Learning hard quantum distributions with generative neural networks

Abstract

Quantum computing has the potential to revolutionize numerous aspects of science and engineering in the future. Various research in the past decades has been undertaken to exhibit quantum advantage over classical computers. On the other hand, building robust quantum computers involves enormous technological challenges. Therefore, numerous attempts are devoted to close the gaps by developing a number of classical simulation algorithms for problems in many body physics. Among these, the most notable are machine learning algorithms. In this paper, we contribute to the development of these toolkits by studying the power of deep generative neural networks, including variational autoencoders and normalizing flows, to represent the distributions generated by quantum computing. Our results show that these modern generative networks are only capable of partially capturing~~capturing~~ features ~~capturing partial feat~~~~ures (add what kind of partial features)~~ of the hard quantum system~~distributions~~ including the overall shapes, but missing the granularities such as local spikes and annihilations in the distributions~~,~~ This incomplete information reconstruction~~which~~ might be due to the inherent limitation for classical algorithms when encountering the complex quantum states with exponential growing Hilbert space. The results also show that variational autoencoders perform slightly better than normalizing flows for many cases, and complimentary strength between the expressibility of variational autoencoders and normalizing flows are observed and worth further exploration.

Formatted: Font color: Auto

1. INTRODUCTION

We have seen rapid developments of quantum computing since Richard Feynman proposed the idea in the early 1980s [1]. As a groundbreaking interdisciplinary research field spanning mathematics, physics, computer science and engineering, quantum computing is poised to shape our technological future.

Potentializing to transform almost everything in our life before long, quantum computing is on the brink to revolutionize numerous domains, including cryptography and cybersecurity, pharmaceutical research, material sciences, artificial intelligence, and many more. Tech giants, including IBM, Google, Alibaba, and many others, are racing to break through and dominate this space [2] [3] [4] [5]

However, building quantum computers involves immense technological challenges~~: quantum states are intrinsically more fragile and can lose their coherence easily~~. Over the past couple of decades, experimentalists have constructed small scale quantum computers of various types, including superconductors or trapped ions. Even though tremendous efforts have been invested to develop the much-desired quantum computing technology, the ultimate “holy grail” of scalable quantum computer available for regular users is still far from practically feasible. Therefore, a natural question arises, how well can we represent and simulate quantum states using classical computers?

The fundamental challenge to describe quantum many-body problems arises from the feature that the Hilbert space dimension grows exponentially with the size of quantum system, making it prohibitively hard even for the most powerful classical computer to fully represents the states of the system. However, recently researchers have explored using modern cutting-edge machine learning techniques, particularly artificial neural networks (ANN) to encode quantum systems. For instance, Carleo and Troyer introduced a variational presentation of quantum states based on ANN with variable number of hidden neurons [6]. Gao and Duan demonstrated the advantage of deep over shallow neural network on representing most physical states including the ground state of many-body Hamiltonians [7]. Rocchetto, Grant, Strelchuk, Carleo and Severini have employed variational autoencoders (VAE), a type of generative AI model to efficiently learn states that are easy to simulate classically and can also compress and reduce the representation dimensions for states that are very hard for classical computers [8].

In this paper, we will employ generative AI models, specifically, VAE and normalizing flows (NF) to learn certain many-body distributions generated from quantum devices that aim to demonstrate quantum advantage. Two types of hard quantum distributions will be studied: one generated by quantum Fourier sampling by Fefferman and Umans [9], and the other generated by qsim quantum circuits [2] [10]. Furthermore, we will analyze the performance of VAE and NF learning.

2. TECHNICAL BACKGROUND OF QUANTUM COMPUTING

The elementary building block of quantum computing is the quantum bit (qubit), the quantum version of a classical bit.

Unlike classical bits that can only be in a binary physical state of 0 or 1, qubits can be in a superposition state of |0⟩ and |1⟩ [11], denoted by Eq. (1).

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \|\alpha\|^2 + \|\beta\|^2 = 1 \qquad (1)$$

For a qubit in superposition, its state is not fully known until it is measured; upon measurement, the wave function of the qubit collapses into one of two states based on probability. Entanglement of multi qubits in superposition states makes their computational power exponentially greater than their classical counterparts.

The states of a qubit can be illustrated on a Bloch sphere, named after physicist Felix Bloch [12], as shown in Figure 1~~Figure 1~~.

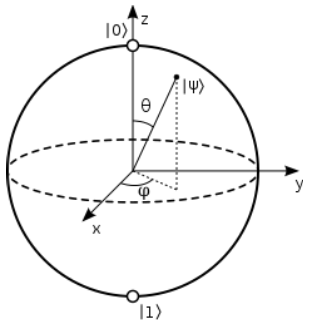


Figure 1. Bloch sphere depicting the states and positions of a qubit.

The evolution of qubits can be mathematically described as unitary operation in Hilbert space, which can be illustrated as the rotation around the x, y, and z axes by means of quantum gates or matrix operations. A sample operation of rotating a single qubit around x axis is shown as the unitary matrix in Eq. (2).

$$R_x(\gamma) = \begin{bmatrix} cos\left(\frac{\gamma}{2}\right) & -isin\left(\frac{\gamma}{2}\right) \\ -sin\left(\frac{\gamma}{2}\right) & cos\left(\frac{\gamma}{2}\right) \end{bmatrix} \qquad (2),$$

Where γ is the rotation angle, and is in the range 0 to 2π.

~~Gamma is in range 0 to 2pi~~

The benefit of exponentially growing power makes quantum computing one of the most active science research fields currently. A famous example demonstrating the advantage of quantum over classical computer is the task of factoring a large integer with *n* digits. The time complexity of any best classical algorithm is *O(exp(n))*; whereas Shor has demonstrated that this problem can be solved by quantum computer in polynomial time [13]. The term, quantum supremacy, refers to this type of advantage and dominance of quantum over classical computing.

~~Add something about how quantum computers don't compute things exactly (approximation)~~

Formatted: Font color: Auto

Formatted: Font: (Default) Times New Roman, Italic, Font color: Auto

Formatted: Font: (Default) Times New Roman, Font color: Auto

Formatted: Font: (Default) Times New Roman, Italic, Font color: Auto

Formatted: Font: (Default) Times New Roman, Font color: Auto

Formatted: Font: (Default) Times New Roman, Italic, Font color: Auto

Formatted: Font: (Default) Times New Roman, Font color: Auto

Formatted: Font color: Auto

Formatted: Body Text

A key challenge for quantum computers is quantum decoherence [14], which describes the phenomenon that quantum states are intrinsically more fragile and can lose their coherence by transitioning to classical states. Entanglement of superposition states provides the foundation of quantum computing, however, decoherence causes qubits to lose their existence in superposition of states from minuscule environmental disturbance. Quantum decoherence is a process of information loss from a system to the environment in a short time frame [15]; it can leads to substantial computation errors. Increasing qubit coherence time [16] and quantum error correction [17] are the key resolutions to overcome this fundamental decoherence problem.

