# Adversarial training of quantum Born machine

Haozhen Situ*
*College of Mathematics and Informatics, South China Agricultural University, Guangzhou 510642, China*

Zhimin He†
*School of Electronic and Information Engineering, Foshan University, Foshan 528000, China*

Lvzhou Li‡
*School of Data and Computer Science, Sun Yat-Sen University, Guangzhou 510006, China*

Shenggen Zheng§
*Institute for Quantum Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China*



Generative adversarial network (GAN) is an effective machine learning framework to train unsupervised generative models, and has drawn lots of attention in recent years. In the meantime, there are some researches that use parameterized quantum circuits to generate simple patterns. In this paper, we present a quantum version of GAN, where parameterized quantum circuits are trained by an adversarial discriminator, to generate new samples that follows the probability distribution of a given training dataset. Two families of quantum circuits, both composed of simple one-qubit rotation and two-qubit controlled-phase gates, are considered. The parameters are learned through classical gradient descent optimization. The results of a small-scale proof-of-principle numerical experiment demonstrate that quantum circuits can be trained in an adversarial way for generative tasks.

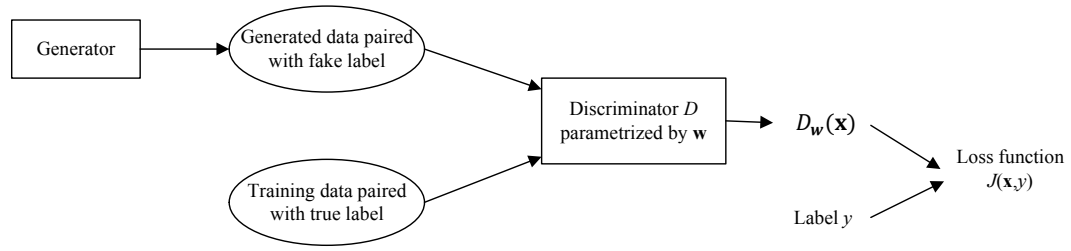
## I. INTRODUCTION

The era of quantum computing is around the corner. In 2016, IBM provided access to its quantum computer to the community through a cloud platform called IBM Quantum Experience [1]. A quantum computing competition among IT giants including Microsoft, Google, Intel is under way. Because quantum computing has the admirable capability of processing exponentially high-dimensional data, quantum machine learning [2] is expected to be one of the most intriguing future applications of quantum computers. Many theoretical and experimental researches [3–13] on machines learning problems with the help of quantum computing have been taken in the last decade.

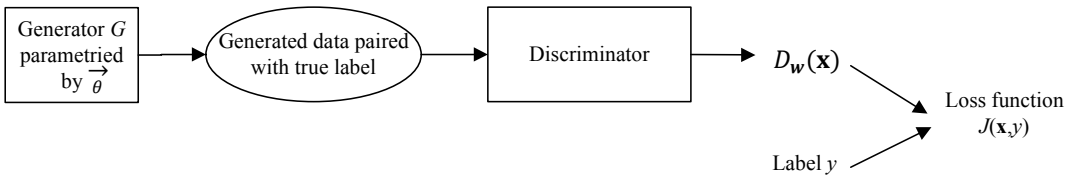
Designing explicit algorithms for artificial intelligence problems, *e.g.,* image and speech recognition, is very difficult or even impossible. Machine learning tries to solve these problems by parameterizing structured models and using training data to learn the parameters. There are two main classes of machine learning tasks, supervised learning and unsupervised learning. The goal of supervised learning is building the relation between given data samples and their labels, while unsupervised learning aims at discovering the intrinsic patterns, properties or structures of unlabeled data samples. Compared with advanced supervised learning techniques, unsupervised learning is more intractable and challenging, because it requires one to efficiently represent, learn and sample from high-dimensional probability distributions [14]. It's reasonable to conjecture that quantum computing has the potential to tackle unsupervised learning problems more effective and efficient than classical computing.

Dimension reduction is a common unsupervised learning technique. Classical autoencoders can give low dimensional representations of data in higher dimensional space, effectively compressing the data. A quantum generalisation of classical feedforward neural networks was proposed [15], establishing a formal connection between classical and quantum feedforward neural networks. The quantum neural network reduces to a classical neural network for a particular setting of parameters. This quantum neural network was trained to implement a quantum autoencoder for compressing quantum states onto fewer qubits and re-discovering the teleportation protocol. Ref. [16] trained a parameterized quantum circuit to implement a quantum autoencoder for compressing ground states of the Hubbard model and molecular Hamiltonians. In these works, parameterized quantum circuits take the place of the machine

*Electronic address: situhaozhen@gmail.com
†Electronic address: zhmihe@gmail.com
‡Electronic address: lilvzh@mail.sysu.edu.cn
§Electronic address: zhengsg@sustc.edu.cn




(a)update the discriminator's parameters $\mathbf{w}$ in order to minimize $J(\mathbf{x}, y)$




(b)update the generator's parameters $\vec{\theta}$ in order to minimize $J(x, y)$

FIG. 1: Building blocks of generative adversarial network. Step (a) and (b) are alternately carried out until the generator outputs the same data distribution as the target distribution.

learning parameterized models such as neural networks. The circuit parameters are trained by classical optimization methods such as gradient descent. This hybrid quantum-classical framework has attracted much attention in the quantum computing community.

