds of communication between the client and server.

The performance of our method has been tested numerically in a number of examples. Due to the exponential complexity of simulating quantum gates on classical computers, such analysis is necessarily limited to scenarios involving a small number of qubits. On the other hand, our method is meant to be eventually run on real quantum computers, on which the complexity of QAEGate grows only polynomially. The current progress of quantum hardware suggests that a real-world test may take place not too far in the future.

An application of the techniques developed in this paper is learning unknown quantum gates [31–33]. In this task, the goal is to imprint an unknown gate into the state of a quantum memory from which the gate can be accessed at a later time. Our QAEGate can be adapted to this task by adjusting the structure of the encoder and the decoder so that quantum gates can be encoded into quantum states. Referring to Figure 5, the idea is to choose different number of qubits a and a' for the input and output wires of the encoder, respectively. In particular, one can set a to zero, meaning that the encoder produces a quantum state of a' qubits. The number of qubits a' can be regarded as the size of the quantum memory of a learning machine designed to learn a generic gate from a given parametric family of quantum gates. An interesting feature of our method is that it allows one to set the memory size

(*i.e.* to fix a') and to achieve approximate learning subject to constraints on the available quantum memory.

Another interesting direction of future research is assessing the power of quantum processors [34, 35]. It has been often pointed out that the number of qubits in a quantum processor is not an appropriate measure, because the presence of noise generally limits the set of achievable computations. Hence, it is important to have more inclusive measures, that take into account not only the raw number of qubits, but also the size of the set of operations implementable on them. The approach of quantum gate compression sheds light into this problem by providing a rigorous notion of "effective size" of a quantum processor, defined as the minimum number of qubits on which the operations of the processor can be compressed with sufficiently high fidelity.

More specifically, the set of operations implemented by a given quantum processor can be regarded as a parametric family of (generally noisy) quantum gates. When the processor is not universal, such a family may be compressible into operations acting on a smaller number of qubits, meaning that the computations achieved by the processor can be in principle simulated with a smaller quantum computer. The size of this minimal simulator can then be regarded as a measure of the computing power of the original quantum processor. Such a measure is independent of the hardware implementation, and could be in principle evaluated by connecting the given processor to a reference quantum computer that runs the QAEGate algorithm, or a generalization of it working with noisy input gates.

A limitation of the above approach is that the evaluation of the minimum number of qubits requires a trusted quantum computer of the same size of the original processor. On the other hand, once a sufficiently large universal quantum computer becomes available, it can be used as a standard reference for assessing the power of other quantum processors, and to construct efficient simulations that can be run on universal quantum computers of smaller size.

9 Declarations

9.1 Competing interests

We declare that the authors have no competing interests as defined by Springer, or other interests that might be perceived to influence the results and/or discussion reported in this paper.

9.2 Authors' contributions

Zhu and Li provided the main ideas for the work, while Zhu, Bai, and Wang designed the numerical experiments. Zhu and Chiribella drafted the main manuscript text, and all authors reviewed the manuscript.

9.3 Funding

This work was supported by funding from the Hong Kong Research Grant Council through grants no. 17300918 and no. 17307520, through the Senior Research Fellowship Scheme SRFS2021-7S02, the Croucher Foundation, and by the John Templeton Foundation through grant 62312, The Quantum Information Structure of Spacetime (qiss.fr). YXW acknowledges funding from the National Natural Science Foundation of China through grants no. 61872318. Research at the Perimeter Institute is supported by the Government of Canada through the Department of Innovation, Science and Economic Development Canada and by the Province of Ontario through the Ministry of Research, Innovation and Science. The opinions expressed in this publication are those of the authors and do not necessarily reflect the views of the John Templeton Foundation.

