Impact of the form of weighted networks on the quantum extreme reservoir computation

Aoi Hayashi,[1,2,3,*] Akitada Sakurai,[2,3] Shin Nishio,[1,2,3] William J. Munro,[2,3,4] and Kae Nemoto[1,2,3,†]

[1]*School of Multidisciplinary Science, Department of Informatics, SOKENDAI (the Graduate University for Advanced Studies), 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan*

[2]*Okinawa Institute of Science and Technology Graduate University, Onna-son, Okinawa 904-0495, Japan*

[3]*National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan*

[4]*NTT Basic Research Laboratories & Research Center for Theoretical Quantum Physics, 3-1 Morinosato-Wakamiya, Atsugi, Kanagawa, 243-0198, Japan*

(Received 14 November 2022; revised 22 July 2023; accepted 18 September 2023; published 16 October 2023)

The quantum extreme reservoir computation (QERC) is a versatile quantum neural network model that combines the concepts of extreme machine learning with quantum reservoir computation. Key to QERC is the generation of a complex quantum reservoir (feature space) that does not need to be optimized for different problem instances. Originally, a periodically driven system Hamiltonian dynamics was employed as the quantum feature map. In this work we capture how the quantum feature map is generated as the number of time-steps of the dynamics increases by a method to characterize unitary matrices in the form of weighted networks. Furthermore, to identify the key properties of the feature map that has sufficiently grown, we evaluate it with various weighted network models that could be used for the quantum reservoir in image classification situations. At last, we show how a simple Hamiltonian model based on a disordered discrete time crystal with its simple implementation route provides nearly optimal performance while removing the necessity of programming of the quantum processor gate by gate.

DOI: 10.1103/PhysRevA.108.042609

I. INTRODUCTION

In recent years we have seen the steady growth of the number of qubits available on a variety of quantum processors [1–4]. This has lead to the new phase of quantum computer development, often called the "NISQ" era. Here NISQ stands for noisy intermediate-scale quantum, which indicates that the quantum processor is too small to implement logical quantum operations and hence is inherently noisy. The number of qubits in these quantum processors (well in excess of 50 [3,5,6]) has already reached the point where the quantum computational tasks they can perform are intractable in a conventional computer, however, noise prevents us fromextracting the quantum advantage such quantum computer promise. Hence, for the NISQ era to mark its significance in computer history, quantum advantages for real applications have to be demonstrated.

Many of the current NISQ processors are designed to operate via quantum gates [1,7–9]. To run a quantum algorithm, we need to obtain a quantum gate circuit from the quantum algorithm and then decompose each quantum gate into one implementable on the quantum processor at hand. The noise in these quantum processors necessitates the optimization of quantum gate circuits to minimize their effect. As long as the physical qubits are directly used for computation, quantum algorithms also need to be relatively short and resilient to noise. Variational quantum algorithms (VQAs) have attracted a lot of attention from this viewpoint and have been intensively investigated [10–12]. However, there have been several issues with them, with the most significant obstacle being the difficulty in the optimization of the variational models [13,14]. VQAs are a type of model of quantum neural networks (QNNs). It is well known that there are other models for using QNNs. One such example is quantum reservoir computation [15–19], which should be expected to be more implementation friendly. Similarly to the chaotic dynamics used in the (classical) reservoir computation [20,21], the quantum reservoir generates complex dynamics in the quantum system. Realizing such dynamics using a quantum gate circuit approach is, however, not that simple [1,22–25]. We do not require precise programming to generate a sufficient complexity in the quantum reservoir to realize our quantum algorithm [16,18]. Instead, an effective quantum reservoir can potentially be generated by a simpler quantum system giving us a better way to utilize the computational power of QNNs.

Recently the quantum extreme reservoir computation (QERC) was proposed [26] as a more advanced yet simpler QNN model based on reservoir computation and extreme machine learning [27]. This model uses a quantum reservoir to generate a quantum neural network which is then used for extreme machine learning. The use of quantum reservoirs

*aoi.hayashi@oist.jp
†kae.nemoto@oist.jp

for extreme machine learning has been discussed. However, there had been no attempts to image classifications until recent years [19], and the previous models have been presented only in a general form. Unlike the previous models, the QERC was able to perform the MNIST image classification task, which is considered an important task in computer vision. This model has numerically shown to achieve the highest accuracy in classifying handwritten digits using the MNIST dataset with the smallest number of qubits [28]. An interesting feature of this approach is that it utilizes a discrete time crystal (DTC) as the feature map, which is much simpler to implement than the quantum gate circuit needed to generate a random unitary matrix. This suggests that, if we could understand the mechanism associated with using the complexity of the quantum dynamics for generating an effective feature space, it would become possible to design quantum feature maps more efficiently.

One of the versatile methods to study the complexity of quantum dynamics is to characterize it as a complex network. Such network approaches have allowed us to quantify the complexity of quantum states around critical points of quantum phase transitions [29], to build a graphical calculus for Gaussian pure states [30], and to reveal the preferential attachment mechanism during the melting process of the DTC [31]. Furthermore, these network approaches can be also applied to analysis of quantum machine learning models. In Ref. [26], the performance behavior of the QERC with the DTC dynamics was explored with the complex network emerging in the Hilbert space of the DTC dynamics [31].

Now let us outline the focus and structure of this paper. We will use the QERC proposed in Ref. [26] as a tool to investigate the role of the feature map in quantum neural networks. This model provides a convenient platform to do so, as the quantum contribution fully relies on the quantum reservoir. We start in Sec. II with a description of the QERC model. In Sec. II, we also present our method to characterize the unitary map, which is responsible for the quantum reservoir, as a complex network. By our method, we investigate the feature map properties with the DTC and some random unitaries and will observe the difference in their dynamics in Sec. III. Then, in Sec. IV, by benchmarking the QERC performance with those unitaries with concrete practical tasks, we discuss what properties of the performance arises based on the difference in the models' dynamics. We will confirm that the difference is not only the quantum reservoir properties, but also, in fact, can be exploited for a better QERC performance for the practical tasks. In Sec. V, we summarize our results.

II. QERC MODEL AND ITS CHARACTERIZATION

Let us begin with a brief description of QERC. As shown in Fig. 1(a), QERC can be described in terms of three key components: the encoder, the quantum reservoir, and the classical processor.

Encoder: Here, the data to be classified is preprocessed (if necessary) and encoded into the initial state of the quantum reservoir. In more detail, as shown in Fig. 1(a), a principal component analysis (PCA) map is used for the preprocessing of the classical data. Then an appropriate encoding strategy needs to be chosen for the problem at hand. For a quantum reservoir of L qubits, the $2L$ most significant parameters from the PCA map will be encoded by single-qubit rotations.

Quantum reservoir: In this step, the quantum reservoir provides the feature space for QERC. The quantum dynamics of the quantum reservoir determines the feature-map properties for the quantum computation, which is given by the unitary operator $\hat{U}$.

Classical processor: In this final step, the state given by the unitary operator $\hat{U}$ acting on the initial state is measured projectively on the computational basis. The process will be repeated to obtain the amplitude distribution of the state generated by the unitary operator. This amplitude distribution is then processed through a one-layer neural network (ONN).

We immediately notice that this is a hybrid quantum-classical algorithm. In QERC the feature space is provided by the quantum reservoir, whereas the optimization is carried out on the classical processor (ONN). Typically the quantum reservoir does not need to be optimized for different problem instances [15,18,26].