Besides dimension reduction, generative modelling is another important unsupervised learning subject, which aims to model the distribution of training data and generate new data accordingly. Sampling from output distributions of random quantum circuits must take exponential time in a classical computer [17], which suggests that quantum circuits exhibit stronger representational power than classical neural networks. Ref. [18] trained shallow parameterized quantum circuits to generate GHZ states, coherent thermal states and Bars and Stripes images. Experiments have been done on an ion-trap quantum computer [19]. Ref. [20] developed a gradient-based learning scheme to train deep parameterized quantum circuits for generation of Bars and Stripes images and mixture of Gaussian distributions. These quantum generative models are also known as Born machines as the output probabilities are determined by Born's rule.

Among numerous training methods for generative models, Generative Adversarial Network (GAN) [21] has its unique characteristic and advantage. The idea is to introduce a discriminator to play the role of the generator's adversary, so their competition forms a two-player game. The objective of the generator is to produce samples resembling training samples, *a.k.a.,* real samples, while the objective of the discriminator is to distinguish real samples from generated ones, *a.k.a.,* fake samples. The process of training the discriminator and the generator alternately is described in Fig. 1. In step (a), the discriminator's parameters $\mathbf{w}$ are updated in order to minimize a loss function $J$. The purpose of this step is to make the discriminator a better adversary, so the generator has to try harder to fool the discriminator. In step (b), the output samples of the generator are labeled as real and then fed into the discriminator. The generator's parameters $\vec{\theta}$ are updated in order to minimize the loss function $J$, trying to make the discriminator believe the samples are real. After this step, the generator's ability improves a little bit. By repeating these two steps, the game will reach an equilibrium point, where the generator is able to generate the same statistics as the real samples and the prediction accuracy of the discriminator is $1/2$, not better than a random guess.

Compared with other generative models with explicit likelihoods such as the Boltzmann machines [22], normalizing flows [23–27], and variational autoencoders [28, 29], GAN as an implicit generative model can be more expressive due to less restrictions in network structures. Many variations of GAN have been proposed including conditional GAN [30], laplacian pyramid GAN [31], deep convolutional GAN [32] and Infogan [33]. GAN has been used to perform different tasks, *e.g.,* image generation[34], video generation [35], poem generation [36] and music generation [37].

The idea of quantum generative adversarial learning was recently explored theoretically in Ref. [38]. In Ref. [39], a quantum GAN consists of quantum generator and quantum discriminator was numerically implemented to generate simple quantum states. Ref. [40] derived an adversarial algorithm for the problem of approximating an unknown quantum pure state.

In this paper, we investigates adversarial training of quantum Born machine through a hybrid quantum-classical gradient descent approach. Two families of parameterized quantum circuits are adopted as the generator. Classical

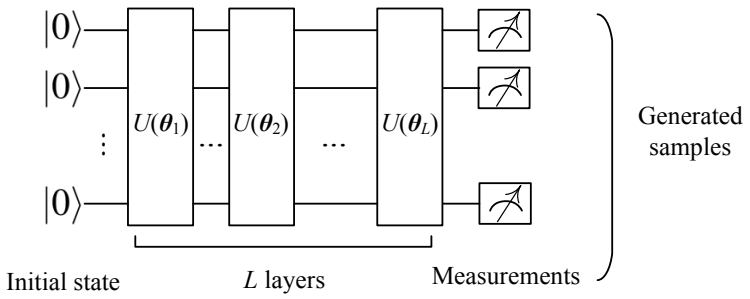



FIG. 2: The generative quantum circuit with $L$ layers

discriminator and optimization method are used to update the quantum generator's parameters. The paper is organized as follows. In section II, we present the constituents of the quantum generator and the classical discriminator. The loss function and optimization method is also described. Then the adversarial training algorithm is provided, together with the gradient estimation method for updating the parameters of the quantum generator. In section III, we report the numerical experiment that testify the effectiveness of our model. A brief conclusion follows in section IV.

## II. MODEL ARCHITECTURE AND TRAINING METHOD

In this section, we present the architecture of our generative quantum circuits built with simple one-qubit rotation and two-qubit controlled-phase gates, and the adversarial training scheme.

### A. Generative quantum circuit

The quantum circuit for generation of $N$-bit samples involves $N$ qubits, the layout of which is described in Fig. 2. The input quantum state is initialized to $|0\rangle^{\otimes N}$, and then passed through $L$ layers of unitary operations. At the end of the circuit, all the qubits are measured in the computational basis. The measurement outcomes are gathered to form an $N$-bit sample $x$. Each layer is composed of several one-qubit rotation gates and controlled-phase gates. Fig. 3 shows the arrangement of the gates in one layer. Three rotation operations are first applied to each qubit. This process can be written as

$$\prod_{i=1}^{N} R_z^i(\theta_{l,3}^i) R_x^i(\theta_{l,2}^i) R_z^i(\theta_{l,1}^i), \tag{1}$$

where the superscript $i$ denotes the $i$th qubit, the subscript $l$ denotes the $l$th layer. $R_x(\theta)$ and $R_z(\theta)$ are rotation gates, *i.e.,*

$$R_x(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}, R_z(\theta) = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}. \tag{2}$$

The number of parameters/gates in this process is $3N$ per layer. The choice of these operators is because any one-qubit unitary can be decomposed into this sequence of rotation operators [41].