3. Hard Quantum Distributions

We will study two types of quantum distributions in this paper.

3.1. Distribution generated by quantum Fourier sampling by Fefferman and Umans [9]~~[9]~~

The first is a class of hard states introduced by Fefferman and Umans [9], produced by a certain quantum computational process. As pointed out in section 4 of ref. [9], a straightforward implementation of sampling from this distribution is to apply quantum Fourier transformation to a uniform superposition encoded by a function *h* related to the matrix permanent, and then perform the standard basis measurement. On the other side, there also exist a classical implementation of the sampling as described in ref. [8]. It is shown that such classical sampling algorithms are hard because it contains the matrix permanent problem which is not possible to compute in polynomial time. A description of matrix permanent is included in Section 6.2.

The distribution that we will study originated from ref Fefferman and Umans [9].For the purpose of exposition, we will restate the construction process as below.

Start with *m* monomials and an encoding function $h\colon [m] \rightarrow \{0,1\}^n$, an efficiently computable polynomial is defined as below:

$$Q(X_1, X_2, \dots, X_n) = \sum_{z \in [m]} {X_1}^{h(z)_1} {X_2}^{h(z)_2} \dots {X_n}^{h(z)_n}, \qquad (3)$$

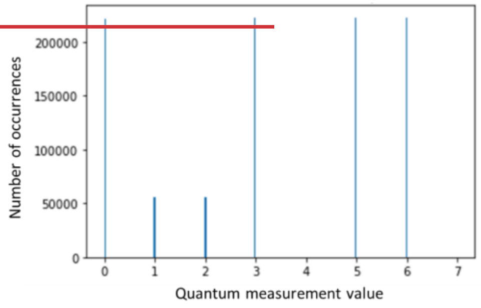
Where $h(z)_i$ means taking the *i-th* bit. In this research, *h* is chosen to map the *i-th* permutation out of *n!* choices, which is related to the matrix permanent as pointed out in ref. [8]. Specifically, in this paper, $h\colon [0, n! - 1] \rightarrow \{0,1\}^{n^2}$.

Given a fixed integer L and a set vector $y = (y_1, y_2, \dots, y_n) \in [0, L-1]^n$. Construct another vector $Z_y = (z_{y_1}, z_{y_2}, \dots, z_{y_n})$ such that z_{y_i} equals to the complex *L-ary* root of 1 raised to power of y_i. Finally, the hard quantum distribution is defined as below.

$$P[y] = \frac{|Q(Z_y)|^2}{L^n n!}, \qquad (4)$$

Plots depicting the empirical probability distribution generated by above quantum process with 3 and 8 qubits are illustrated in Figure 2~~Figure 2~~ and Figure 3~~Figure 3~~ respectively. The values of x-coordinate of Figure 2~~Figure 2~~ range from *0* to *7*, corresponding to the possible measurement outcomes from *0* to $2^3 - 1$ on a 3-qubit quantum system. Similarly, The values of x-coordinate of Figure 3~~Figure 3~~ range from *0* to *255*, which equals $2^8 - 1$ on an 8-qubit system.

~~x-axis: quantum measurement outcomes (make binary string clear)~~



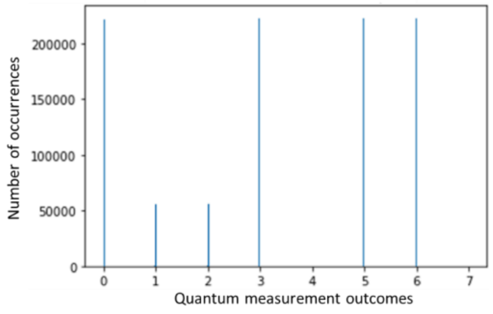
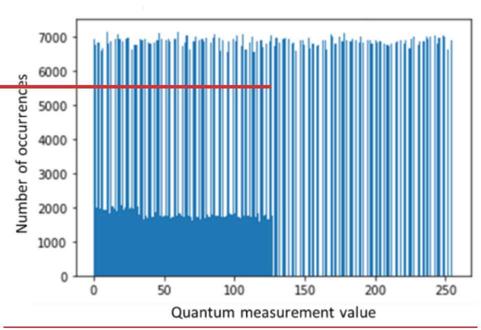


Figure 2. Distribution generated by the quantum Fourier sampling circuit on 3-qubit quantum computer proposed by Fefferman and Umans [9]

Formatted: Font color: Auto



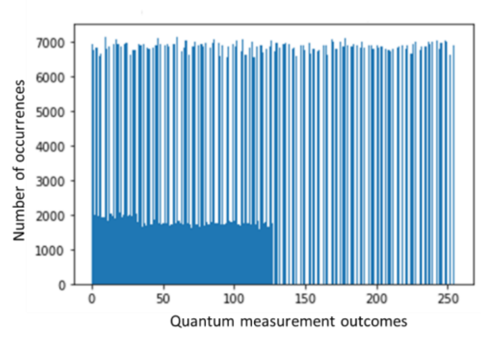


Figure 3. Distribution generated by the quantum Fourier sampling on 8-qubit quantum computer proposed by Fefferman and Umans [9]

3.2. Hard distribution generated by qsim quantum circuits

We will study another type of distribution which is generated from a grid of quantum circuits assembled with certain randomness, as described in ref. [10]. The quantum circuit is illustrated in Figure 4~~Figure 4~~, which is taken from ref. [2] and [10].