Now our interest in this paper is the properties we require for the quantum reservoir and their influence on the performance of QERC. In particular, we want to show that how we set the quantum reservoir is important. In this work, we first employ the DTC model used in [26] as our choice of a quantum reservoir. The DTC model has a parameter that controls the complexity of the dynamics, namely, starting with the perfect discrete time crystal when the rotation parameter error $\epsilon = 0$, the dynamics gradually deviates from a DTC, acquiring its complexity as ϵ increases. This parameter ϵ represents an error in the single-qubit rotation in the DTC Hamiltonian, which is given by

$$\hat{H}(t) = \begin{cases} \hat{H}_1 = \hbar g(1-\epsilon)\sum_l \hat{\sigma}_l^x, & t \in [0, T/2), \\ \hat{H}_2 = \hbar \sum_{lm} J_{lm}\hat{\sigma}_l^z\hat{\sigma}_m^z + \hbar \sum_l D_l\hat{\sigma}_l^z, & t \in [T/2, T), \end{cases} \tag{1}$$

where $\hat{\sigma}_l^a$ $(a = x, y, z)$ represent the Pauli operators on the lth qubit. Next, T is the cycle of driving, while the DTC cycle is $2T$. Furthermore, g is the rotation strength, and in this case, we set $gT = \pi$. Now $J_{lm} = J_0/|l-m|^\alpha$ is the coupling strength between the qubits l and m with a power-law decay that scales with a constant α. Finally, D_l is a disordered external field for each qubit l. Unless explicitly stated, all the $D_l T$ are set to zero in this work.

The time-periodic system is conveniently characterized by the Floquet operator $\hat{\mathcal{F}} = \exp[-i\hat{H}_2 T/2\hbar]\exp[-i\hat{H}_1 T/2\hbar]$ where the stroboscopic time evolution can be obtained by the unitary operator $\hat{U}(nT, 0) = \hat{\mathcal{F}}^n$ for $n \in \mathbb{N}$. Hence, we use the unitary operator $\hat{U}(nT, 0)$ for different values of n to characterize the quantum reservoir.

A. Characterization of the unitary matrices

The next step is the characterization of the unitary operator $\hat{U}(nT, 0)$. This unitary operator acts as a map between the input and the output states given by the feature map used for

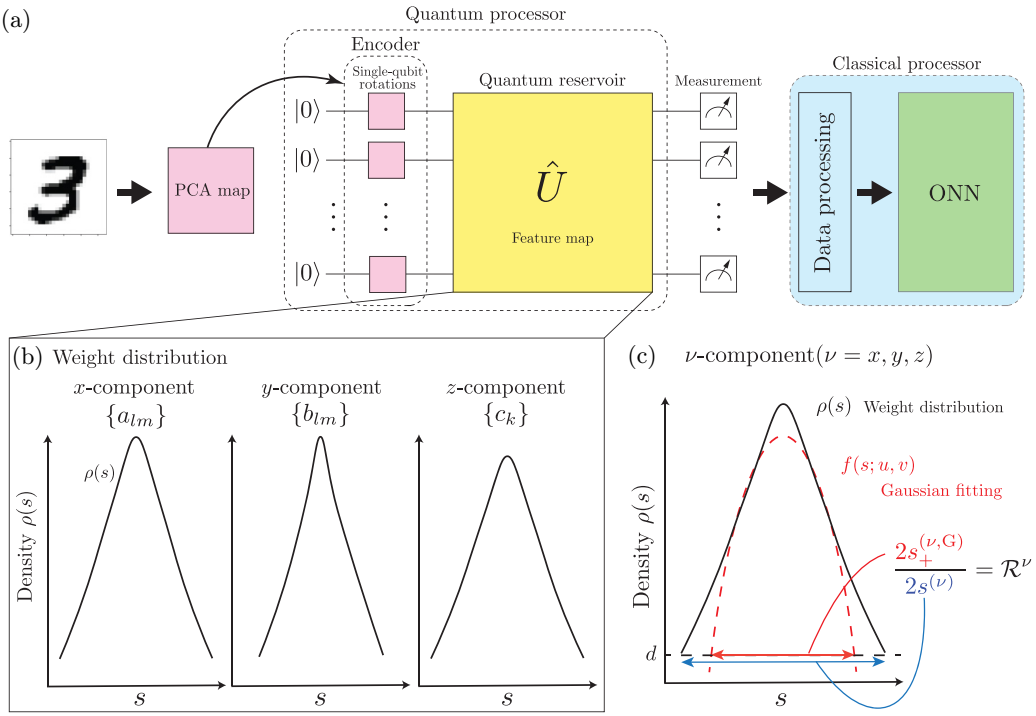

FIG. 1. (a) Schematic architecture of the QERC processor. It begins with an image of size 28×28 pixels, is processed through principal component analysis, and is compressed to $2L$ components (where L is the number of qubits). Using these $2L$ components, an initial state corresponding to the image is created by single-qubit rotations. The quantum reservoir then lets the initial state evolve. By projective measurements on the computational basis, the final state is converted to classical information. The amplitude distribution of this classical information is fed into the ONN. (b) Schematics of the weight distributions for the x, y, and z components. Panel (c) summarizes the definition of the ratio $\mathcal{R}^\nu = s_+^{(\nu,\mathrm{G})}/s^{(\nu)}$ for the ν component, where $s^{(\nu)}$ is given by $s^{(\nu)} = (s_{\max}^{(\nu)} - s_{\min}^{(\nu)})/2$ with $s_{\max\,(\min)}^{(\nu)}$ being the maximum (minimum) value of s in the ν component. Next one can consider a Gaussian fitting function $f(s;u,v) = u\exp(-vs^2)$, which allows us to define $s_+^{(\nu,\mathrm{G})} := \sqrt{\ln(u/d)/v}$ where d corresponds to the possible minimum nonzero height of the density function $\rho(s) = h(s)/N_v ds$, that is, $d = (N_v ds)^{-1}$.

the quantum computation. Such a map can be considered as a weighted network [32,33]. However, as the unitary operators are defined on the complex field, the translation to a weighted network is not trivial. Here we apply a generator decomposition of a unitary matrix U in the unitary group $\mathrm{U}(N)$, where N is the dimension of the unitary matrix to represent the unitary operator as a weighted network. The generators of $\mathrm{U}(N)$ are the Hermitian matrices forming the Lie algebra.

Now a unitary matrix $U \in \mathrm{U}(N)$ can be written in the form $U = e^{-iG}$ where G is a Hermitian matrix. This Hermitian matrix can be represented with real coefficients a_{lm}, b_{lm}, and c_k by the decomposition of G with respect to the generators λ as

$$G = \sum_{l<m} \left(a_{lm}\lambda_{lm}^x + b_{lm}\lambda_{lm}^y\right) + \sum_k c_k \lambda_k^z, \tag{2}$$

where those λ generators are the generalized Gell-Mann matrices [34–36]

$$\left(\lambda_{lm}^\nu\right)_{ij} = \delta_{li}\delta_{mj}\sigma_{12}^\nu + \delta_{lj}\delta_{mi}\sigma_{21}^\nu \quad (\nu = x, y), \tag{3}$$

$$\lambda_k^z = \sqrt{\frac{2}{k(k+1)}}\,\mathrm{diag}(\underbrace{1,\ldots,1}_{k\text{ times}}, -k, 0, \ldots, 0), \tag{4}$$

$$\lambda_N^z = I. \tag{5}$$

Here δ_{ij} represents the Kronecker delta, σ_{ij}^ν $(\nu = x, y)$ is the (i, j) component of the Pauli matrices and I the identity matrix. Next G as a weight matrix has three components x, y, and z with $\{a_{lm}\}$, $\{b_{lm}\}$, and $\{c_k\}$ being the x, y, and z contributions of the weight matrix, respectively.

The Hermitian matrix G obtained from the unitary matrix U is not necessarily unique. To uniquely determine G for a given unitary matrix, we employ the *principal logarithm* of a matrix [37] in our numerical analysis. If A is a complex-valued matrix of dimension N with no eigenvalues on the negative real line $\mathbb{R}^-$, then there is a unique natural logarithm X of a matrix A such that all of its eigenvalues lie in the strip $\{z : -\pi < \mathrm{Im}(z) < \pi\}$. Here X is called the *principal logarithm* of A and denoted by $X = \log(A)$. For the DTC model, we compute the Hermitian matrix for period n, $G(n) = i\log[\hat{U}(nT, 0)]$ noting that $G(n)$ is not simply equal to $n \times G(1)$.