We also need to entangle the qubits by performing controlled-$U$ gates between the qubits. This process can be written as

$$\prod_{i=1}^{N} CU^i_{(i \bmod N)+1}, \tag{3}$$

where the superscript $i$ denotes the control qubit, and the subscript $(i \bmod N) + 1$ denotes the target qubit. Each unitary is characterized by three parameters, so the number of parameters in this process is $3N$ per layer. However,

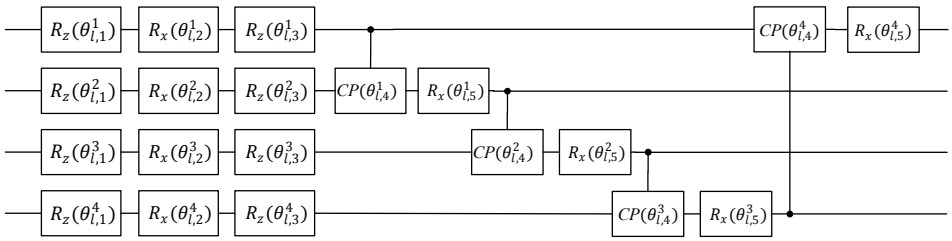



FIG. 3: A layer of the quantum circuit for four qubits

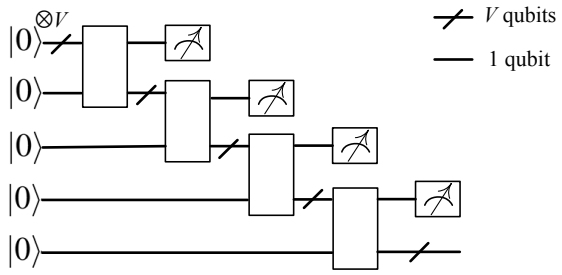



FIG. 4: The generative MPS quantum circuit with $N = 4$ nodes

Ref. [42] has pointed out that this process can be simplified as

$$\prod_{i=1}^{N} R_x^{(i \bmod N)+1}(\theta^i_{l,5}) CP^i_{(i \bmod N)+1}(\theta^i_{l,4}), \tag{4}$$

where

$$CP(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\theta} \end{pmatrix} \tag{5}$$

is the parameterized controlled-phase gate. Now the entangling process only has $2N$ parameters/gates per layer. The total number of parameters/gates in the quantum circuit is $5NL$. The set of all parameters can be denoted as a vector $\vec{\theta} = \{\theta_1, \ldots, \theta_{5NL}\}$ for convenience of expression.

### B. Generative MPS quantum circuit

Besides the aforementioned family of quantum circuits, we also consider another family of quantum circuits, which are called "MPS quantum circuits" [43]. Fig. 4 illustrates the structure of the MPS quantum circuit, which looks like a maximally unbalanced tree with $N$ nodes. Each node is a quantum ansatz which inputs and outputs $V + 1$ qubits. The uppermost output qubit of each node is measured in the computational basis and the other $V$ qubits flow to the next node. The $N$ measurement outcomes comprise the $N$-bit generated sample $x$. Each node can contain $L \geqslant 1$ layers which have the same gates and layouts as the layers depicted in Fig. 3. The number of parameters/gates in one node is $5L(V + 1)$, so the number of parameters/gates in an MPS quantum circuit is $5NL(V + 1)$. The input qubits are all initialized to $|0\rangle$ in our numerical experiment.

If the MPS quantum circuit is implemented using quantum devices. Each qubit that has been measured can be set to $|0\rangle$ and reused as the input of the next node. So only $V + 1$ qubits are actually needed in the circuit evaluation process. The sample dimension $N$ is only related to the depth of the circuit. Fig. 5 gives an equivalent form of the MPS circuit in order to illustrate the idea of qubit recycling. This quantum circuit has advantage in physical implementation because near-term quantum devices have limited number of qubits.