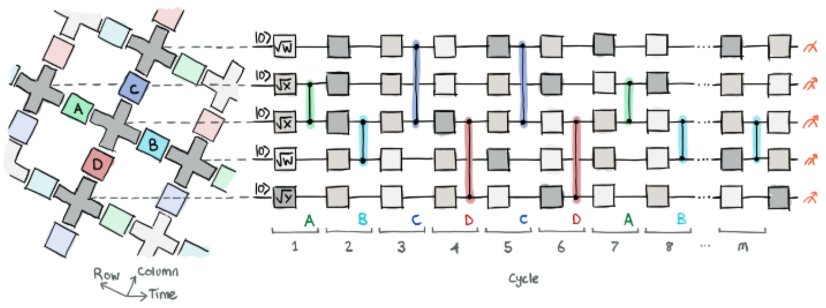


Figure 4. Qsim quantum circuit ~~assembly~~ taken from ref. [2] and [10]

Sampling of this distribution is implemented using pseudo-quantum simulator *qsim* [18] [19] via *PennyLane-Cirq* [20] plugin in Python. *Qsim* is a full wave function simulator written in C++, developed by Google QuantumAI team. It is powerful enough to run simulations of up to 40 qubits on a 90 core Intel Xeon workstation [18].

Quantum gates involved in this circuit include *RX(π/2)* and *RY(π/2),* and the $\sqrt{W}$ gate, where $W = \frac{X+Y}{2}$. Other 2-qubits gates used in this circuits are *SISWAP* and *CPhase:*

$$SISWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{i}{\sqrt{2}} & 0 \\ 0 & \frac{i}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

$$CPhase = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{-i\varphi} \end{bmatrix} \quad (6)$$

Plots depicting this empirical distribution generated by above qsim circuit with 8 and 12 qubits are illustrated in

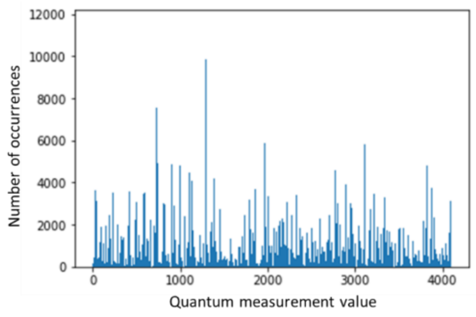


Figure 5~~Figure 5~~ and

Figure 6~~Figure 6~~ respectively. The values of x-coordinate of

Figure 5~~Figure 5~~ range from *0* to *255*, corresponding to the possible measurement outcomes from *0* to $2^8 - 1$ on an 8-qubit quantum system. Similarly, The values of x-coordinate of

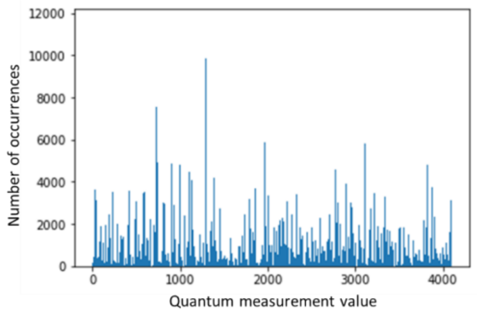


Figure 6~~Figure 6~~ range from *0* to *4091*, which equals $2^{12} - 1$ on a 12-qubit system.

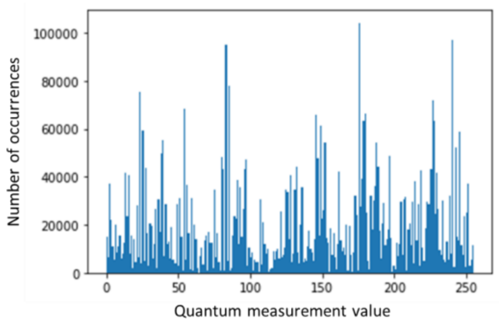


Figure 5. Quantum distribution generated from measuring a 8-qubit system by executing on the qsim circuits proposed in ref. [2] and [10]

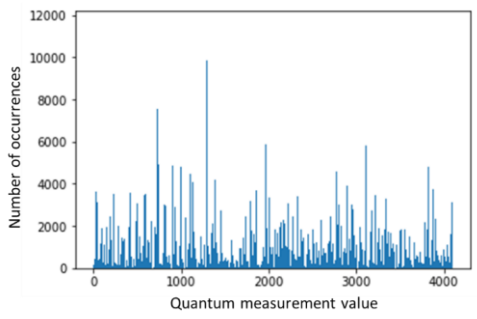


Figure 6. Quantum distribution generated from measuring a 8-qubit system by executing on the qsim circuits proposed in ref. [2] and [10]

4. Generative neural networks

In this research we explore two types of generative artificial intelligence models, variational autoencoder and normalizing flows, to learn hard quantum distributions.

4.1. Variational autoencoder

Variational autoencoder (VAE) is a generative artificial intelligence method initially proposed by Kingma and Welling [21]. VAE is based upon autoencoder, which is a type of unsupervised model which performs efficient dimension reduction. The autoencoder architecture is illustrated in Figure 7~~Figure 7~~. The encoder network compresses input data to a much smaller latent space which captures the essential data structure; then a decoder network reconstructs the output through a reverse process.

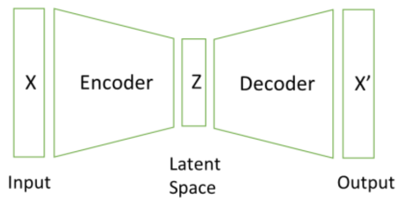


Figure 7. Illustration of an autoencoder

VAE takes autoencoder to the next level; it is capable of generating new contents by introducing a random sampler process in the latent space; such sampling is usually set to Gaussian. Therefore, VAE provides a statistical description of each latent space value. Such probabilistic nature allows VAE to generate many creative and diverse outputs. The architecture of VAE is illustrated in Figure 8~~Figure 8~~.

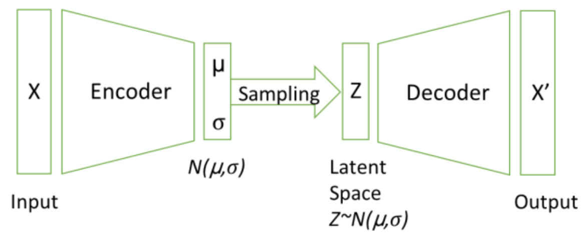


Figure 8. Illustration of a VAE

In order for the decoder to generate meaningful outputs, it is critical for latent space to be regularized. In practice, such regularization is achieved by adding a Kullback-Leibler divergence term to the loss function. So, the total VAE loss function is expressed as Eq. (7).

$$L = \|x - x'\|^2 + D_{KL}[N(\mu, \sigma), N(0,1)] \qquad (7)$$

Ever since its inception a decade ago, VAE has found enormous success in a variety of applications across data and image creation, anomaly detection by understanding the underlying distribution, etc.