To convert the weight matrix to its weight distribution, we first count how many coefficients are in a certain value window $(s, s + ds]$ for $s, ds \in \mathbb{R}$. This gives us a histogram $h(s)$ to show how likely the coefficients are to take a certain value $(s, s + ds]$. In numerical calculations, we take 100 segments for each coefficient set to determine the value of ds: let the support of the histogram denoted by a sequence $S = \{s_0, s_1, \ldots, s_{M-1}\}$, where $s_i - s_{i-1} = ds > 0$ for all $i = 0, 1, \ldots, M-1$ and M is the number of segments in the

range $(s_0, s_{M-1}]$. Then, once we define s_0, s_{M-1}, and M, we have $ds = (s_{M-1} - s_0)/M$. After obtaining the histogram, we have a density function $\rho(s) = h(s)/N_\nu ds$ where N_ν is the number of elements in the component $\nu = x, y$, or z, that is, $N_\nu = \sum_{i=0}^{M-2} h(s_i)$. We refer to this as the weight distribution of the ν components ($\nu = x, y, z$) [see Fig. 1(b)].

B. Characterization of the weight distributions

We will show weight distributions for the DTC model in different configurations and other models in Sec. III. To quantitatively characterize those weight distributions, we calculate two quantities for each weight distribution. The first is the empirical standard deviation σ_ν is given by the standard deviation of values of elements in a component ν, for example, in the case of $\nu = x$, $\sigma_x = \sqrt{\mathrm{var}_{lm}(a_{lm})}$.

Our second quantity is a ratio that represents how far the weight distribution reaches from its center compared with a Gaussian function approximating the weight distribution, which is analogous to the MP rank defined in Ref. [38]. The ratio $\mathcal{R}^\nu$ for the weight distribution of the ν component is given by

$$\mathcal{R}^\nu = \frac{s_+^{(\nu,\mathrm{G})}}{s^{(\nu)}}. \tag{6}$$

The denominator $s^{(\nu)}$ is defined as a quantity representing how far the weight distribution reaches from its center in the horizontal axis. Since the weight distribution $\rho(s)$ is not necessarily symmetric with respect to $s = 0$, $s^{(\nu)}$ is given by $s^{(\nu)} = (s^{(\nu)}_{\mathrm{max}} - s^{(\nu)}_{\mathrm{min}})/2$ with $s^{(\nu)}_{\mathrm{max\ (min)}}$ being the maximum (minimum) value of s in the ν component, that is, $\max\{s \in S | \rho(s) \neq 0\}$ ($\min\{s \in S | \rho(s) \neq 0\}$). Next, $s_+^{(\nu,\mathrm{G})}$ represents how far a Gaussian distribution function reaches from $s = 0$ obtained by a Gaussian fitting to the weight distribution. The Gaussian function reaches $s = \infty$ in general. Thus, we introduce a cutoff for the Gaussian function. In more detail, let $f(s; u, v)$ denote the Gaussian function given by the Gaussian fitting with fitting parameters u, v: $f(s; u, v) = u \exp(-vs^2)$. To introduce the cutoff, we consider cross points between $f(s; u, v)$ and a horizontal line at a value of d, that is, $d = f(s; u, v)$. Then, $s_+^{(\nu,\mathrm{G})}$ is given as $s_+^{(\nu,\mathrm{G})} = \sqrt{\ln(u/d)/v}$. In our numerical calculations, we set $d = (N_\nu ds)^{-1}$, which corresponds to the possible minimum nonzero height of the density function $\rho(s) = h(s)/N_\nu ds$. In Fig. 1(c), the definition of $\mathcal{R}^\nu$ is summarized.

C. Simulation setup for the QERC

We begin our considerations here by first directly evaluating the properties of the feature map generated by our DTC dynamics using the method outlined above. Setting a computational task is not essential to do the analysis, however, it is extremely useful when later we compare these properties to the performance of the QERC. It is convenient to set a computational task to evaluate both at the same time and on a similar footing. In this paper, we use the well-known MNIST dataset [39] where each image has 784 ($= 28 \times 28$) pixels. We employ PCA to reduce each image data to the $2L$ components, which can then be encoded in the initial state of the quantum reservoir of L qubits by single-qubit rotations. Finally, to optimize the parameters of the ONN, we employ

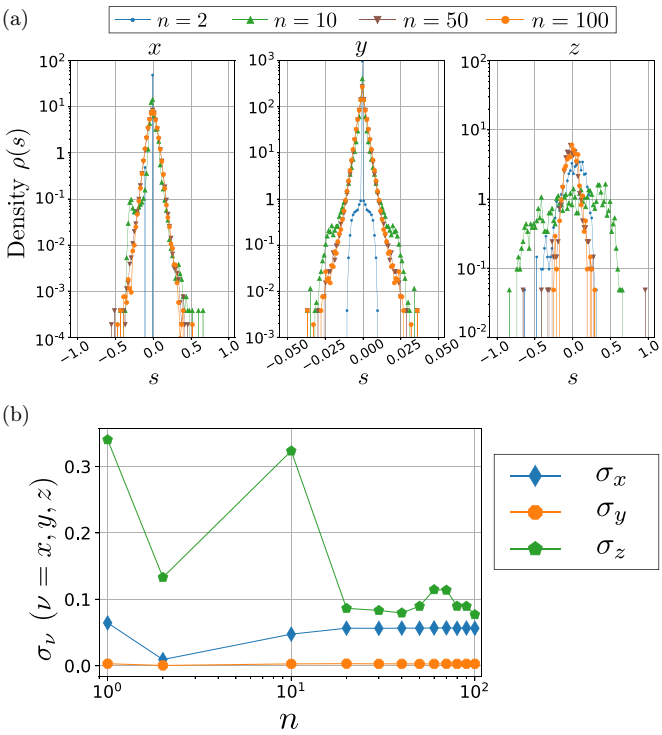

FIG. 2. (a) Convergence of the weight distribution in the DTC model. From the left to right panels, the weight distribution functions for a_{lm}, b_{lm}, and c_k are depicted where the colors correspond to different periods. The blue (dot), green (upward triangle), brown (downward triangle), and orange (filled circle) curves are for $n = 2, 10, 50$, and 100, respectively. (b) The n dependency of the empirical standard deviations, $\sigma_x = \sqrt{\mathrm{var}_{lm}(a_{lm})}$ (blue with diamonds), $\sigma_y = \sqrt{\mathrm{var}_{lm}(b_{lm})}$ (orange with hexagons), and $\sigma_z = \sqrt{\mathrm{var}_k(c_k)}$ (green with pentagons).

the stochastic gradient descent method used in Ref. [26]. Throughout this work, our parameters are set as $L = 10$, $J_0 T = 0.12$ with $\alpha = 1.51$, These are compatible with the current ion trap experiments [40]. We also set $\epsilon = 0.03$ as the highest accuracy rate that has been reported for the QERC with this parameter value [26].

III. QUANTUM RESERVOIR WEIGHT DISTRIBUTION

It is important to emphasize that, unless perfectly periodical, the quantum dynamics of the DTC model deviates from its initial state in time as it evolves. This allows for the growth of the complexity in the system. To observe such complexity growth in the unitary dynamics, we evaluate the weight distribution of $G(n)$ for various time periods: $n = 2, 10, 50$, and 100. From Eq. (2), we can determine the real coefficients a_{lm}, b_{lm}, and c_k characterizing $G(n)$ for each n. The Hermitian weight matrix $G(n)$ is equivalent to the n-period effective Hamiltonian up to the constant factor $\hbar/nT$. Thus, the diagonal (corresponding to $\{c_k\}$) and off-diagonal (corresponding to $\{a_{lm}\}$, $\{b_{lm}\}$) entries of $G(n)$ are associated with the energies of the basis states and the transition energies between the basis states, respectively.

In Fig. 2(a) we plot the weight distribution for the x (a_{lm}), y (b_{lm}), and z (c_k) components of $G(n)$, respectively. Each color

(symbol) represents a different period of the time evolution. For $n = 2$ (blue line), we observe very sharp peaks at $s = 0$ for the x and y components. As these components correspond to the off-diagonal entries of the Hermitian matrix $G(n)$, the sharp peaks around $s = 0$ mean very few transitions between the basis states for this time period. However, for large periods, $n = 50$ (brown curve), 100 (orange curve), the weight distributions for all the components converge to a similar shape that is approximately quadratic in the log-scaled plots (Gaussian in the linear plot). In the middle of these two time regions, at $n = 10$ (green curve) the x and y components have already converged to the typical distribution, however, the z component has broadened the most. This suggests that there is a tradeoff in this time regime; the stationary elements of the z component significantly suppress the effect of the x and y components. This trade-off captures the dynamics of the DTC melting slowly in time in this ($\epsilon = 0.03$) parameter regime.