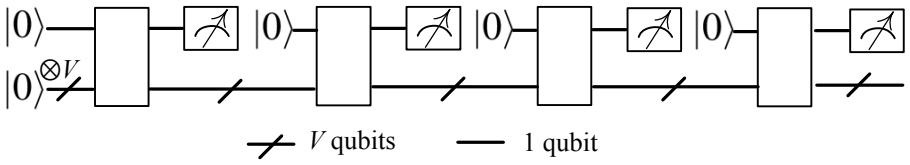



FIG. 5: The generative MPS quantum circuit with reused qubits

**Algorithm 1** Adversarial training algorithm of quantum Born machine

**Input:** $\{x_i\}$: the training set; $L$: number of layers; $V$: number of ancilla qubits (only for MPS circuit); **batch_size**: mini-batch size; **d_step**: times of updating $\mathbf{w}$ in one epoch; **g_step**: times of updating $\theta$ in one epoch;
**Output:** $\theta$: the parameters of the generator

**for** number of training epoches **do**
  **for d_step** steps **do**
    Sample a mini-batch of **batch_size** samples by running the quantum circuit **batch_size** times. Label them as "fake".
    Sample a mini-batch of **batch_size** samples from the training dataset. Label them as "real".
    Use these samples and labels to calculate the loss $J$.
    Update the discriminator's parameters $\mathbf{w}$ to minimize $J$.
  **end for**
  **for g_step** steps **do**
    Sample a mini-batch of **batch_size** samples by running the quantum circuit **batch_size** times. Label them as "real".
    Use these samples and labels to calculate the loss $J$.
    Update the generator's parameters $\theta$ to minimize $J$.
    (Notice that the discriminator's parameters $\mathbf{w}$ are fixed in this step.)
  **end for**
**end for**

### C. Discriminator

A discriminator $D$ is introduced to distinguish between real samples and generated samples. We use a shallow feedforward neural network to serve as the discriminator. The input layer has the same dimension as the sample dimension. Only one hidden layer is employed. The output layer has only one output value in $[0, 1]$, which represents the discriminator's prediction about the probability of the input sample being real. An output $D(x) = 1$ means the discriminator believes the input sample $x$ is definitely real, while an output $D(x) = 0$ means it believes the input sample $x$ is definitely fake.

The loss function we adopt is the commonly used binary cross entropy function

$$J(\mathbf{x}, \mathbf{y}) = -\frac{1}{\mathbf{num\_samples}} \sum_i y_i \log D(x_i) + (1 - y_i) \log(1 - D(x_i)), \tag{6}$$

where $(x_i, y_i) \in (\mathbf{x}, \mathbf{y})$ denotes the $i$th sample and its label, $y_i = 1(0)$ for real (fake) labels. The loss function evaluates how close the predictions $D(x_i)$ are to the desired labels $y_i$. If $D(x_i) = y_i$ for every $(x_i, y_i)$, then $J(\mathbf{x}, \mathbf{y}) = 0$.

The discriminator is trained by gradient descent:

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha_D \cdot \frac{\partial J}{\partial \mathbf{w}}, \tag{7}$$

where $\mathbf{w}$ groups all the parameters of the discriminator, and $\alpha_D$ is the learning rate, which can be adapted during the training process to decrease convergence time.

### D. Adversarial training

The adversarial training algorithm of quantum Born machine is described in Algorithm 1. The training process iterates for $t$ epoches, or until some stopping criterion is reached, *e.g.,* convergence on the loss function. At each epoch, the discriminator and the generator are updated **d_step** and **g_step** times, respectively.

### E. Gradient of the generator parameters

To update the generator's parameters $\vec{\theta} = \{\theta_1, \theta_2, \ldots\}$, we also employ gradient descent:

$$\vec{\theta} \leftarrow \vec{\theta} - \alpha_G \cdot \frac{\partial J}{\partial \vec{\theta}}, \tag{8}$$

where $\alpha_G$ is the learning rate, which can be adapted during the training process to decrease convergence time. We need to know the gradient

$$\frac{\partial J}{\partial \theta_i} = -\sum_{x \in \{0,1\}^N} \log D(x) \frac{\partial P_{\vec{\theta}}(x)}{\partial \theta_i}, \tag{9}$$

where $P_{\vec{\theta}}(x)$ is the probability of getting measurement outcome $x$ from the quantum circuit parameterized with $\vec{\theta}$. From Ref. [20] we know that

$$\frac{\partial P_{\vec{\theta}}(x)}{\partial \theta_i} = \frac{1}{2}(P_{\vec{\theta}^+}(x) - P_{\vec{\theta}^-}(x)), \tag{10}$$

where $\vec{\theta}^{\pm} = \vec{\theta} \pm \frac{\pi}{2}\mathbf{e}^i$, $\mathbf{e}^i$ is the $i$th unit vector in parameter space (i.e., $\theta_i \leftarrow \theta_i \pm \frac{\pi}{2}$, with other angles unchanged). Substitute Eq. (10) into Eq. (9), we have

$$\begin{aligned}
\frac{\partial J}{\partial \theta_i} &= -\frac{1}{2} \sum_{x \in \{0,1\}^N} (P_{\vec{\theta}^+}(x) - P_{\vec{\theta}^-}(x)) \log D(x) \\
&= \frac{1}{2} \sum_{x \in \{0,1\}^N} P_{\vec{\theta}^-}(x) \log D(x) - \frac{1}{2} \sum_{x \in \{0,1\}^N} P_{\vec{\theta}^+}(x) \log D(x) \\
&= \frac{1}{2} \mathbb{E}_{x \sim P_{\vec{\theta}^-}(x)} \log D(x) - \frac{1}{2} \mathbb{E}_{x \sim P_{\vec{\theta}^+}(x)} \log D(x).
\end{aligned} \tag{11}$$

In order to estimate the gradient of each $\theta_i$, we have to perform repeated evaluations of the circuits with parameters $\vec{\theta}^+$ and $\vec{\theta}^-$. In the case of small-scale numerical simulation, the wave function is accessible so the expectation can be directly calculated, or we can first calculate the probability distribution $P_{\vec{\theta}^+}$ and $P_{\vec{\theta}^-}$ from the wave function and then sample **grad_est** samples for estimation.