4.2. Normalizing flows

A normalizing flow (NF) [22] is a flow-based probabilistic generative model. NF explicitly models a probabilistic distribution by a series of change-of-variable operations. The essence of NF is to gradually transform a simple density through a series of reversible operations to generate a target complex distribution. The initial idea went back to Whitening transformations by Johnson [23] in 1966. The modern term "*Normalizing Flow"* was introduced by Tabak and Turner [24], where they formalized NF as a sequence of invertible simple maps.

Similar to the decoder of a VAE, the series of transformation functions in NF reconstruct the complex target distribution *P(x)* from a simple distribution *P(z),* which is often set to Gaussian. A key feature of these functions is to maintain unit total probability, i.e., $\sum P(x) = 1$. Therefore, it is necessary for all these functions to be bijective, i.e. both surjective (onto) and injective (1 to 1). A natural consequence of using only bijectors is that the latent space of NF must be the same size as the inputs and outputs.

Mathematically, let the bijector be $x = f(z)$, and the inverse to be $z = g(x)$. Probability of x can be expressed as Eq. (8):

$$P(x) = P\big(g(x)\big)|det\ (J_g)| \qquad (8)$$

Where the last term $det\ (J_g)$ is the determinant of the Jacobian of $g(x)$. The Jacobian is a matrix of derivatives of z with respect to x, as expressed in Eq. (9):

$$J_g = \begin{pmatrix} \frac{\partial z_1}{\partial x_1} & \cdots & \frac{\partial z_1}{\partial x_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial z_N}{\partial x_1} & \cdots & \frac{\partial z_N}{\partial x_N} \end{pmatrix} \qquad (9)$$

It can be seen that invertibility and differentiability are the must-have features for all NF bijectors, in order to ensure the applicability of the change-of-variable theorem and computation of the Jacobian.

What makes NF really powerful is to chain together multiple bijectors, similar to multiple layers of a deep neural network. Suppose $x = f_1(z_1)$, $z_1 = f_2(z_2)$, ..., $z_{N-1} = f_N(z_N)$; and the inverse function be $z_N = g_N(z_{N-1})$, ..., $z_2 = g_2(z_1)$, $z_1 = g_1(x)$. Then Eq. (8) becomes Eq. (10) as below.

$$P(x) = P\left(g_N\left(g_{N-1} \dots \left(g_1(x)\right)\right)\right) |det\,(J_{g_1})||det\,(J_{g_2})| \dots |det\,(J_{g_N})| \qquad (10)$$

Trainable parameters of a NF are these bijectors. The loss function of NF is defined as the negative log-likelihood as Eq. (11) in a very straightforward way:

$$L = -\,log(P(x)) = -\,log\left[P\left(g_N\left(g_{N-1} \dots \left(g_1(x)\right)\right)\right)\right] - \textstyle\sum_i log\,(|det\,(J_{g_i})|) \qquad (11)$$

NF has become an extremely powerful generative tool with wide applications across density estimation, data, image, audio and video generation, anomaly detection, etc.

Comparing NF to other generative models like VAE and generative adversarial network (GAN), NF provides the advantage of directly modeling the probability density and likelihood which is very hard for VAE and GAN. On the other side, the latent space of NF must keep the same dimension, therefore NF does not allow for data compression which can be done by VAE easily.

5. Results

We have generated two types of hard distributions from quantum Fourier sampling as proposed by Fefferman and Umans [9] and by execution on the qsim circuit as in ref. [2] [10]. We will now study the difficulty of learning these distributions using the sophisticated machine learning tools, VAE and NF, as described in previous section.

5.1. VAE study of hard quantum distributions

We code the VAE network using the Keras/Tensorflow framework [25]. Tensorflow is a widely used open-source software for machine learning, developed by the Google Brain team. Tensorflow is available for various computer programming languages, including C++, Java, Python, etc. For our paper, all the coding is done in Python.

Following the steps described in section 4.1, for our VAE network, we chose the latent space dimension   to be 8. A high-level architecture diagram of this VAE network is illustrated in Figure 9~~Figure 9~~. The encoder network is set as two dense layers with *ReLU* activation

Formatted: Font: Not Italic, Check spelling and grammar functions. The decoder network is exactly reverse of encoder. Total number of trainable parameters of this VAE is 5,014,152, making it trainable on the CPU of a PC.

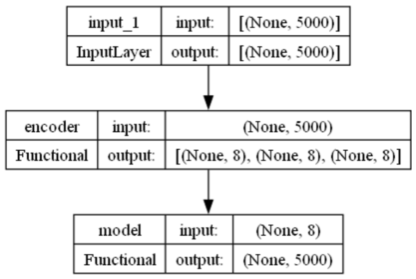
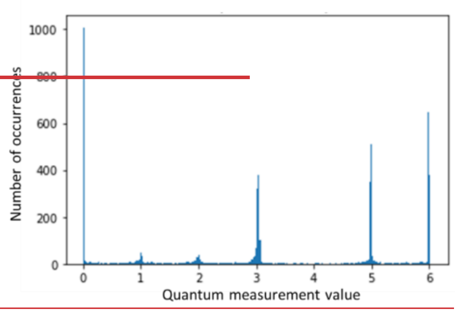


Figure 9. High level architecture of our VAE network. The top boxes show the input layer, where the distribution data are assembled in groups of 5000 sample each. The middle boxes show the encoder layer, which consists of two linear dense layers with ReLU activation function and dimension of 500 and 8 respectively. Random sampling of the latent space results in 3 encoder outputs: the mean μ, the standard deviation σ, and the Gaussian sample z; each of them has the same dimension 8 as the latent space. The bottom boxes show the decoder layer, which has exactly the reverse structure of the encoder network. Details inside encoder and decoder are not shown to keep the size of this diagram fit here.

We train VAE with the hard quantum distributions as described in Section 3.1 and 3.2 respectively. After the training, Gaussian samples are drawn from the latent space, and then passed through the decoder to reconstruct the desired distributions.

Figure 10~~Figure 10~~ and Figure 11~~Figure 11~~ show the VAE reconstruction results for the quantum Fourier sampling on a 3 and 8-qubit quantum computer proposed by Fefferman and Umans as described in Section 3.1. . Comparing them with Figure 2~~Figure 2~~ and Figure 3~~Figure 3~~, it can be easily observed that VAE performs well on learning the simpler distribution generated by quantum device with 3 qubits, but struggles to learn the essential pattern of the more complex 8-qubits data.