The behavior for those components can be quantitatively observed by the empirical standard deviation for the weight distributions, defined in Sec. II B. In Fig. 2(b), the empirical standard deviations are depicted against the number of periods n. While the z component is broadened at $n = 10$, the x and y components gradually get higher standard deviations and then they converge.

A. Comparing the weight distribution

To capture the characteristics of the weight distribution for the DTC model, we first introduce the Haar measure sampling of unitary operators [25,41,42]. The Haar measure sampling can be considered to exhibit a typical complexity that a quantum computer may provide, and its gate implementation is usually given through unitary t design [22,23]. Hence the similarity and disparity in the weight functions for these cases would give us valuable insights into understanding the DTC dynamics and its role in QERC. In our analysis, to obtain a typical distribution given by the Haar measure sampling, the $N \times N$ unitary matrix U_{H} is created using the QR decomposition [43] where $N = 2^L = 2^{10} = 1024$. We compare this unitary map, which we refer to as the Haar-random model, to the converged weight distribution of the DTC model.

Now as shown in Figs. 3(a) and 3(b), the typical weight distributions are approximately Gaussian for all components x, y, and z. Here we use only one sample from the Haar-random model since one sample and not the average of many samples will be used within the QERC. Further, we do not lose generality as discussed in Appendix D.

We can now compare the DTC model to the Haar-random model, where we characterize the weight distributions of the DTC model with two properties, broadness, and tail. Although the DTC's y component has a narrower distribution compared to the Haar-random model, the broadness of the distribution for the x and z components are comparable as shown by the empirical standard deviations in Table I.

However, only the DTC model has a tail in the weight distribution of the x component; a few large elements at the edge of the weight distribution. To quantitatively observe tails in the weight distribution, we calculate the ratio $\mathcal{R}$ defined

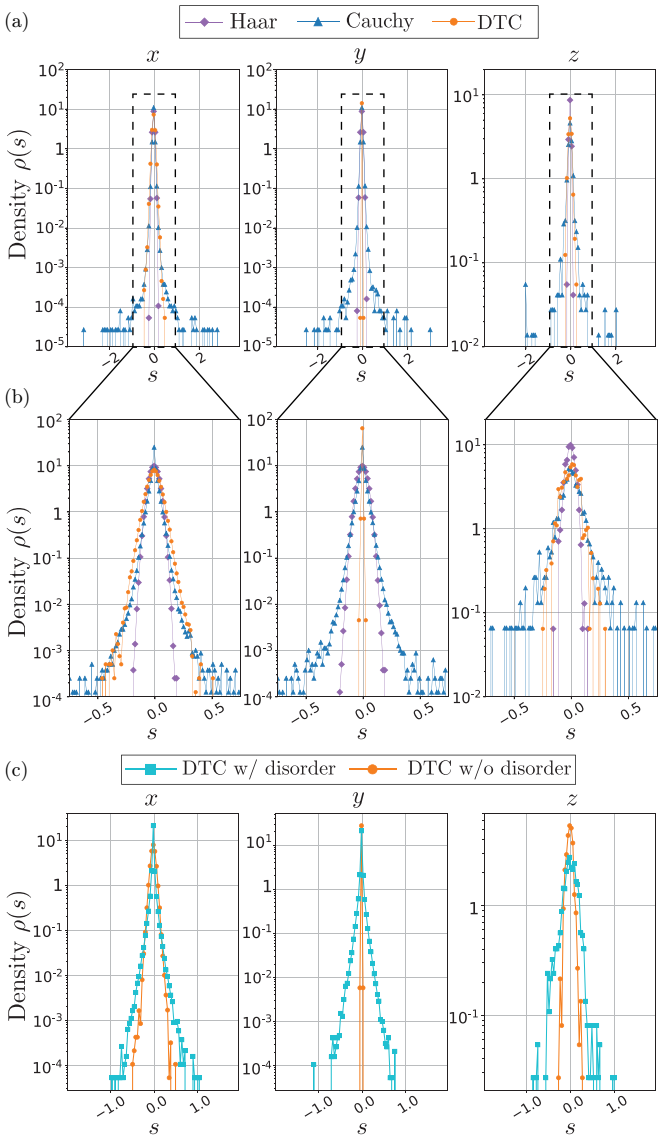

FIG. 3. Comparison of the Haar-random, Cauchy-random, and DTC models for $n = 100$ in (a). In each case from the left to right panels, the distributions of the x, y, and z components are depicted, respectively. In (b), the weight distributions of those models are shown for the range $[-0.75, 0.75]$. Finally, in (c), a comparison is shown between the DTC model with and without disorder for $n = 100$.

in Sec. II B. Table I shows the averages of the ratios for the DTC and Haar models where $\overline{\mathcal{R}}$ denotes the average, that is, $\sum_{\nu=x,y,z} \mathcal{R}^{\nu}/3$. One can see that the averaged ratio of the DTC model is smaller than that of the Haar model, which is close to the unit. It implies that the weight distribution of the DTC model deviates from that of the Haar model in terms of the tail.

Next, to explore the difference associated with the tail we found in the DTC models distribution, we employ the Cauchy distribution. The reason for this is as follows. In classical reservoir computation, the Cauchy distribution was used to obtain the edge of chaos where the reservoir computation should be optimal [44]. Hence it is interesting to see the properties of the feature map generated by the Cauchy distribution.

TABLE I. The empirical standard deviations and the ratios $\mathcal{R}$ for different models. $\overline{\mathcal{R}}$ denotes the average of the ratios over three components, that is, $\overline{\mathcal{R}} = \sum_{v=x,y,z} \mathcal{R}^v/3$.

	σ_x	σ_y	σ_z	$\overline{\mathcal{R}}$
Haar	0.0400	0.0402	0.0406	0.8592
Cauchy	0.0402	0.0394	0.2733	0.0978
DTC	0.0564	0.0028	0.0770	0.5189
DDTC	0.0410	0.0393	0.1914	0.2245

The Cauchy distribution is given by

$$\text{Cauchy}(x;\gamma) = \left(\frac{1}{\pi\gamma}\right)\frac{1}{1+(x/\gamma)^2}, \tag{7}$$

where γ is the scale parameter. Since the Cauchy distribution has a power-law tail, one would expect that the weight distribution exhibits a long tail. The unitary matrix U_C for this Cauchy-random model is defined as follows. First, we generate an $N \times N$ Hermitian matrix A whose real and imaginary parts in each independent entry are drawn from the Cauchy distribution (7). Then we define the unitary matrix U_C as $U_C = e^{-iA}$.

Further, we set $\gamma = 0.04$ in Eq. (7) for consistency with the corresponding parameter of the Haar-random model ($\sigma \approx 0.04$), and the size N is set as $N = 2^L = 1024$.

Applying the decomposition (2) to $G_C = i\log(U_C)$, we obtain each weight distribution for the three components x, y, and z, which are depicted in Figs. 3(a) and 3(b). The weight distributions for the Cauchy-random model definitely have a tail, much longer than that in the DTC model, for all the components x, y, and z. Table I shows the averaged ratio of the Cauchy-random model, which is even smaller than the other two models. This reflects the nature of the Cauchy distribution (7), and we will come back to this point later.

IV. RELATION BETWEEN THE QERC PERFORMANCE AND THE WEIGHT DISTRIBUTION

As we have characterized the three models through the weight distribution, let us now turn our attention to the performance of the QERC employing these three different models as its feature space. In the previous work [26] it was shown that the accuracy of the QERC increases with the number of the time periods of the DTC model saturating near $n = 50$. The behavior of this accuracy rate can be predicted from the time evolution of the weighted distributions seen in Fig. 2, as the unitary map of the DTC model acquires the typical complexity around $n = 50$. To illustrate this further, Fig. 4 summarizes the comparison between the accuracy rate and the weight distribution. Here we plot the accuracy rates for training (blue dot) and testing (orange downward triangle) against the time period n in the DTC model and insert the weight distributions for $n = 2, 10, 50, 100$.