9.4 Availability of data and materials

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

[1] Preskill, J.: Quantum computing in the NISQ era and beyond. Quantum **2**, 79 (2018)

[2] Arute et al., F.: Quantum supremacy using a programmable superconducting processor. Nature **574**(7779), 505–510 (2019)

[3] Zhong et al., H.-S.: Quantum computational advantage using photons. Science **370**(6523), 1460–1463 (2020)

[4] Broadbent, A., Fitzsimons, J., Kashefi, E.: Universal blind quantum computation. In: 2009 50th Annual IEEE Symposium on Foundations of Computer Science, pp. 517–526 (2009). IEEE

[5] Arrighi, P., Salvail, L.: Blind quantum computation. International Journal of Quantum Information **4**(05), 883–898 (2006)

[6] Morimae, T., Fujii, K.: Blind quantum computation protocol in which alice only makes measurements. Physical Review A **87**(5), 050301 (2013)

[7] Schmidhuber, J.: Deep learning in neural networks: An overview. Neural networks **61**, 85–117 (2015)

[8] Romero, J., Olson, J.P., Aspuru-Guzik, A.: Quantum autoencoders for efficient compression of quantum data. Quantum Science and Technology **2**(4), 045001 (2017)

[9] Bottou, L.: Stochastic gradient descent tricks. In: Neural Networks: Tricks of the Trade, pp. 421–436. Springer, ??? (2012)

[10] Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., *et al.*: A view of cloud computing. Communications of the ACM **53**(4), 50–58 (2010)

[11] Barz, S., Kashefi, E., Broadbent, A., Fitzsimons, J.F., Zeilinger, A., Walther, P.: Demonstration of blind quantum computing. science **335**(6066), 303–308 (2012)

[12] Yang, Y., Chiribella, G., Hayashi, M.: Communication cost of quantum processes. IEEE Journal on Selected Areas in Information Theory **1**(2), 387–400 (2020)

[13] Sheng, Y.-B., Zhou, L.: Distributed secure quantum machine learning. Science Bulletin **62**(14), 1025–1029 (2017)

[14] Bondarenko, D., Feldmann, P.: Quantum autoencoders to denoise quantum data. Physical Review Letters **124**(13), 130502 (2020)

[15] Achache, T., Horesh, L., Smolin, J.: Denoising quantum states with Quantum Autoencoders–Theory and Applications (2020)

[16] Nielsen, M.A., Chuang, I.L.: Programmable quantum gate arrays. Physical Review Letters **79**(2), 321 (1997)

[17] Yang, Y., Renner, R., Chiribella, G.: Optimal universal programming of unitary gates. Physical Review Letters **125**(21), 210501 (2020)

[18] Choi, M.-D.: Completely positive linear maps on complex matrices. Linear algebra and its applications **10**(3), 285–290 (1975)

[19] Jamiołkowski, A.: Linear transformations which preserve trace and positive semidefiniteness of operators. Reports on Mathematical Physics **3**(4), 275–278 (1972)

[20] Nielsen, M.A., Chuang, I.: Quantum computation and quantum information. American Association of Physics Teachers (2002)

[21] Chiribella, G., D'Ariano, G.M., Perinotti, P.: Transforming quantum operations: Quantum supermaps. EPL (Europhysics Letters) **83**(3), 30004 (2008)

[22] Chiribella, G., D'Ariano, G.M., Perinotti, P.: Theoretical framework for quantum networks. Physical Review A **80**(2), 022339 (2009)

[23] Cong, I., Choi, S., Lukin, M.D.: Quantum convolutional neural networks.

Nature Physics **15**(12), 1273–1278 (2019)

[24] Farhi, E., Goldstone, J., Gutmann, S.: A quantum approximate optimization algorithm (2014)

[25] Debnath, S., Linke, N.M., Figgatt, C., Landsman, K.A., Wright, K., Monroe, C.: Demonstration of a small programmable quantum computer with atomic qubits. Nature **536**(7614), 63–66 (2016)

[26] Kiefer, J., Wolfowitz, J., *et al.*: Stochastic estimation of the maximum of a regression function. The Annals of Mathematical Statistics **23**(3), 462–466 (1952)

[27] McClean, J.R., Boixo, S., Smelyanskiy, V.N., Babbush, R., Neven, H.: Barren plateaus in quantum neural network training landscapes. Nature communications **9**(1), 4812 (2018)