## III. NUMERICAL SIMULATION

We demonstrate our proposal using the canonical machine learning dataset known as Bars and Stripes (BAS). It's widely used to test generative models. The dataset contains $m \times m$ binary images with only bar patterns or stripe patterns. There are $2^m$ different vertical bar patterns and $2^m$ different horizontal stripe patterns. The all-black and all-white patterns are counted in both bar patterns and stripe patterns. So there are $2^{m+1} - 2$ different BAS patterns in an $m \times m$ image. We assume all BAS patterns appear with equal probability. Obviously each pixel can be encoded in one qubit, so an $m \times m$ image can be encoded in $m^2$ qubits. We restrict our experiments to the case of $m = 2$, because it's difficult to simulate more qubits effectively using an ordinary PC. Higher dimensional experiments will be done in the future if an intermediate-scale near-term quantum device is available.

The discriminator is implemented using the widely used deep learning framework PyTorch [44]. The discriminator has only one hidden layer made up of 50 hidden neurons with ReLU activation functions. The output neuron uses the Sigmoid activation function. The stochastic gradient optimizer Adam (Adaptive Moment Estimation) [45] is used instead of Eq. (7) to update the discriminator's parameters. The initial learning rate for Adam is $10^{-3}$.

The generative quantum circuit is simulated directly by calculating the evolution of the wavefunction. An $N$-qubit wavefunction is encoded in a $2^N$-dimensional vector. After performing a single-qubit operation $u_{11}|0\rangle\langle 0| + u_{12}|0\rangle\langle 1| + u_{21}|1\rangle\langle 0| + u_{22}|1\rangle\langle 1|$ on the $k$th qubit, the wavefunction is transformed to

$$\begin{aligned}
\alpha'_{*\ldots*0_k*\ldots*} &= u_{11} \cdot \alpha_{*\ldots*0_k*\ldots*} + u_{12} \cdot \alpha_{*\ldots*1_k*\ldots*}, \\
\alpha'_{*\ldots*1_k*\ldots*} &= u_{21} \cdot \alpha_{*\ldots*0_k*\ldots*} + u_{22} \cdot \alpha_{*\ldots*1_k*\ldots*},
\end{aligned} \tag{12}$$

where $\alpha$ and $\alpha'$ are amplitudes before and after transformation. The case of two-qubit operation can be deduced analogously.

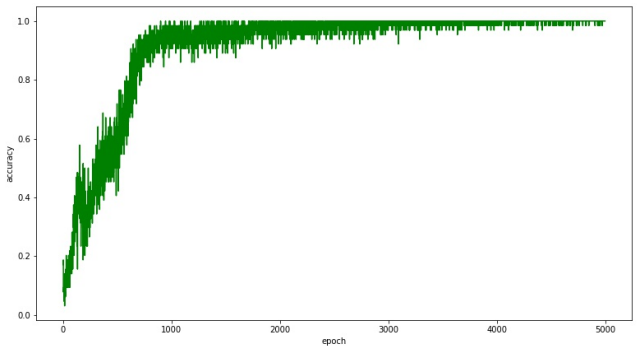



FIG. 6: Accuracy of the generator w.r.t. the number of epoches

The two numerical experiments differ in the architecture of the quantum generator. The first experiment uses the general quantum circuit described in section II A, while the second experiment uses the MPS quantum circuit described in section II B.

### A. Numerical experiment 1

In the first numerical experiment, the quantum generator is the general quantum circuit presented in section II A. After a lot of trials, we choose the hyper-parameters $L = 2$, **batch_size** $= 64$, **d_step** $= 1$, **g_step** $= 1$, **grad_est** $= 100$. For simplicity we use a constant learning rate $\alpha_G = 2 \times 10^{-2}$. The number of trainable parameters of the generator is 40. The parameters are randomly initialized in interval $(-\pi, \pi)$. Unlike the training of classification model, the stopping criterion of GAN is very tricky, so we simply run the training algorithm for 5000 epoches.

We first examine the accuracy of the generator. The accuracy of the generator in some epoch is defined as the ratio of the number of valid samples (i.e., BAS patterns) in one mini-batch to **batch_size**. The generator accuracy w.r.t. the number of epoches is depicted in Fig. 6. We can see that the accuracy increases very quickly and achieves nearly 100% in 1000 epoches, which means that it's not difficult for the generator to learn to avoid producing non-BAS patterns.