The broadness in the x and y components of the weight distribution are essential for the quantum reservoir to achieve a higher performance. The trade-off between the x, y components and z component is reflected in the average accuracy

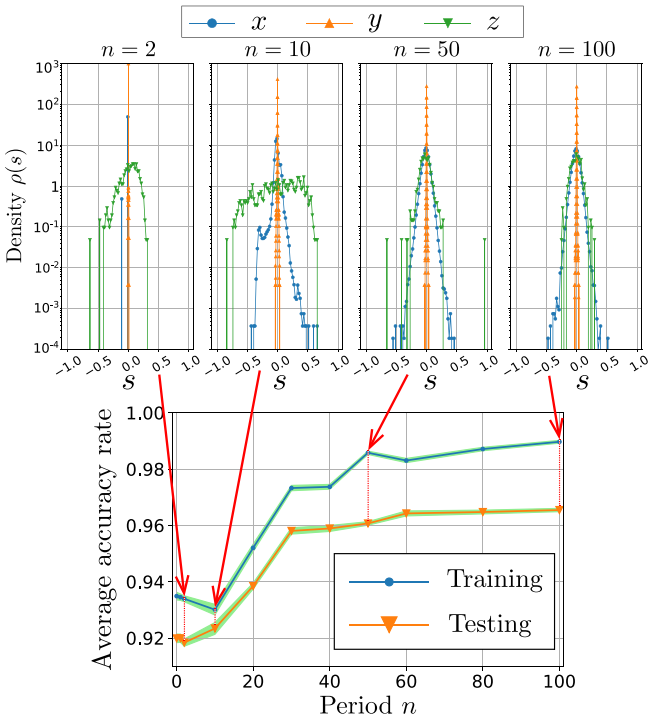

FIG. 4. Average accuracy rates for training and testing with the associated standard deviation against the period in the DTC model. The blue (dot) and orange (downward triangle) curves correspond to training and testing, respectively. At each datapoint, the average and the standard deviation are taken for 250 to 300 epochs in the ONN optimization.

rates. This suggests that even if the system dynamics is complex enough, within a short coherent regime, the system does not evolve enough to achieve the computational power the system would promise.

The complexity generated in a finite system has to be bounded, and unlike unitary maps from the Haar-measure sampling, the DTC model does not reach the maximum randomness allowed for the system to have certain tendencies in its dynamics. Next, we further investigate the effect of this difference in these models on the performance of the QERC.

A. Tails in the distribution and the performance

The unitary operators we characterized through the weight distributions directly serve as the feature map for the QERC. We will now explore how these different weight distributions and their associated feature maps affect the performance of the QERC with the MNIST dataset.

Table II(a) presents the accuracy rates for each model (see also Appendix A). Before the comparison of the quantum feature maps, we first provide the performance of the case where the PCA components are directly fed into the ONN (without quantum feature maps). One can immediately observe that the case, denoted by "PCA" in Table II(a), has the lower accuracy rates in both training and testing than any other cases with quantum feature maps. It states that those quantum feature maps significantly help the QERC achieving a high performance.

TABLE II. Average accuracy rates with the associated standard deviation of the various feature models with (a) the MNIST and (b) FashionMNIST datasets. For the FashionMNIST case, we picked three classes: t-shirt, pullover, and dress. The average and the standard deviation are taken from 250 to 300 epochs and $\Delta_{\text{acc.}}$ denotes the gap between training and testing. The label "PCA" denotes the case where the PCA components are directly fed into the ONN (without any quantum feature maps). The results for the DTC cases with or without disorder are for the period $n = 100$. In the random models, the accuracy rates are from a specific realization.

(a) MNIST	Testing acc. (std.)	Training acc. (std.)	$\Delta_{\text{acc.}}$
PCA	0.8635(±0.0009)	0.8688(±0.0005)	0.0053
Haar	0.9657(±0.0005)	0.9949(±0.0003)	0.0292
Cauchy	0.9673(±0.0005)	0.9945(±0.0003)	0.0272
DTC	0.9655(±0.0006)	0.9897(±0.0005)	0.0242
DDTC	0.9671(±0.0005)	0.9911(±0.0005)	0.0240
(b) FashionMNIST			
Haar	0.9427(±0.0017)	0.9744(±0.0017)	0.0317
Cauchy	0.9440(±0.0029)	0.9720(±0.0023)	0.0280
DTC	0.9388(±0.0039)	0.9662(±0.0049)	0.0274
DDTC	0.9407(±0.0050)	0.9589(±0.0044)	0.0182

Next, we compare the accuracy rates of the quantum feature maps. In testing, the DTC ($n = 100$) model is comparable to the Haar-random model, whereas not so in training. Moreover, it is interesting to notice that the Haar-random model does not give the best testing accuracy rate in this setting, but the Cauchy-random model does. These observations may suggest that the tail in the weight distribution of the DTC and Cauchy models contributes to the higher accuracy rate.

To see the relation between the accuracy rate and the weight distribution, we show the correlation between the testing accuracy rate and the quantities we used to characterize the weight distribution: the averaged empirical standard deviation ($\overline{\sigma}$) and ratio ($\overline{\mathcal{R}}$) in Fig. 5(a). The quantities correspond to the vertical and horizontal axes, respectively, and the color of the markers indicates the testing accuracy rate. One can find that the color becomes darker as the ratio $\overline{\mathcal{R}}$ gets smaller.

To investigate this further, we introduce the t-random model whose unitary is defined using the Student's t-distribution in the same way to generate the Cauchy-random model. The Student's t-distribution is defined as

$$g(x; 0, \gamma_t, \nu) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\gamma_t\sqrt{\pi\nu}\Gamma\left(\frac{\nu}{2}\right)}\left(1 + \frac{(x/\gamma_t)^2}{\nu}\right)^{-(\nu+1)/2}, \quad (8)$$

where $\Gamma(\cdot)$ denotes the Gamma function and γ_t is a scale parameter. ν is a parameter determining how heavy the tail of the distribution is. This parameter connects the standard Cauchy ($\nu = 1$) and normal ($\nu = \infty$) distributions when $\gamma_t = 1$. In our context, this parameter allows us to generate the weight distribution located in between the Haar- and Cauchy-random models in Fig. 5(a) with $\gamma_t = \sigma = \gamma = 0.04$.

In Fig. 5(a), the stars correspond to the averaged data points over ten realizations of the unitary for each value of ν ($\nu = 1, 2, 3, 5, 10$, and 100). The error bars correspond to the standard deviations of each data point (see Appendix B for detailed results of the t-random model). One can find that

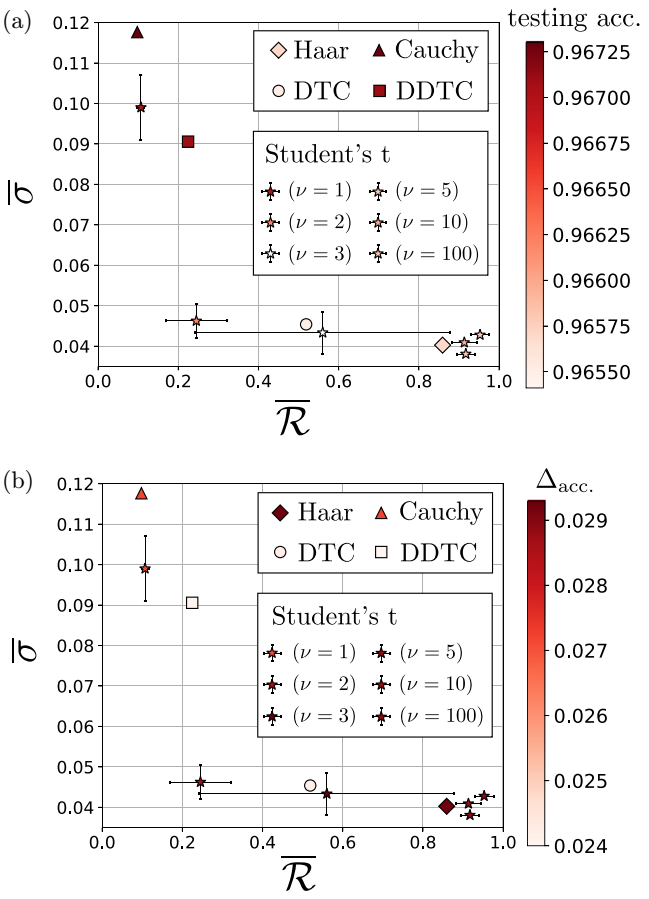

FIG. 5. Scatterplot of the models we consider in this paper against $\overline{\sigma}$ and $\overline{\mathcal{R}}$ with colored markers indicating (a) the testing accuracy rate and (b) the gap $\Delta_{\text{acc.}}$ of accuracy rates between training and testing.

the ratio and the testing accuracy rate tend to be smaller and darker, respectively, as ν gets smaller. Moreover, data points close to each other in the plot have similar testing accuracy rates. This simulation with the Student's t distribution illustrates the correlation between the testing accuracy rate and the tail in the weight distribution.