[28] Broughton, M., Verdon, G., McCourt, T., Martinez, A.J., Yoo, J.H., Isakov, S.V., Massey, P., Halavati, R., Niu, M.Y., Zlokapa, A., et al.: Tensorflow quantum: A software framework for quantum machine learning. arXiv preprint arXiv:2003.02989 (2020)

[29] Baxter, R.J.: Exactly Solved Models in Statistical Mechanics. Courier Corporation, ??? (2007)

[30] Williamson, D.F., Parker, R.A., Kendrick, J.S.: The box plot: a simple visual method to interpret data. Annals of internal medicine **110**(11), 916–921 (1989)

[31] Bisio, A., Chiribella, G., D'Ariano, G.M., Facchini, S., Perinotti, P.: Optimal quantum learning of a unitary transformation. Physical Review A **81**(3), 032324 (2010)

[32] Mo, Y., Chiribella, G.: Quantum-enhanced learning of rotations about an unknown direction. New Journal of Physics **21**(11), 113003 (2019)

[33] Sedlák, M., Ziman, M.: Probabilistic storage and retrieval of qubit phase gates. Physical Review A **102**(3), 032618 (2020)

[34] Bishop, L.S., Bravyi, S., Cross, A., Gambetta, J.M., Smolin, J.: Quantum volume. Quantum Volume. Technical Report (2017)

[35] Moll, N., Barkoutsos, P., Bishop, L.S., Chow, J.M., Cross, A., Egger, D.J., Filipp, S., Fuhrer, A., Gambetta, J.M., Ganzhorn, M., *et al.*: Quantum optimization using variational algorithms on near-term quantum devices. Quantum Science and Technology **3**(3), 030503 (2018)

[36] Allen-Zhu, Z.: Natasha 2: Faster non-convex optimization than sgd. arXiv

preprint arXiv:1708.08694 (2017)

[37] Sweke, R., Wilde, F., Meyer, J.J., Schuld, M., Fährmann, P.K., Meynard-Piganeau, B., Eisert, J.: Stochastic gradient descent for hybrid quantum-classical optimization. Quantum **4**, 314 (2020)

[38] Cirq Developers: Cirq. Zenodo (2022). https://doi.org/10.5281/ZENODO.7465577. https://zenodo.org/record/7465577

[39] Mitarai, K., Negoro, M., Kitagawa, M., Fujii, K.: Quantum circuit learning. Physical Review A **98**(3), 032309 (2018)

A Proof of Theorem 1

According to [36], we have the following lemma about convergence of SGD for nonconvex functions:

Lemma 1 If a function $f(x)$ is L_1-smooth and we perform SGD update $x_{t+1} \leftarrow x_t - \eta \nabla f_i(x_t)$ each time for a random $i \in [n]$, then the convergence rate is $T = \mathcal{O}(\frac{L_1}{\epsilon^4})$ for finding $\mathbb{E}[\|\nabla f(x)\|^2] \leq \epsilon^2$.

Therefore, we first discuss the smoothness property of the loss function $\mathcal{L}(\boldsymbol{\theta})$ in the training of QAEGate model. Note that the first- and second-order Lipschitz smoothness are equivalent to Lipschitz continuity of the first- and second-order derivatives. So we start with a lemma about the Lipschitz continuity of multivariate functions [37].

Lemma 2 Given some function $f\colon \mathbf{R}^M \to \mathbf{R}$, if all partial derivatives of f are continuous, then for any $a, b \in \mathbf{R}$ the function $f\colon [a,b]^M \to \mathbf{R}$ is L-Lipschitz continuous with

$$L = \sqrt{M} \max_{j \in \{1,\ldots,M\}} \sup_{\boldsymbol{x} \in [a,b]^M} \left| \frac{\partial f(\boldsymbol{x})}{\partial x_j} \right|. \tag{3}$$

Next we prove a more general result about the smoothness of measurement probability of quantum circuits, and then show that $\mathcal{L}(\boldsymbol{\theta})$ fits into this result.