But our goal is not merely producing correct BAS patterns. The distribution of the generated patterns is expected to be the same as that of the training dataset, i.e., uniform distribution in our case. KL divergence is usually used to measure how one probability distribution diverges from a second, expected probability distribution, which is defined by

$$\mathrm{KLD}(p_d \| p_\theta) = -\sum_x p_d(x) \log \frac{p_\theta(x)}{p_d(x)}, \tag{13}$$

where $p_d$ and $p_\theta$ are the real data distribution and the generated distribution, respectively. $\mathrm{KLD}(p_d \| p_\theta)$ is non-negative and equals zero if and only if $p_d = p_\theta$ almost everywhere. The distribution of the generated samples can be estimated by their frequency of occurrences in one mini-batch. In numerical simulation, the exact distribution can be obtained from the wave function. We draw the KL divergence as a function of the number of epoch in Fig. 7, from which we can see that the generated distribution gradually approaches the real data distribution.

We also plot the loss functions of both the generator and the discriminator w.r.t. the number of epoches in Fig. 8. When the adversarial game reaches equilibrium, the output of the discriminator is 1/2 for both real and generated samples. By substituting $D(x_i) = 1/2$ into Eq. (6), we have $J_{final} = -\log \frac{1}{2} \approx 0.693$. From Fig. 8 we can see that both loss functions converge to $J_{final}$ after 4000 epoches.

### B. Numerical experiment 2

In the second numerical experiment, the quantum generator is the MPS quantum circuit presented in section II B. After a lot of trials, we choose the hyper-parameters $L = 1$, $V = 2$, **batch_size** $= 64$, **d_step** $= 1$, **g_step** $=$

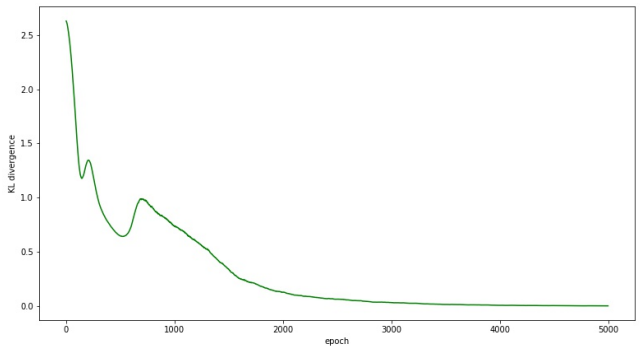



FIG. 7: KL divergence as a function of the number of epoches

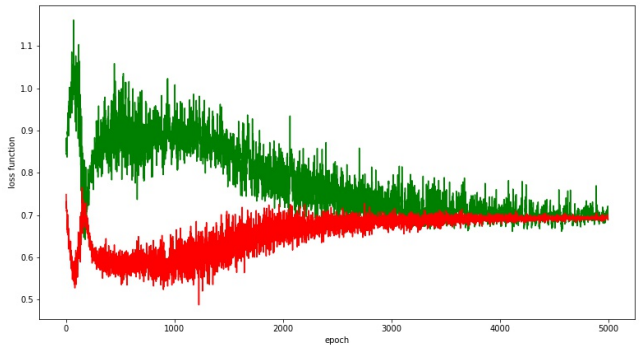



FIG. 8: Loss functions of the generator(in green) and the discriminator(in red) w.r.t. the number of epoches

1,**grad_est** $= 100$. For simplicity we use a constant learning rate $\alpha_G = 2 \times 10^{-2}$. The number of trainable parameters of the generator is 60. The parameters are randomly initialized in interval $(-\pi, \pi)$.

The generator accuracy w.r.t. the number of epoches is depicted in Fig. 9, which shows that the accuracy increases very quickly and achieves nearly 100% after 1000 epoches. The variation of the KL divergence is depicted in Fig. 10. We can see that the generated distribution gradually approaches the real data distribution. The variation of the loss functions of both the generator and the discriminator w.r.t. the number of epoches is plotted in Fig. 11. We can see that both loss functions converge to $J_{final}$ after 2000 epoches.

## IV. CONCLUSION

We believe the quantum version of GAN is very promising, due to the great potential and application of GAN in classical machine learning and the advantage of quantum computing. In this paper, we propose a quantum version of GAN composed of a Born machine as the generator and a classical feedforward neural network as the discriminator. Gradient descent is used to train the parameters of both the generator and the discriminator. By carefully designing the generator and choosing the hyper-parameters, we show our quantum GAN can generate simple BAS data distribution.

Interesting future research directions include generating other classical or quantum data with higher dimension, choosing the layout of the generative quantum circuit, modelling the generator with non-unitary quantum circuits, employing variants of GAN framework, using more heuristics to guide the training, and in-depth theoretical analysis of quantum GAN.