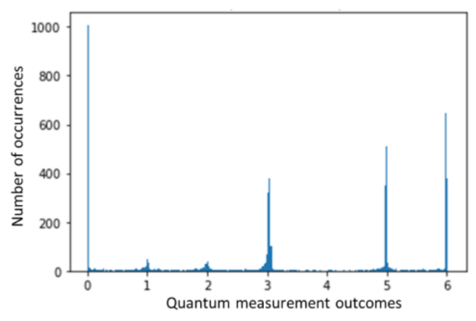
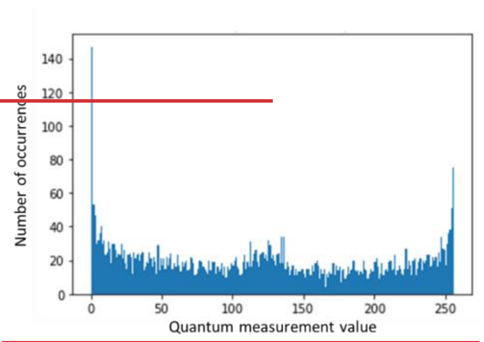


Figure 10. VAE reconstruction of quantum Fourier sampling on a 3-qubit quantum computer proposed by Fefferman and Umans [9]~~[ref]~~



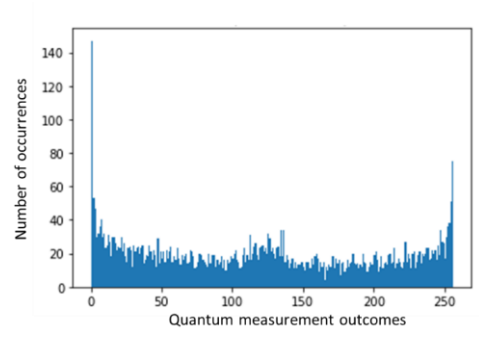


Figure 11. VAE reconstruction of quantum Fourier sampling on an 8-qubit quantum computer proposed by Fefferman and Umans [9]~~[ref]~~

VAE reconstruction results of quantum distributions generated by execution on 8 and 12-qubit qsim circuits are shown in Figure 12~~Figure 12~~ and Figure 13~~Figure 13~~. Compared to

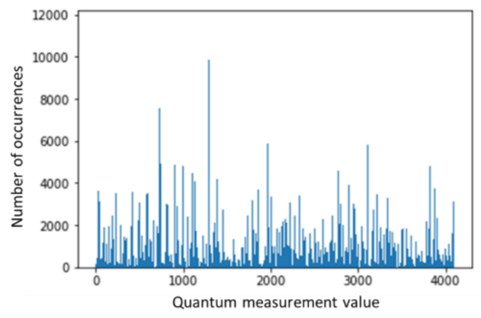
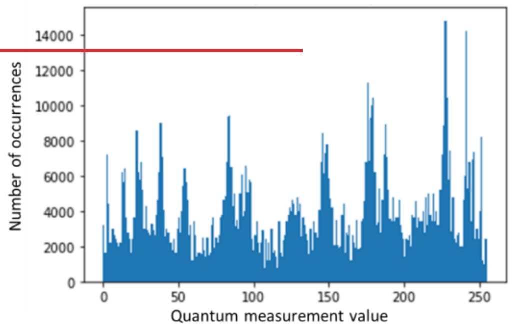


Figure 5~~Figure 5~~ and

Figure 6~~Figure 6~~, VAE performed reasonably fine on learning the distributions; however substantial discrepancies are still notable.



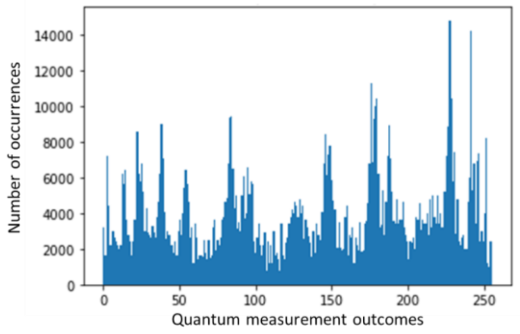
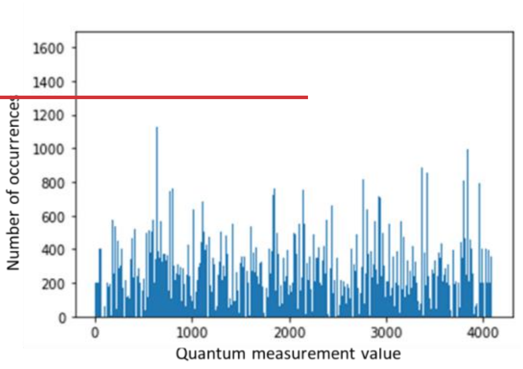


Figure 12. VAE reconstruction of quantum distributions generated by execution on an 8-qubit qsim circuit proposed in ref. [2] [10]



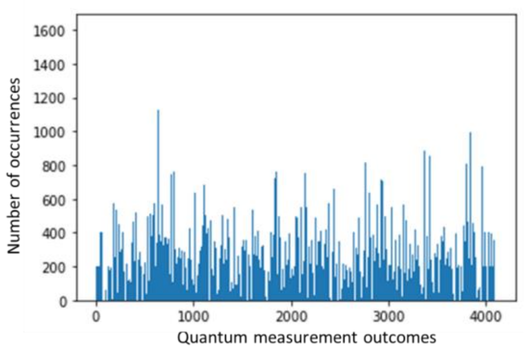


Figure 13. VAE reconstruction of quantum distributions generated by execution on a 12-qubit qsim circuit proposed in ref. [2] [10]

5.2. NF study of hard quantum distributions

We build the NF network using the Tensorflow_Probability (TFP) framework [26]. TFP is a python library built on TensorFlow and combines deep learning with probabilistic modeling. Following the steps described in Section 4.2., we coded the NF network in Python/TFP by following similar steps as used by White [27]. As described in Section 3.2, the key components of NF are invertible and differentiable bijectors. To chain up these bijectors, we employ a masked autoregressive network which is ready to use in TSP. An autoregressive network satisfies the property that, for each dimension i of the data, distribution of x_i is only dependent on x_j for $0 \leq j < i$. The benefit of autoregressive network is so that the Jacobian matrix is always lower-diagonal, ensuring super-fast computation of the determinant. Therefore, autoregressive network allows computation-efficient forward transformation, making it a popular choice of NF training.