Lemma 3 Consider a quantum circuit consisting of any number of fixed unitary gates and M variable unitary gates $U_1(\theta_1), \ldots, U_M(\theta_M)$, where $U_j(\theta_j) := \exp(i\theta_j H_j)$ for some Hermitian operator H_j. Then the probability of any measurement outcome of the output of this circuit is L_1-smooth and L_2-second-order smooth with respect to $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_M)$, where

$$L_1 = 4MH_{\max}^2 \tag{4}$$

$$L_2 = 8M^{3/2}H_{\max}^3 \tag{5}$$

where $H_{\max} := \max_j \|H_j\|_2$ and $\|\cdot\|_2$ is the induced operator norm defined as $\|A\|_2 := \sup \frac{\|Ax\|_2}{\|x\|_2}$.

Proof Let $|\psi_0\rangle$ be the initial state of the circuit. Without loss of generality, we assume the variable unitary gates are labeled as $U_1(\theta_1)$ to $U_M(\theta_M)$ according to the order they are applied. Then the output of this circuit can be written as $|\psi_{\boldsymbol{\theta}}\rangle := V_M U_M(\theta_M) \dots V_1 U_1(\theta_1) V_0 |\psi_0\rangle$, where $V_1, \dots, V_M$ denotes the fixed unitary gates. A measurement outcome can be described by a positive operator-valued measure (POVM) element P with the property that P and $I - P$ are both positive semidefinite, where I is the identity operator of the output. The probability of this outcome is

$$f(\boldsymbol{\theta}) := \mathrm{Tr}[P|\psi_{\boldsymbol{\theta}}\rangle\langle\psi_{\boldsymbol{\theta}}|] = \langle\psi_{\boldsymbol{\theta}}|P|\psi_{\boldsymbol{\theta}}\rangle\,. \tag{6}$$

To study the smoothness of $f(\boldsymbol{\theta})$, we take the partial derivatives:

$$\begin{aligned}
\frac{\partial f(\boldsymbol{\theta})}{\partial\theta_j} &= \frac{\partial\langle\psi_{\boldsymbol{\theta}}|}{\partial\theta_j} P|\psi_{\boldsymbol{\theta}}\rangle + \langle\psi_{\boldsymbol{\theta}}|P\frac{\partial|\psi_{\boldsymbol{\theta}}\rangle}{\partial\theta_j} = 2\Re\left(\frac{\partial\langle\psi_{\boldsymbol{\theta}}|}{\partial\theta_j} P|\psi_{\boldsymbol{\theta}}\rangle\right)\\
\frac{\partial^2 f(\boldsymbol{\theta})}{\partial\theta_j\partial\theta_k} &= 2\Re\left(\frac{\partial^2\langle\psi_{\boldsymbol{\theta}}|}{\partial\theta_j\partial\theta_k} P|\psi_{\boldsymbol{\theta}}\rangle + \frac{\partial\langle\psi_{\boldsymbol{\theta}}|}{\partial\theta_j} P\frac{\partial|\psi_{\boldsymbol{\theta}}\rangle}{\partial\theta_k}\right)\\
\frac{\partial^3 f(\boldsymbol{\theta})}{\partial\theta_j\partial\theta_k\partial\theta_l} &= 2\Re\left(\frac{\partial^3\langle\psi_{\boldsymbol{\theta}}|}{\partial\theta_j\partial\theta_k\partial\theta_l} P|\psi_{\boldsymbol{\theta}}\rangle + \frac{\partial^2\langle\psi_{\boldsymbol{\theta}}|}{\partial\theta_j\partial\theta_k} P\frac{\partial|\psi_{\boldsymbol{\theta}}\rangle}{\partial\theta_l} + \frac{\partial^2\langle\psi_{\boldsymbol{\theta}}|}{\partial\theta_j\partial\theta_l} P\frac{\partial|\psi_{\boldsymbol{\theta}}\rangle}{\partial\theta_k} + \frac{\partial\langle\psi_{\boldsymbol{\theta}}|}{\partial\theta_j} P\frac{\partial^2|\psi_{\boldsymbol{\theta}}\rangle}{\partial\theta_k\partial\theta_l}\right)
\end{aligned}$$