Considering the implementability of the QERC with an effective quantum feature map, one may wonder what physical system is close to the Cauchy- and t-random (for $\nu = 1$) models in Fig. 5(a) and achieves a high testing accuracy rate since the DTC model is far from such models, and there still seems to be room to improve it. We here consider disorder to the DTC model in Eq. (1), and actually, the disordered DTC (DDTC) model has a higher accuracy rate, as we will see later.

In Eq. (1), we set $D_l T = 0$ to enable us to consider the DTC model without the disorder. That constraint can now be relaxed. The disorder in Floquet systems has been considered to be important to suppress the thermalization and stabilize the DTCs [8,40]. Actually, it is more realistic to have a little disorder in such quantum systems, and so it is worth checking if our QERC's performance is robust to such disorder. We choose the disorder terms $D_l T$ in Eq. (1) independently drawn from a uniform distribution on $[0, 2\pi)$. As illustrated in Fig. 3(c), the introduction of the disorder changes the form of the weight

distribution, and it is characterized by the empirical standard deviation and the ratio (see Table I). So the DDTC model is now located around the upper left region of Fig. 5(a), and one can find that the model achieves a similar testing accuracy rate to that of the Cauchy- and t-random (with $\nu = 1$) models [see also Table II(a)].

So far, we only discussed our testing accuracy and it is important we now turn our attention to the training accuracy and the difference between the testing and training accuracy rates. The difference $\Delta_{\text{acc.}}$ is an important parameter in terms of the overfitting and the generalization performance of this machine learning model. Neural networks often show the effects of overfitting [45,46] where the neural networks are too well optimized to the training date and lose their flexibility to deal with the testing data. The generalization performance is hence an important factor in designing QNNs.

In Table II, we also provide the training accuracy and difference $\Delta_{\text{acc.}}$ for the various feature models we consider. We also plot the correlation between the generalization performance and the properties of the weight distribution in Fig. 5(b). One can observe that the Cauchy-random model has the smallest gap $\Delta_{\text{acc.}}$ among the artificial models, and the value of $\Delta_{\text{acc.}}$ tends to be higher for a larger value of the ratio. It is strongly suggestive that the tail in the weight distribution helps the QERC to acquire the generalization performance suppressing the training accuracy rate. One can also find that the physical models (DTC and DDTC) have even smaller gaps. Therefore, from our analysis, the DDTC model gives the best generalization performance and a nearly optimal testing accuracy rate among quantum feature maps we show. It is an encouraging fact that a simple Hamiltonian system could perform at least as good as a t-designed unitary map, which makes the implementation of such QNNs much simpler and more feasible.

B. Simulations in other settings

First, we consider the optimizer for the ONN. We used the stochastic gradient descent method as the optimizer for the ONN in these numerical simulations. The method has been broadly employed in many situations. Hence, our observations above can be seen in many scenarios. Moreover, we found the same effect of the tail with a more technical optimizer AdaGrad [47] (see Appendix C).

Next, we direct our attention to the dataset. So far, we used the MNIST dataset to benchmark quantum feature maps and concluded that the tail in the weight distribution contributes to the testing accuracy rate and generalization performance. In this section, we see the difference in the QERC performance between the quantum feature models we considered with another dataset. To see the difference in the generalization performance, we need to choose a dataset carefully. If it is a simple dataset such as the two-dimensional (2D) isotropic Gaussian samples demonstrated in Ref. [26], all the feature models would have high accuracy rates, and it should be hard to see the difference between our models. In contrast, if we choose a hard dataset such as the FashionMNIST [48], all the models we have may achieve poor performance and there should not be any room to discuss generalization.

For our calculations, we choose the FashionMNIST dataset with a few classes since classifying all the classes is too hard for the QERC. Then we picked three classes: t-shirt, pullover, and dress, and show the accuracy rate in Table II(b). One can see the same trend we obtain with the MNIST dataset: the models with the tail tend to have higher testing accuracy rates and better generalization. This examination suggests that the choice of a tailed feature map is an effective technique to push the QERC performance up more when the performance is good but not perfect.

V. DISCUSSION AND CONCLUSION

In this work, we performed network analysis on the unitary maps used in the QERC. Such unitary maps U can be converted to a weighted Hermitian matrix $G = i \log U$ that is characterized by the set of three weighted distribution functions. We observed that the weight distribution for the DTC model grows in time to near $n = 50$ where it converges to its typical shape. We compared the DTC model weight distribution against those associated with the Haar-random and Cauchy-random models. The DTC and Haar-random models are similar with respect to the Gaussian-like broadness of their weight distributions, whereas the DTC and Cauchy-random models are similar in terms of the tails in their respective weight distribution. This suggests that the unitary map for DTC's period over $n = 50$ is nearly as complex as the one

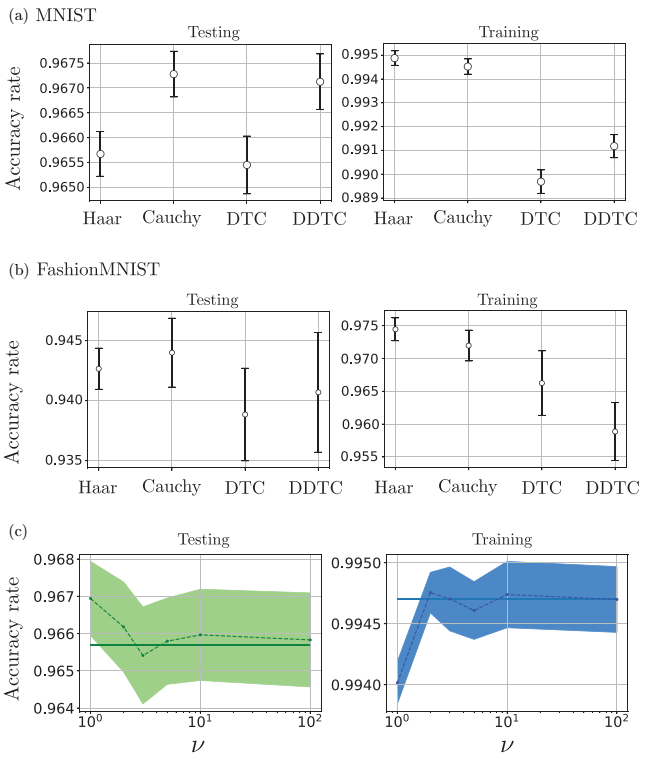

FIG. 6. Plot of the accuracy rates (with error bars) shown in Table II with (a) the MNIST and (b) FashionMNIST datasets. (c) Accuracy rates for testing and training of the t-random model against ν with the MNIST dataset. The dashed lines and shaded areas correspond to the accuracy rates averaged over ten realizations for each value of ν. The horizontal solid lines are the averaged accuracy rates of the Haar-random model.