To set up the NF network used in this paper, we chain up five layers of masked autoregressive networks. The loss function is set as the negative log probability as in Eq. (11) to train this NF. The layers of this NF network consist of a series of AutoregressiveNetworks interconnected by TFOpLambda operators. The total number of trainable parameters of this VAE is 98,640, again making it trainable on the CPU of a PC. The architecture diagram generated by Keras.utils.plot_model() is too large to fit in one page; but it can be found on github [28].

We train this NF network with the hard quantum distributions as described in Section 3.1 and 3.2 respectively. Afterwards, the reconstruct distributions are drawn from the trained NF.

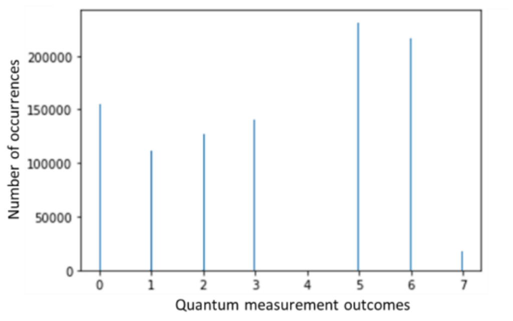


Figure 14~~Figure 14~~ and Figure 15~~Figure 15~~ show the NF reconstruction results of the quantum Fourier sampling on 3 and 8-qubit quantum computers proposed by Fefferman and Umans as described in Section 3.1. A close-up comparison with Figure 2~~Figure 2~~ and Figure 3~~Figure 3~~ shows that NF learned the overall distribution of both 3 and 8 qubits distributions; but it misses certain granular state features. Notably,

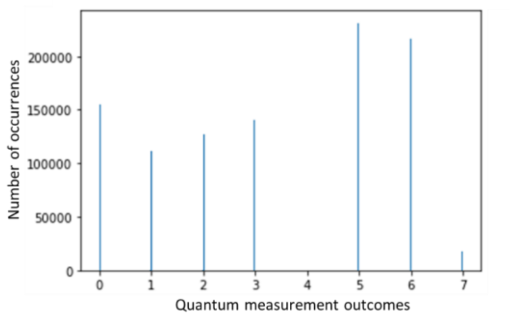
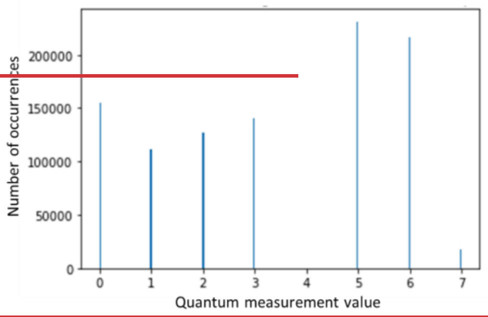
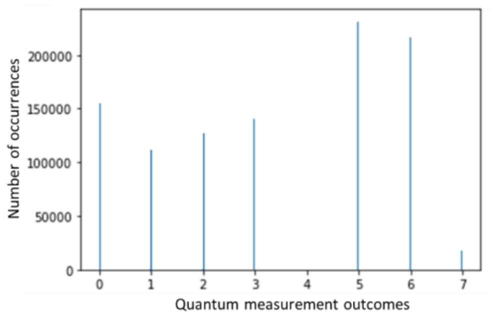


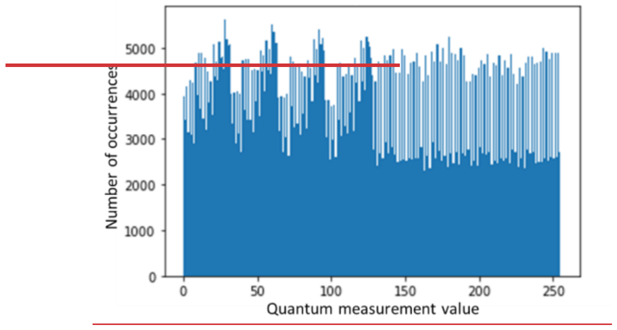
Figure 14~~Figure 14~~ overstates the probability of position 1 and 2 on the left side of Figure 2~~Figure 2~~; and Figure 15~~Figure 15~~ misses the probability annihilation for certain discrete numbers on right side of Figure 3~~Figure 3~~.





~~Bolder 1 & 2?~~

Figure 14. NF reconstruction of quantum Fourier sampling on a 3-qubit quantum computer proposed by Fefferman and Umans [9]



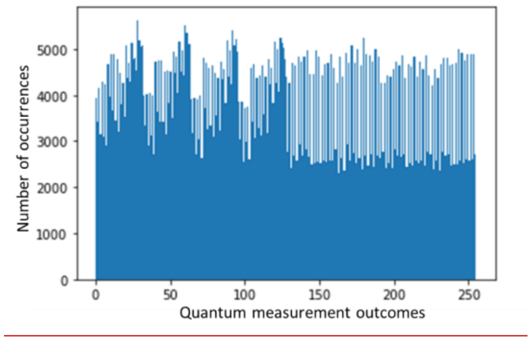


Figure 15. NF reconstruction of quantum Fourier sampling on an 8-qubit quantum computer proposed by Fefferman and Umans [9]

NF reconstruction results of quantum distributions generated by execution from 8 and 12-qubit qsim circuits are shown in Figure 16~~Figure 16~~ and Figure 17~~Figure 17~~ . Comparing to

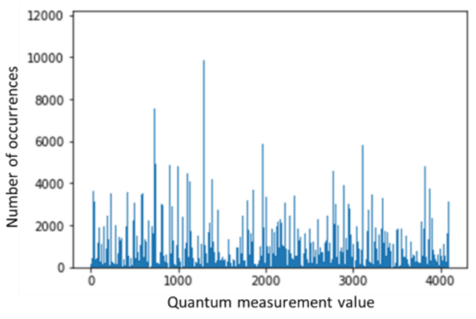
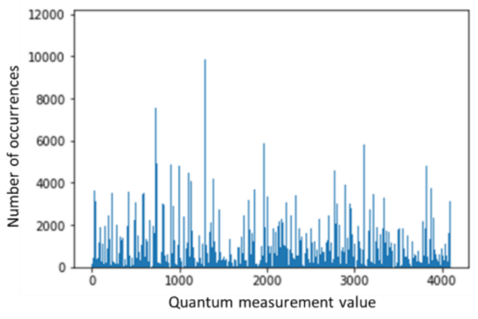


Figure 5~~Figure 5~~ and

Figure 6~~Figure 6~~, substantial discrepancies are observed. For the 8-qubit data, Figure 16~~Figure 16~~ captures the feature of concentration of measurement results toward the right-hand side of