These partial derivatives can be bounded with vector and operator norms. For a unitary operator U, $\|U\|_2 = 1$. For the POVM element P, $\|P\|_2 \le 1$. The second-order partial derivatives can be bounded as:

$$\begin{aligned}
\left|\frac{\partial^2 f(\boldsymbol{\theta})}{\partial\theta_j\partial\theta_k}\right| &\le 2\left|\frac{\partial^2\langle\psi_{\boldsymbol{\theta}}|}{\partial\theta_j\partial\theta_k} P|\psi_{\boldsymbol{\theta}}\rangle + \frac{\partial\langle\psi_{\boldsymbol{\theta}}|}{\partial\theta_j} P\frac{\partial|\psi_{\boldsymbol{\theta}}\rangle}{\partial\theta_k}\right|\\
&\le 2\left\|\frac{\partial^2\langle\psi_{\boldsymbol{\theta}}|}{\partial\theta_j\partial\theta_k}\right\|_2 \|P\|_2 \||\psi_{\boldsymbol{\theta}}\rangle\|_2 + 2\left\|\frac{\partial\langle\psi_{\boldsymbol{\theta}}|}{\partial\theta_j}\right\|_2 \|P\|_2 \left\|\frac{\partial|\psi_{\boldsymbol{\theta}}\rangle}{\partial\theta_k}\right\|_2\\
&\le 2\max_{a,b}\left\|\frac{\partial^2|\psi_{\boldsymbol{\theta}}\rangle}{\partial\theta_a\partial\theta_b}\right\|_2 + 2\max_a\left\|\frac{\partial|\psi_{\boldsymbol{\theta}}\rangle}{\partial\theta_a}\right\|_2^2
\end{aligned} \tag{7}$$

Similarly, for the third-order partial derivatives,

$$\begin{aligned}
\left|\frac{\partial^3 f(\boldsymbol{\theta})}{\partial\theta_j\partial\theta_k\partial\theta_l}\right| &\le 2\left|\frac{\partial^3\langle\psi_{\boldsymbol{\theta}}|}{\partial\theta_j\partial\theta_k\partial\theta_l} P|\psi_{\boldsymbol{\theta}}\rangle + \frac{\partial^2\langle\psi_{\boldsymbol{\theta}}|}{\partial\theta_j\partial\theta_k} P\frac{\partial|\psi_{\boldsymbol{\theta}}\rangle}{\partial\theta_l} + \frac{\partial^2\langle\psi_{\boldsymbol{\theta}}|}{\partial\theta_j\partial\theta_l} P\frac{\partial|\psi_{\boldsymbol{\theta}}\rangle}{\partial\theta_k} + \frac{\partial\langle\psi_{\boldsymbol{\theta}}|}{\partial\theta_j} P\frac{\partial^2|\psi_{\boldsymbol{\theta}}\rangle}{\partial\theta_k\partial\theta_l}\right|\\
&\le 2\max_{a,b,c}\left\|\frac{\partial^3|\psi_{\boldsymbol{\theta}}\rangle}{\partial\theta_a\partial\theta_b\partial\theta_c}\right\|_2 + 6\left(\max_{a,b}\left\|\frac{\partial^2|\psi_{\boldsymbol{\theta}}\rangle}{\partial\theta_a\partial\theta_b}\right\|_2\right)\left(\max_a\left\|\frac{\partial|\psi_{\boldsymbol{\theta}}\rangle}{\partial\theta_a}\right\|_2\right)
\end{aligned} \tag{8}$$

Therefore, to show that the second- and third-order partial derivatives are bounded, it suffices to show that $\max_a \left\|\frac{\partial|\psi_{\boldsymbol{\theta}}\rangle}{\partial\theta_a}\right\|_2, \max_{a,b}\left\|\frac{\partial^2|\psi_{\boldsymbol{\theta}}\rangle}{\partial\theta_a\partial\theta_b}\right\|_2$ and $\max_{a,b,c}\left\|\frac{\partial^3|\psi_{\boldsymbol{\theta}}\rangle}{\partial\theta_a\partial\theta_b\partial\theta_c}\right\|_2$ are all bounded. We observe that $\frac{\partial U_j(\theta_j)}{\partial j} = \frac{\partial\exp(i\theta_j H_j)}{\partial j} = iH_jU_j(\theta_j)$, and we have