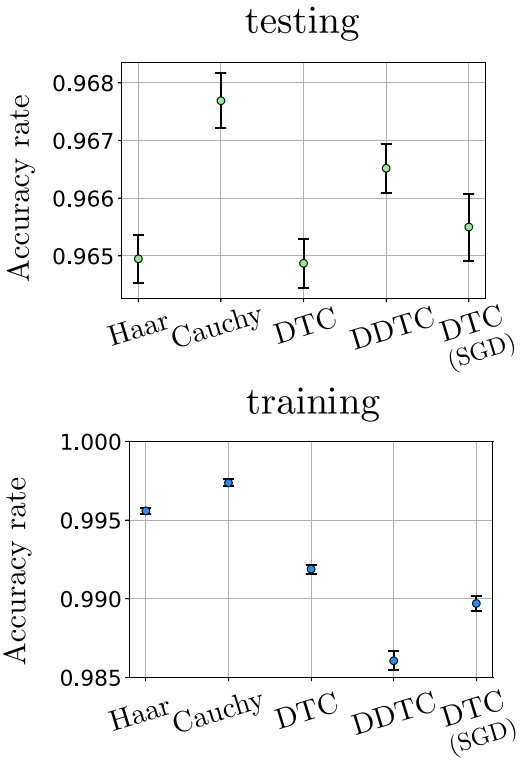

FIG. 7. Accuracy rates for testing and training of various models with AdaGrad. The same specific realizations are used for the random models as in the numerical simulations in the text. At each data point, the average is taken from 250 to 300 epochs. The error bar indicates the associated standard deviation.

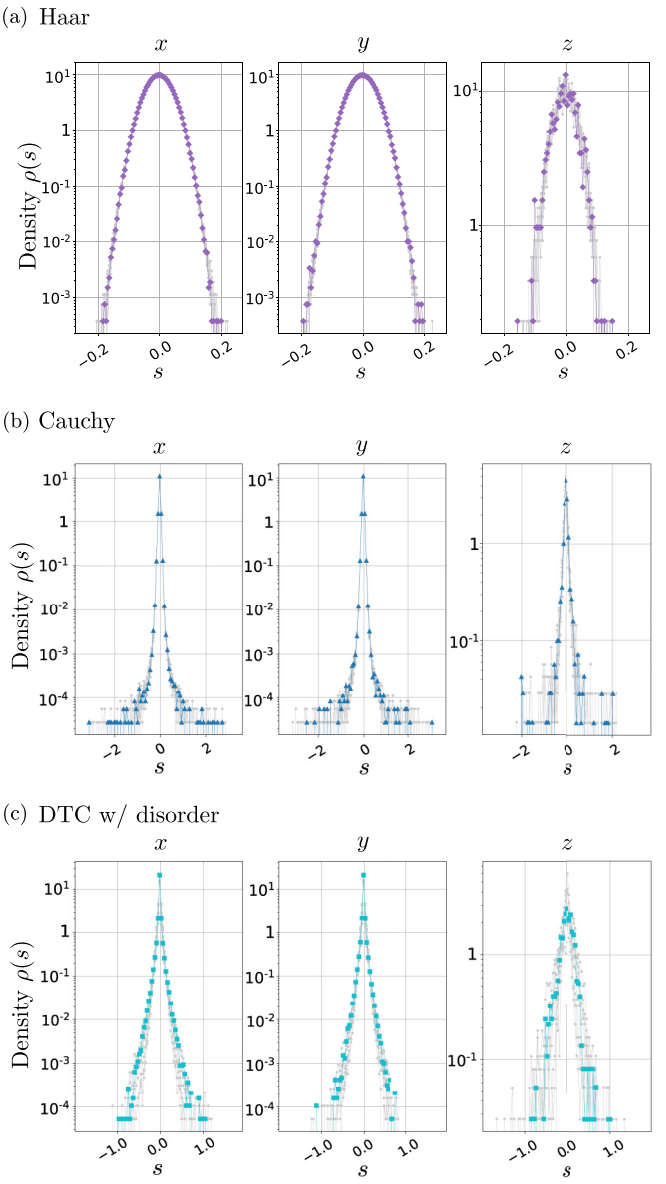

FIG. 8. Weight distributions of the (a) Haar random model, (b) the Cauchy random model, (c) and the disordered DTC model. The total number of realizations in each case is 10. The colored curves are corresponding to each curve in Figs. 3(a), 3(b), and 3(d), respectively. The other realizations are plotted as gray curves.

by the Haar-random model, yet it still has power-law characteristics and so is not totally random.

Next, in the comparison of the performance of the QERC with the MNIST and FashionMNIST datasets, we found that the power-law tendency (tail) in the unitary map contributes to the high testing accuracy rate by suppressing the training accuracy rate. This indicates that, at least for certain image classification problems, the tendency in the feature map in QNNs would help the QNNs to acquire better generalization performance. Although similar observations have been noted in classical neural network models, including reservoir computation [20,21,44,49], this is the first time for it to be observed in the QNN scenario.

Not only could the properties found here serve as a guideline for designing more effective feature maps in the future, but our network approach to the quantum machine learning model could also be used to investigate other useful network properties in quantum feature spaces. Furthermore, the approach could provide a physical interpretation of information processing in quantum machine learning schemes. As the long tail in the weight distribution implies strong connectivities between certain basis states in the network dynamics, our approach may provide physical insights in the performance of quantum machine learning.

Finally, the fact that the QERC can perform with the feature map generated by a simple Hamiltonian model, such as the DTC with the disorder, is encouraging for the QERC's implementation. It strongly suggests that it can significantly reduce the overhead for the feature map in many other QNNs.

On the other hand, another overhead seems to remain; the reconstruction of the probability amplitudes of all computational basis states may require an exponentially large number of samples of the same initial states. However, it is nontrivial whether such a precise reconstruction is needed for the ONN. In fact, we need to acquire enough information for the task performed for the ONN. Hence, it remains an open question how much it is possible to reduce the measurement overhead while maintaining the good performance with a simple Hamiltonian model used for the quantum feature map.

ACKNOWLEDGMENTS

We thank V. M. Bastidas for valuable discussions. This work is partly supported by the MEXT Quantum Leap Flagship Program (MEXT Q-LEAP) under Grant No. JPMXS0118069605, the JSPS Kakenhi Grant No. 21H04880, and JST the establishment of university fellowships towards the creation of science technology innovation under Grant No. JPMJFS2136.

APPENDIX A: PLOT WITH ERROR BARS ASSOCIATED WITH TABLE II

Figure 6 is a plot associated with Table II. It clearly shows the difference in the training and testing accuracy rates between different models. In Fig. 6(a), the improvements in testing by the tail are significantly observed. In Fig. 6(b), one may think that the effect of the tail is not so significant. However, it is noteworthy that the training accuracy rate is significantly low in the Cauchy and DDTC models compared to the Haar and DTC models, respectively, whereas the testing accuracy rate is slightly higher. Moreover, the error bars are associated with epochs, and we usually pick the neural network parameters at a single epoch with good performance. Hence, we can extract the improvement by the tail in such a practice situation in the FashionMNIST case.

APPENDIX B: PERFORMANCE OF THE T-RANDOM MODEL

Figure 6(c) shows the accuracy rates for training and testing against ν. The dashed lines and shaded areas correspond to the accuracy rates averaged over ten realizations for each value of ν and the associated standard deviations, respectively. The horizontal solid lines are the averaged accuracy rates of the Haar-random model. As ν increases, that is, as the effect of the tail decreases, the accuracy rates for both training and testing converge to the averaged one of the Haar-random model.

APPENDIX C: PERFORMANCE WITH THE ADAGRAD OPTIMIZER

In this section, we will see the QERC performance with another optimizer for the ONN instead of the stochastic gradient descent (SGD) we use in the text. Here, we adopt AdaGrad [47] as the optimizer. It uses an adaptive algorithm to update the parameters in the ONN based on the geometry of the data observed in the earlier iteration steps. Hence, it is considered that the training loss converges more quickly than with SGD. In Fig. 7, the accuracy rates for training and testing are shown for various models. The accuracy rates for the DTC model with SGD are also plotted for comparison. In fact, the training accuracy rate tends to be higher than that of the SGD case, and consequently, the generalization is poorer. In this case, it can still be observed that the tail affects the testing accuracy rate and the generalization performance. Furthermore, the DDTC model has a nearly optimal testing accuracy rate and the best generalization in the AdaGrad case as well as the SGD case.

TABLE III. Realization- and epoch-averaged accuracy rates of the various models. The epoch average involves from 250 to 300 epochs. The realization average is taken over ten realizations for each model with the means (and standard deviations in parentheses) shown. $\Delta_{\text{acc.}}$ denotes the difference between the means for training and testing.