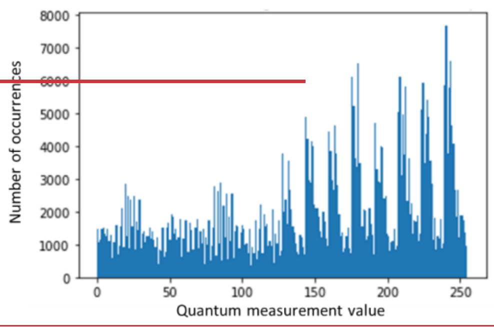
Figure 5~~Figure 5~~, notably around the measurement values around 150, 180, and 230. However, it misses the spikes around measurement values of 25 and 80 of

Figure 5~~Figure 5~~. Similarly for the 12-qubit data, Figure 17~~Figure 17~~ shows a more even



distribution and misses the spikes in

Figure 6~~Figure 6~~. The results suggest that even though NF is capable of learning the overall shape of the distributions, it looks to lack the capability to capture the more granular features existent in these hard quantum distributions.



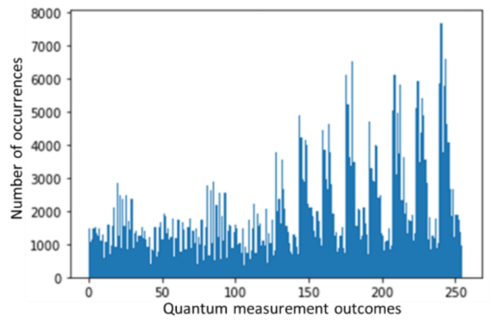
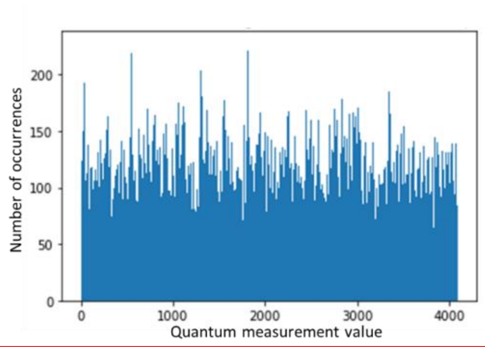


Figure 16. NF ~~reconstruction of quantum distributions~~ generated by execution on an 8-qubit qsim circuit proposed in ref. [2] [10]



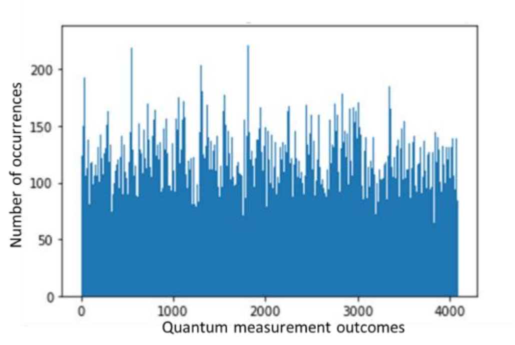


Figure 17. NF reconstruction of quantum distributions generated by execution on a 12-qubit qsim circuit proposed in ref. [2] [10]

6. Conclusions and Discussions

6.1. Performance analysis of VAE and NF

Two types of hard distributions are generated by following the quantum Fourier sampling as proposed by Fefferman and Umans [9] and by execution on the qsim circuit as in ref. [2] [10]. We have employed two modern machine learning techniques, VAE and NF, to learn these hard quantum distributions. It is interesting to compare the performance of VAE and NF. VAE, as a more mature model extensively explored for image generation, only requires a much smaller size of latent space. Therefore, training of VAE is usually more stable and uses less computer resource. On the other hand, NF, as a newer flowed base model, keeps the latent space the same size as inputs, therefore it is capable of obtaining an exact estimate of likelihood of the samples. However, training of NF is typically less stable and more computing resource heavy. In practice of this research, we indeed encountered NaN (Not-a-Number) error during the NF debugging processes, which indicates that the loss function diverges. These errors can be fixed by fine tuning the hyperparameters of the NF networks

Looking at the VAE and NF results together, we observe that VAE performs reasonably well except for the 8-qubits quantum distribution related to the permanent as in Figure 11~~Figure 11~~. Interestingly, NF demonstrates the better performance for this quantum distribution as in Figure 15~~Figure 15~~. The complementary strength of VAE and NF shall be worth further exploration.

It is worth noting that the networks in this research have been set up to be sufficiently small so it can be trained on the CPU of a personal computer. A much larger classical neural network will apparently be better at learning. Actually, if the number of neurons grows at exponential speed as the number of qubits, it will not only be able to learn the input sample, but also overfit the distribution certainly. The question is, will a classical neural network be able to learn quantum distribution with polynomial instead of exponential growing number of neurons? If so, what type of quantum distributions can it learn, and what types of distribution it lacks the capability?

Why are the generative AI models, VAE and NF, only capable of capturing partial features of these quantum distributions? Will their performance be significantly enhanced by revising the architecture of these generative NN models, adjusting hyperparameters, and increasing the amount of training data? Or will the performance of these classical NNs be inherently limited due to lack of expressibility when encountering the complex quantum states with exponential growing Hilbert space? The answer is still far from certain. But our study seems to suggest the latter case.

6.2. Other quantum distributions

The approach of this paper can certainly be applied to other complex quantum problems. Another good candidate is boson sampling, which evaluates the expected values of permanents of matrices by sampling of bosons scattered by a linear interferometer, proposed by Aaronson and Arkhipov [29]. The photonic implementation of boson sampling is regarded as the most promising approach toward building linear optical quantum computing devices. In brief, for a system of N photons, a typical simplified form for the probability of detecting t_k photons at the k^{th} mode is:

$$p(t_1, t_2, \ldots, t_N) = \frac{|Perm\ U_T|^2}{\prod_{i=1}^{N} t_i!}, \qquad (12)$$

~~(Define perm mathematically, which closely related to the definition of determinant)~~

w~~W~~here *Perm* stands of the permanent of matrix U_T. For an $N \times N$ matrix $A = (a_{ij})$, Perm is defined as

$$Perm(A) = \sum_{\sigma \in S_N} \prod_{i=1}^{N} a_{i,\sigma(i)}, \qquad (13)$$

where the summation of σ is over all permutations of *1, 2, …, N*.

The definition of *Perm* is very similar to the determinant, except that there are no alternative positive and minus signs in the permutations. However, the computation complexity of *Perm* is much larger than that of the determinant; specifically, complexity of Perm is *O(exp(N))*, compared to *O(N^3)* for the determinant.