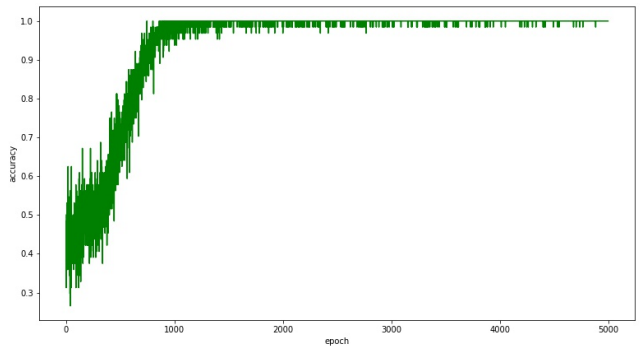



FIG. 9: Accuracy of the MPS generator w.r.t. the number of epoches

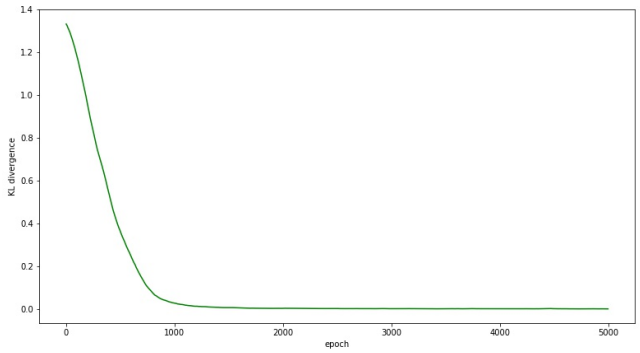



FIG. 10: KL divergence as a function of the number of epoches

**Acknowledgement**

This work is supported by the National Natural Science Foundation of China (Nos. 61772565, 61502179, 61602532, 61472452), the Natural Science Foundation of Guangdong Province of China (No. 2017A030313378), the Project of Department of Education of Guangdong Province (No.2017KQNCX216), the Science and Technology Program of Guangzhou City of China (No. 201707010194), the Fundamental Research Funds for the Central Universities (No.

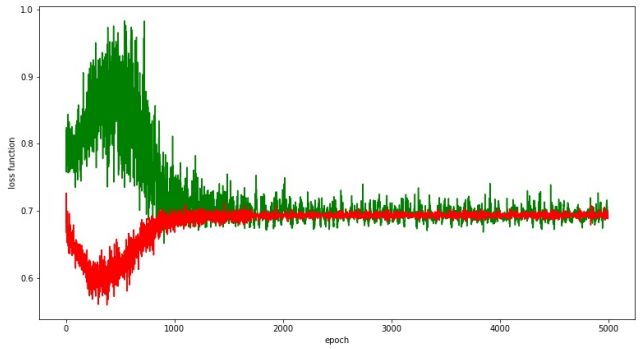



FIG. 11: Loss functions of the MPS generator(in green) and the discriminator(in red) w.r.t. the number of epoches

17lgzd29), the Project of Department of Education of Guangdong Province.(No. 2017KQNCX216), and the Research Foundation for Talented Scholars of Foshan University (No. gg040996).