MNIST	Testing (std.)	Training (std.)	$\Delta_{\text{acc.}}$
Haar	0.9657 ($\pm$0.0011)	0.9947 ($\pm$0.0003)	0.0290
Cauchy	0.9678 ($\pm$0.0013)	0.9943 ($\pm$0.0002)	0.0265
DTC w/disorder	0.9668 ($\pm$0.0005)	0.9910 ($\pm$0.0002)	0.0242

APPENDIX D: REALIZATION-AVERAGED ACCURACY RATES

In Fig. 8, we show that the weight distribution of each model is depicted with ten realizations. The colored curves here correspond to those in Figs. 3(a)–3(c). We clearly observe that the width and tail properties of the weight distribution are not strongly dependent on the particular realizations.

The realization-averaged accuracy rates for the Haar-random, Cauchy-random, and disordered DTC models are also shown in Table III. As the mean difference $\Delta_{\text{acc.}}$ shows, the models which have a tail in the weight distribution potentially avoid overlearning. The disordered DTC model achieves the highest generalization performance.

[1] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell *et al.*, Nature (London) **574**, 505 (2019).

[2] N. Friis, O. Marty, C. Maier, C. Hempel, M. Holzäpfel, P. Jurcevic, M. B. Plenio, M. Huber, C. Roos, R. Blatt, and B. Lanyon, Phys. Rev. X **8**, 021012 (2018).

[3] H.-S. Zhong, H. Wang, Y.-H. Deng, M.-C. Chen, L.-C. Peng, Y.-H. Luo, J. Qin, D. Wu, X. Ding, Y. Hu *et al.*, Science **370**, 1460 (2020).

[4] L. S. Madsen, F. Laudenbach, M. F. Askarani, F. Rortais, T. Vincent, J. F. F. Bulmer, F. M. Miatto, L. Neuhaus, L. G. Helt, M. J. Collins *et al.*, Nature (London) **606**, 75 (2022).

[5] M. Gong, S. Wang, C. Zha, M.-C. Chen, H.-L. Huang, Y. Wu, Q. Zhu, Y. Zhao, S. Li, S. Guo *et al.*, Science **372**, 948 (2021).

[6] IBM Quantum, IBM Research Blog (Nov. 16th, 2021).

[7] K. Wright, K. M. Beck, S. Debnath, J. M. Amini, Y. Nam, N. Grzesiak, J.-S. Chen, N. C. Pisenti, M. Chmielewski, C. Collins *et al.*, Nat. Commun. **10**, 5464 (2019).

[8] P. Frey and S. Rachel, Sci. Adv. **8**, eabm7652 (2022).

[9] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke *et al.*, Rev. Mod. Phys. **94**, 015004 (2022).

[10] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, Nature (London) **567**, 209 (2019).

[11] M. Schuld and N. Killoran, Phys. Rev. Lett. **122**, 040504 (2019).

[12] M. Noori, S. S. Vedaie, I. Singh, D. Crawford, J. S. Oberoi, B. C. Sanders, and E. Zahedinejad, Phys. Rev. Appl. **14**, 034034 (2020).
[13] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, Nat. Commun. **9**, 4812 (2018).
[14] L. Bittel and M. Kliesch, Phys. Rev. Lett. **127**, 120502 (2021).
[15] K. Fujii and K. Nakajima, Phys. Rev. Appl. **8**, 024030 (2017).
[16] S. Ghosh, A. Opala, M. Matuszewski, T. Paterek, and T. C. H. Liew, npj Quantum Inf. **5**, 35 (2019).
[17] R. Martínez-Peña, G. L. Giorgi, J. Nokkala, M. C. Soriano, and R. Zambrini, Phys. Rev. Lett. **127**, 100502 (2021).
[18] R. A. Bravo, K. Najafi, X. Gao, and S. F. Yelin, PRX Quantum **3**, 030325 (2022).
[19] P. Mujal, R. Martínez-Peña, J. Nokkala, J. Garcí-Beni, G. L. Giorgi, M. C. Soriano, and R. Zambrini, Adv. Quantum Technol. **4**, 2100027 (2021).
[20] N. Bertschinger, T. Natschläger, and R. Legenstein, *Advances in Neural Information Processing Systems,* Vol. 17 (MIT Press, Cambridge, MA, 2004).
[21] J. Boedecker, O. Obst, J. T. Lizier, N. M. Mayer, and M. Asada, Theory Biosci. **131**, 205 (2012).
[22] Y. S. Weinstein, W. G. Brown, and L. Viola, Phys. Rev. A **78**, 052332 (2008).
[23] A. W. Harrow and R. A. Low, Commun. Math. Phys. **291**, 257 (2009).
[24] C. Neill, P. Roushan, K. Kechedzhi, S. Boixo, S. V. Isakov, V. Smelyanskiy, A. Megrant, B. Chiaro, A. Dunsworth, K. Arya *et al.*, Science **360**, 195 (2018).
[25] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner, J. M. Martinis, and H. Neven, Nat. Phys. **14**, 595 (2018).
[26] A. Sakurai, M. P. Estarellas, W. J. Munro, and K. Nemoto, Phys. Rev. Appl. **17**, 064044 (2022).
[27] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, Neurocomputing **70**, 489 (2006).
[28] J. Martyn, G. Vidal, C. Roberts, and S. Leichenauer, arXiv:2007.06082.
[29] M. A. Valdez, D. Jaschke, D. L. Vargas, and L. D. Carr, Phys. Rev. Lett. **119**, 225301 (2017).
[30] N. C. Menicucci, S. T. Flammia, and P. van Loock, Phys. Rev. A **83**, 042335 (2011).
[31] M. P. Estarellas, T. Osada, V. M. Bastidas, B. Renoust, K. Sanaka, W. J. Munro, and K. Nemoto, Sci. Adv. **6**, eaay8892 (2020).
[32] M. V. Altaisky, arXiv:quant-ph/0107012.
[33] F. Tacchino, C. Macchiavello, D. Gerace, and D. Bajoni, npj Quantum Inf. **5**, 26 (2019).
[34] W. Greiner and B. Müller, Symmetries in quantum mechanics, in *Quantum Mechanics: Symmetries* (Springer, Berlin, 1994), pp. 1–55.
[35] K. Nemoto, J. Phys. A: Math. Gen. **33**, 3493 (2000).
[36] G. Kimura, Phys. Lett. A **314**, 339 (2003).
[37] N. J. Higham, *Functions of Matrices: Theory and Computation* (Society for Industrial and Applied Mathematics, Philadelphia, 2008), pp. xx–425.
[38] M. Mahoney and C. Martin, in *International Conference on Machine Learning* (PMLR, Cambridge, MA, 2019), pp. 4284–4293.
[39] L. LeCun, C. Cortes, and C. Burges, http://yann.lecun.com/exdb/mnist/ (1998).
[40] J. Zhang, P. W. Hess, A. Kyprianidis, P. Becker, A. Lee, J. Smith, G. Pagano, I.-D. Potirniche, A. C. Potter, A. Vishwanath, N. Y. Yao, and C. Monroe, Nature (London) **543**, 217 (2017).
[41] J. Emerson, E. Livine, and S. Lloyd, Phys. Rev. A **72**, 060302(R) (2005).
[42] S. Mullane, arXiv:2007.07872.
[43] F. Mezzadri, Notices of the American Mathematical Society **54**, 592 (2007).
[44] L. Kuśmierz, S. Ogawa, and T. Toyoizumi, Phys. Rev. Lett. **125**, 028101 (2020).
[45] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, J. Mach. Learn. Res. **15**, 1929 (2014).
[46] R. Caruana, S. Lawrence, and C. Giles, *Advances in Neural Information Processing Systems*, Vol. 13 (MIT Press, Cambridge, MA, 2000).
[47] J. Duchi, E. Hazan, and Y. Singer, J. Mach. Learn. Res. **12**, 2121 (2011).
[48] H. Xiao, K. Rasul, and R. Vollgraf, arXiv:1708.07747.
[49] X. Zhang, X. Lin, and R. A. R. Ashfaq, J. Comput. **13**, 805 (2018).