$$\begin{aligned}
\max_a\left\|\frac{\partial|\psi_{\boldsymbol{\theta}}\rangle}{\partial\theta_a}\right\|_2 &= \max_a \|V_M U_M(\theta_M)\dots V_a i H_a U_a(\theta_a)\dots U_1(\theta_1)V_0|\psi_0\rangle\|_2\\
&\le \max_a \|V_M U_M(\theta_M)\dots V_a\|_2 \|H_a\|_2 \|U_a(\theta_a)\dots U_1(\theta_1)V_0|\psi_0\rangle\|_2
\end{aligned}$$

$$= \max_a \|H_a\|_2 = H_{\max} \tag{9}$$

With similar derivations, we can obtain $\max_{a,b} \left\| \frac{\partial^2 |\psi_{\boldsymbol{\theta}}\rangle}{\partial\theta_a \partial\theta_b} \right\|_2^2 \leq H_{\max}^2$ and $\max_{a,b,c} \left\| \frac{\partial^3 |\psi_{\boldsymbol{\theta}}\rangle}{\partial\theta_a \partial\theta_b \partial\theta_c} \right\|_2 \leq H_{\max}^3$. According to Eqs. (7) and (8), we obtain

$$\left| \frac{\partial^2 f(\boldsymbol{\theta})}{\partial\theta_j \partial\theta_k} \right| \leq 4H_{\max}^2 \tag{10}$$

$$\left| \frac{\partial^3 f(\boldsymbol{\theta})}{\partial\theta_j \partial\theta_k \partial\theta_l} \right| \leq 8H_{\max}^3 \tag{11}$$

Consider $\frac{\partial f(\boldsymbol{\theta})}{\partial\theta_j}$ as a function of $\boldsymbol{\theta}$. Since its partial derivatives $\frac{\partial^2 f(\boldsymbol{\theta})}{\partial\theta_j \partial\theta_k}$ are bounded by $4H_{\max}^2$, according to Lemma 2, $\frac{\partial f(\boldsymbol{\theta})}{\partial\theta_j}$ is L_1'-continuous with $L_1' = 4\sqrt{M}H_{\max}^2$. Then

$$\begin{aligned} \|\nabla f(\boldsymbol{\alpha}) - \nabla f(\boldsymbol{\beta})\|_2 &\leq \sqrt{M} \max_j \left| \frac{\partial f(\boldsymbol{\alpha})}{\partial\theta_j} - \frac{\partial f(\boldsymbol{\beta})}{\partial\theta_j} \right| \\ &\leq \sqrt{M} L_1' \|\alpha - \beta\|_2 = 4MH_{\max}^2 \|\alpha - \beta\|_2 \end{aligned} \tag{12}$$

and therefore f is L_1-smooth with $L_1 = 4MH_{\max}^2$. Similarly, consider $\frac{\partial^2 f(\boldsymbol{\theta})}{\partial\theta_j \partial\theta_k}$ as a function of $\boldsymbol{\theta}$. Since its partial derivatives $\frac{\partial^3 f(\boldsymbol{\theta})}{\partial\theta_j \partial\theta_k \partial\theta_l}$ are bounded by $8H_{\max}^3$, according to Lemma 2, $\frac{\partial^2 f(\boldsymbol{\theta})}{\partial\theta_j \partial\theta_k}$ is L_2'-continuous with $L_2' = 8\sqrt{M}H_{\max}^3$. Then

$$\begin{aligned} \|\nabla^2 f(\boldsymbol{\alpha}) - \nabla^2 f(\boldsymbol{\beta})\|_2 &\leq M \max_{j,k} \left| \frac{\partial^2 f(\boldsymbol{\alpha})}{\partial\theta_j \partial\theta_k} - \frac{\partial^2 f(\boldsymbol{\beta})}{\partial\theta_j \partial\theta_k} \right| \\ &\leq M L_2' \|\alpha - \beta\|_2 = 8M^{3/2} H_{\max}^3 \|\alpha - \beta\|_2 \end{aligned} \tag{13}$$

and therefore f is L_2-second-order smooth with $L_2 = 8M^{3/2}H_{\max}^3$. □

Based on above lemma, we obtain the following theorem:

Theorem 3 (Smoothness) *The loss function $\mathcal{L}(\boldsymbol{\theta})$ in the training of QAEGate model is L_1-smooth and L_2-second-order smooth.*

Proof Let $p(\boldsymbol{\theta})$ be the probability of outcome $|+\rangle$ in the SWAP test when the gates are parameterized with $\boldsymbol{\theta} = (\boldsymbol{\theta_{le}}, \boldsymbol{\theta_{re}}, \boldsymbol{\theta_{ld}}, \boldsymbol{\theta_{rd}})$. Because of $\mathcal{L}(\boldsymbol{\theta}) = 1 - f(\boldsymbol{\theta})$ and $f(\boldsymbol{\theta}) = 2p(\boldsymbol{\theta}) - 1$, we just need to discuss the smoothness of $p(\boldsymbol{\theta})$. Since $p(\boldsymbol{\theta})$ is the probability of a measurement outcome of a unitary circuit, and in this circuit, every variable gate has the form of $U_\theta = \exp(i\theta H)$, we can apply Lemma 3. Thus, $p(\boldsymbol{\theta})$ is L_1-smooth and L_2-second-order smooth where $L_1 = 4MH_{\max}^2$, $L_2 = 8M^{3/2}H_{\max}^3$ and M is the number of variable gates in the circuit. To prove Theorem 3, we only need to show that $H_{\max}$ is bounded.

The variable gates in our circuit only includes the following single- and two-qubit gates:

$$R_X(\theta) := \exp(i\theta\sigma_x),\ R_Y(\theta) := \exp(i\theta\sigma_y),\ R_Z(\theta) := \exp(i\theta\sigma_z) \tag{14}$$

$$XX(\theta) := \exp(i\theta\sigma_x \otimes \sigma_x),\ YY(\theta) := \exp(i\theta\sigma_y \otimes \sigma_y),\ ZZ(\theta) := \exp(i\theta\sigma_z \otimes \sigma_z) \tag{15}$$

Therefore, $H_{\max} = \max\{\|\sigma_x\|_2, \|\sigma_y\|_2, \|\sigma_z\|_2, \|\sigma_x \otimes \sigma_x\|_2, \|\sigma_y \otimes \sigma_y\|_2, \|\sigma_z \otimes \sigma_z\|_2\} = 1$. □

Proof of Theorem 1 Based on Theorem 3, we have proved that $\mathcal{L}(\boldsymbol{\theta})$ is L_1-smooth, where $L_1 = 4M$. In our implementation, there is only one variable for each variable unitary gate, so $M = \dim \boldsymbol{\theta} = \mathcal{O}(n^2)$ here. Then we can apply Lemma 1 and obtain that the convergence rate is $T = \mathcal{O}(\frac{n^2}{\epsilon^4})$ for finding $\mathbb{E}[\|\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})\|^2] \leq \epsilon^2$. In order to estimate the gradients, we have to compute the loss function for $\mathcal{O}(\dim \boldsymbol{\theta}) = \mathcal{O}(n^2)$ times. The total time complexity of the algorithm is $\mathcal{O}(\frac{n^4}{\epsilon^4})$. □

B Details of Numerical Experiments

Data generation.

For each class of gates $U(t)$ we considered in different scenarios, we randomly generate 60 different $U(t_i), t_i \in [0, 2]$ and split them into the training set and the test set with ratio 50 : 10.

Hyperparameters

We set the learning rate η as 0.005, the maximum number of iterations K as 150 and the threshold δ as 0.999 in all experiments.

Impelementation details.

We implement our proposed model QAEGate based on Tensorflow Quantum [28]. The parameterized quanum circuits are constructed by the tools provided in Cirq [38] and the gradients are estimated by the differentiators based on parameter shift rule [39].

Training details.

We employed stochastic gradient descent to train all the models in our experiments, using a single GPU. The training time for all the experiments presented in this paper is less than one hour.