[1] IBM Quantum Experience: http://www.research.ibm.com/ibm-q/
[2] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, Nature 549, 195 (2017).
[3] A.W. Harrow, A. Hassidim, and S. Lloyd, Quantum algorithm for linear systems of equations, Phys. Rev. Lett. 103, 150502 (2009).
[4] N. Wiebe, D. Braun, and S. Lloyd, Quantum algorithm for data fitting, Phys. Rev. Lett. 109, 050505 (2012).
[5] X.D. Cai, C. Weedbrook, Z.E. Su, M.C. Chen, M. Gu, M.J. Zhu, L. Li, N.L. Liu, C.Y. Lu, and J.W. Pan, Experimental quantum computing to solve systems of linear equations, Phys. Rev. Lett. 110, 230501 (2013).
[6] P. Rebentrost, M. Mohseni, and S. Lloyd, Quantum support vector machine for big data classification, Phys. Rev. Lett. 113, 130503 (2014).
[7] S. Lloyd, M. Mohseni, and P. Rebentrost, Quantum principal component analysis, Nat. Phys. 10, 631 (2014).
[8] X.D. Cai, D. Wu, Z.E. Su, M.C. Chen, X.L. Wang, L. Li, N.L. Liu, C.Y. Lu, and J.W. Pan, Entanglement-based machine learning on a quantum computer, Phys. Rev. Lett. 114, 110504 (2015).
[9] C.H. Yu, F. Gao, Q.L. Wang, and Q.Y. Wen, Quantum algorithm for association rules mining, Phys. Rev. A 94, 042311 (2016).
[10] V. Dunjko, J.M. Taylor, and H.J. Briegel, Quantum-enhanced machine learning, Phys. Rev. Lett. 117, 130501 (2016).
[11] A. Monràs, G. Sentís, and P. Wittek, Inductive supervised quantum learning, Phys. Rev. Lett. 118, 190503 (2017).
[12] B.J. Duan, J.B. Yuan, Y. Liu, and D. Li, Quantum algorithm for support matrix machines, Phys. Rev. A 96, 032301 (2017).
[13] C.H. Yu, F. Gao, C.H. Liu, D. Huynh, M. Reynolds, and J.B. Wang, Quantum algorithm for visual tracking, arXiv: 1807.00476.
[14] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, MIT Press (2016)
[15] K.H. Wan, O. Dahlsten, H. Kristjánsson, R. Gardner, and M.S. Kim, Quantum generalisation of feedforward neural networks, npj Quantum Information 3, 36 (2017).
[16] J. Romero, J.P. Olson, and A. Aspuru-Guzik, Quantum autoencoders for efficient compression of quantum data, Quantum Sci. and Technol. 2, 045001 (2017).
[17] S. Boixo, S.V. Isakov, V.N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M.J. Bremner, J.M. Martinis, and H. Neven, Characterizing quantum supremacy in near-term devices, Nat. Phys. 14, 595 (2018).
[18] M. Benedetti, D. Garcia-Pintos, Y. Nam, and A. Perdomo-Ortiz, A generative modeling approach for benchmarking and training shallow quantum circuits, arXiv: 1801.07686.
[19] S. Debnath, N.M. Linke, C. Figgatt, K.A. Landsman, K. Wright, and C. Monroe, Demonstration of a small programmable quantum computer with atomic qubits, Nature 536, 63 (2016).
[20] J.G. Liu and L. Wang, Differentiable learning of quantum circuit Born machine, arXiv: 1804.04168.
[21] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, Generative adversarial nets, in Proceedings of the 27th International Conference on Neural Information Processing Systems (2014) pp. 2672-2680.
[22] G.E. Hinton and R.R. Salakhutdinov, Reducing the dimensionality of data with neural network, Science 313, 504 (2006).
[23] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, Pixel recurrent neural networks, in Proceedings of the 33rd International Conference on Machine Learning (2016) pp. 1747-1756.
[24] L. Dinh, J. Sohl-Dickstein, and S. Bengio, Density estimation using real NVP, arXiv:1605.08803.
[25] D.P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, Improved variational inference with inverse autoregressive flow, in Proceedings of the 30th International Conference on Neural Information Processing Systems (2016) pp. 4743-4751.
[26] D.J. Rezende and S. Mohamed, Variational inference with normalizing flows, in Proceedings of the 32nd International Conference on International Conference on Machine Learning (2015) pp. 1530-1538.
[27] G. Papamakarios, I. Murray, and T. Pavlakou, Masked autoregressive flow for density estimation, in Proceedings of the 31th International Conference on Neural Information Processing Systems (2017) pp. 2335C2344.
[28] D.P. Kingma and M. Welling, Auto-encoding variational Bayes, arXiv:1312.6114.
[29] D.J. Rezende, S. Mohamed, and D. Wierstra, Stochastic backpropagation and approximate inference in deep generative models, in Proceedings of the 31st International Conference on Machine Learning (2014) pp. 1278-1286.
[30] M. Mirza and S. Osindero, Conditional generative adversarial nets, arXiv: 1411.1784.
[31] E.L. Denton, S. Chintala, and R. Fergus, Deep generative image models using a laplacian pyramid of adversarial networks, in Proceedings of the 29th International Conference on Neural Information Processing Systems (2015) pp. 1486-1494.
[32] A. Radford, L. Metz, and S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, arXiv: 1511.06434.
[33] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, Infogan: Interpretable representation learning by information maximizing generative adversarial nets, in Proceedings of the 30th International Conference on Neural

Information Processing Systems (2016) pp. 2172-2180.
[34] S. Reed, Z. Akata, X.C. Yan, L. Logeswaran, B. Schiele, and H. Lee, Generative adversarial text to image synthesis, in Proceedings of the 33rd International Conference on Machine Learning (2016) pp. 1060-1069.
[35] C. Vondrick, H. Pirsiavash, and A. Torralba, Generating videos with scene dynamics, in Proceedings of the 30th International Conference on Neural Information Processing Systems (2016) pp. 613-621.
[36] L.T. Yu, W.N. Zhang, J. Wang, and Y. Yu, SeqGAN: Sequence generative adversarial nets with policy gradient, in Proceedings of the 31st AAAI Conference on Artificial Intelligence (2017) pp. 2852-2858.
[37] L.C. Yang, S.Y. Chou, and Y.H. Yang, MidiNet: A convolutional generative adversarial network for symbolic-domain music generation, arXiv: 1703.10847.
[38] S. Lloyd and C. Weedbrook, Quantum generative adversarial learning, arXiv: 1804.09139.
[39] P. Dallaire-Demers and N. Killoran, Quantum generative adversarial networks, arXiv: 1804.08641.
[40] M. Benedetti, E. Grant, L. Wossnig, and S. Severini, Adversarial quantum circuit learning for pure state approximation, arXiv: 1806.00463.
[41] M.A. Nielsen and I.L. Chuang, Quantum Computation and Quantum Information, Cambridge University Press (2000)
[42] M. Schuld, A. Bocharov, K. Svore, N. Wiebe, Circuit-centric quantum classifiers, arXiv: 1804.00633.
[43] W. Huggins, P. Patel, K.B. Whaley, and E.M. Stoudenmire, Towards quantum machine learning with tensor networks, arXiv: 1803.11537.
[44] https://pytorch.org/
[45] D.P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv: 1412.6980.