U is an $N \times N$ unitary matrix defined by the interferometer; U_T is transformed from *U* by keeping the first *M* columns and repeating times of the *j-th* row. This problem is *#P-hard* for classical computers given the existence of the matrix permanent problem. In fact, the complexity of the best classical algorithm for exact boson sampling is *O(N2^N+MN^2)* for a system with N photons and M outputs modes. Various approaches of realizing boson sampling have been proposed, including the most natural photonic scattering, or using the superconducting resonator device as the interferometer, etc. Learning boson sampling distribution using VAE, NFs and other modern generative NN models will be important both from theoretical and practical

Formatted: Centered

Formatted: Font: Italic

Formatted: Font: Italic

Formatted: Font: Not Italic

Formatted: Font: Italic

Formatted: Font: Italic

Formatted: Font: Italic

Formatted: Font: Italic, Superscript

Formatted: Font: Italic

perspectives. If certain classical NNs with polynomial number of latent space variables are found capable of learning the essentials, significant progress shall happen to study of molecular vibronic spectra, a practical problem closely related to boson sample. Nevertheless, this is beyond the scope of this paper, so we will leave it for further research in the future.

7. SUPPLEMENTARY~~AL~~ MATERIALS

The computer code for building VAE and NF networks, generating quantum distributions and simulations results are available on github [28]; the code can also be provided upon reasonable request.

Add run time/what computer this was run on

Add instructions about how to run the files on github (libraries, what order, etc.)

BIBLIOGRAPHY

[1] R. Feynman, "Simulating Physics with Computers," *International Journal of Theoretical Physics,* vol. 21, no. 6/7, pp. 467-488, 1982.

[2] F. Arute, K. Arya, R. Babbush and e. al., "Quantum supremacy using a programmable superconducting processo," *Nature,* vol. 574, pp. 505-510, 2019.

[3] Y. Kim, A. Eddins, S. Anand, K. X. Wei, E. van den Berg, S. Rosenblatt, H. Nayfeh, Y. Wu, M. Zaletel, K. Temme and A. Kandala, "Evidence for the utility of quantum computing before fault tolerance.," *Nature,* vol. 618, pp. 500-505, 2023.

[4] L. Huang, H. Zhou, K. Feng and C. Xie, "Quantum random number cloud platform," *NPJ Quantum Information,* vol. 7, p. 107, 2021.

[5] A. E. B. R. S. S. I. T. S. G. A. M. B. L. B. A. Z. D. L. F. S. M. G. R. J. S. M. H. D. V. Sivak, "Real-time quantum error correction beyond break-even," *Nature,* vol. 616, pp. 50-55, 2023.

[6] G. Carleo and M. Troyer, "Networks, Solving the Quantum Many-Body Problem with Artificial Neural," *Science,* vol. 355, no. 6325, pp. 602-606, 2017.

[7] X. Gao and L.-M. Duan, "Efficient representation of quantum many-body states with deep neural network," *Nature Communications,* vol. 8, p. 662, 2017.

[8] A. Rocchetto, E. Grant, S. Strelchuk, G. Carleo and S. Severini, "Learning hard quantum distributions with variational," *Nature, NPJ Quantum Information,* vol. 4, no. 28, 2018.

[9] B. Fefferman and C. Chris Umans, "The Power of Quantum Fourier Sampling," *arXiv:1507.05592,* 2015.

[10] T. Isacsson, "Beyond classical computing with qsim," 2023. [Online]. Available: https://pennylane.ai/qml/demos/qsim_beyond_classical/.

[11] Y. Kwak, W. J. Yun, S. Jung and J. Kim, Quantum Neural Networks: Concepts, Applications, and Challenges., 2021.

[12] F. Bloch, "Nuclear induction," *Physical Review,* vol. 70, no. 7-8, pp. 46-474, 1946.

[13] P. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 1994.

[14] D. Bacon, "Decoherence, control, and symmetry in quantum computers," *arXiv,* 2001.

[15] W. H. Zurek, "Decoherence, einselection, and the quantum origins of the classical," *Reviews of Modern Physics,* vol. 75, no. 3, p. 715, 2003.

[16] M. Beau, J. Kiukas, I. L. Egusquiza and A. del Campo, "Nonexponential quantum decay under environmental decoherence," *Phys. Rev. Lett,* vol. 119, p. 130401, 2017.

[17] W. M. Y. W. W. Z. C. S. L. Cai, "Bosonic quantum error correction codes in superconducting quantum circuits," *Fundamental Research,* vol. 1, no. 1, pp. 50-67, 2021.

[18] Google Quantum AI Team, "https://quantumai.google/qsim," [Online]. Available: https://quantumai.google/qsim.

[19] Quantum AI team and collaborators, "qsim," 2020. [Online]. Available: https://github.com/quantumlib/qsim.

[20] Cirq Developers, "Cirq," [Online]. Available: https://github.com/quantumlib/Cirq/graphs/contributors, https://github.com/quantumlib/Cirq.

[21] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *arXiv:1312.6114,* 2013.

[22] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed and B. Lakshminarayanan, "Normalizing flows for probabilistic modeling and inference," *Journal of Machine Learning Research,* vol. 22, pp. 1-64, 2021.

[23] R. Johnson, "The minimal transformation to orthonormality," *Psychometrika,* vol. 31, pp. 61-66, 1966.

[24] E. G. Tabak and C. V. Turner, "A family of nonparametric density estimation," *Communications on Pure and Applied Mathematics,* vol. 66, no. 2, pp. 145-164, 2013.

[25] Google Brain Team, "https://www.tensorflow.org/guide/keras," [Online]. Available: https://www.tensorflow.org/guide/keras.

[26] Google Brain Team, "https://www.tensorflow.org/probability," [Online]. Available: https://www.tensorflow.org/probability.

[27] A. D. White, "Deep Learning for Molecules and Materials," *Living Journal of Computational Molecular Science,* vol. 3, no. 1, p. 1499, 2021.

[28] E. Wang, "https://github.com/Watermelon-Addict/Learning-hard-quantum-distributions-with-VAE-and-NF," [Online]. Available: https://github.com/Watermelon-Addict/Learning-hard-quantum-distributions-with-VAE-and-NF.

[29] S. Aaronson and A. Arkhipov, "The Computational Complexity of Linear Optics," *Theory of Computing,* vol. 9, pp. 143-252, 2013.