Continuous-variable quantum neural networks

Nathan Killoran,[1] Thomas R. Bromley,[1] Juan Miguel Arrazola,[1] Maria Schuld,[1] Nicolás Quesada,[1] and Seth Lloyd[2]
[1]*Xanadu, Toronto, Ontario, Canada M5G 2C8*
[2]*Massachusetts Institute of Technology, Department of Mechanical Engineering, 77 Massachusetts Avenue, Cambridge, Massachusetts 02139, USA*

(Received 10 August 2018; published 31 October 2019)

We introduce a general method for building neural networks on quantum computers. The quantum neural network is a variational quantum circuit built in the continuous-variable (CV) architecture, which encodes quantum information in continuous degrees of freedom such as the amplitudes of the electromagnetic field. This circuit contains a layered structure of continuously parameterized gates which is universal for CV quantum computation. Affine transformations and nonlinear activation functions, two key elements in neural networks, are enacted in the quantum network using Gaussian and non-Gaussian gates, respectively. The non-Gaussian gates provide both the nonlinearity and the universality of the model. Due to the structure of the CV model, the CV quantum neural network can encode highly nonlinear transformations while remaining completely unitary. We show how a classical network can be embedded into the quantum formalism and propose quantum versions of various specialized models such as convolutional, recurrent, and residual networks. Finally, we present numerous modeling experiments built with the STRAWBERRY FIELDS software library. These experiments, including a classifier for fraud detection, a network which generates TETRIS images, and a hybrid classical-quantum autoencoder, demonstrate the capability and adaptability of CV quantum neural networks.

DOI: 10.1103/PhysRevResearch.1.033063

I. INTRODUCTION

After many years of scientific development, quantum computers are now beginning to move out of the laboratory and into the mainstream. Over those years of research, many powerful algorithms and applications for quantum hardware have been established. In particular, the potential for quantum computers to enhance machine learning is truly exciting [1–3]. Sufficiently powerful quantum computers can in principle provide computational speedups for key machine learning algorithms and subroutines such as data fitting [4], principal component analysis [5], Bayesian inference [6,7], Monte Carlo methods [8], support vector machines [9,10], Boltzmann machines [11,12], and recommendation systems [13].

On the classical computing side, there has recently been a renaissance in machine learning techniques based on neural networks, forming the new field of deep learning [14–16]. This breakthrough is being fueled by a number of technical factors, including new software libraries [17–21] and powerful special-purpose computational hardware [22,23]. Rather than the conventional bit registers found in digital computing, the fundamental computational units in deep learning are continuous vectors and tensors which are transformed in high-dimensional spaces. At the moment, these continuous computations are still approximated using conventional digital computers. However, new specialized computational hardware is currently being engineered which is fundamentally analog in nature [24–31].

Quantum computation is a paradigm that furthermore includes nonclassical effects such as superposition, interference, and entanglement, giving it potential advantages over classical computing models. Together, these ingredients make quantum computers an intriguing platform for exploring new types of neural networks, in particular hybrid classical-quantum schemes [32–39]. Yet the familiar qubit-based quantum computer has the drawback that it is not wholly continuous, since the measurement outputs of qubit-based circuits are generally discrete. Rather, it can be thought of as a type of *digital* quantum hardware [40], only partially suited to continuous-valued problems [41,42].

The quantum computing architecture which is most naturally continuous is the *continuous-variable* (CV) model. Intuitively, the CV model leverages the wavelike properties of nature. Quantum information is encoded not in qubits, but in the quantum states of fields, such as the electromagnetic field, making it ideally suited to photonic hardware. The standard observables in the CV picture, e.g., position $\hat{x}$ or momentum $\hat{p}$, have continuous outcomes. Importantly, qubit computations can be embedded into the quantum field picture [43,44], so there is no loss in computational power by taking the CV approach. Recently, the first steps towards using the CV model for machine learning have begun to be explored, showing how several basic machine learning primitives can be built in the CV setting [45,46]. As well, a kernel-based classifier using a CV quantum circuit was trained in Ref. [10]. Beyond these early forays, the CV model remains largely unexplored territory as a setting for machine learning.

In this work, we show that the CV model gives a native architecture for building neural network models on quantum computers. We propose a variational quantum circuit which straightforwardly extends the notion of a fully connected layer structure from classical neural networks to the quantum realm. This quantum circuit contains a continuously parameterized set of operations which are universal for CV quantum computation. By stacking multiple building blocks of this type, we can create multilayer quantum networks which are increasingly expressive. Since the network is made from a universal set of gates, this architecture can also provide a quantum advantage: for certain problems, a classical neural network would require exponentially many resources to approximate the quantum network. Furthermore, we show how to embed classical neural networks into a CV quantum network by restricting to the special case where the gates and parameters of the network do not create any superposition or entanglement.

This paper is organized as follows. In Sec. II, we review the key concepts from deep learning and from quantum computing which set up the remainder of the paper. We then introduce our basic continuous-variable quantum neural network model in Sec. III and explore it in detail theoretically. In Sec. IV, we validate and showcase the CV quantum neural network architecture through several machine learning modeling experiments. We conclude with some final thoughts in Sec. V.

II. BACKGROUND

In this section, we give a high-level synopsis of both deep learning and the CV model. To make this work more accessible to practitioners from diverse backgrounds, we will defer the more technical points to later sections. Both deep learning and CV quantum computation are rich fields; further details can be found in various review papers and textbooks [14,16,40,47–49].

A. Neural networks and deep learning

The fundamental construct in deep learning is the *feedforward neural network* (also known as the *multilayer perceptron*) [16]. Over time, this key element has been augmented with additional structure—such as convolutional feature maps [50], recurrent connections [51], attention mechanisms [52], or external memory [53]—for more specialized or advanced use cases. Yet the basic recipe remains largely the same: a multilayer structure, where each layer consists of a linear transformation followed by a nonlinear "activation" function. Mathematically, for an input vector $\mathbf{x} \in \mathbb{R}^n$, a single layer $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}^m$ performs the transformation

$$\mathcal{L}(\mathbf{x}) = \varphi(W\mathbf{x} + \mathbf{b}), \tag{1}$$

where $W \in \mathbb{R}^{m\times n}$ is a matrix, $\mathbf{b} \in \mathbb{R}^m$ is a vector, and φ is the nonlinear function. The objects W and $\mathbf{b}$—called the *weight matrix* and the *bias vector*, respectively—are made up of free parameters θ_W and θ_b. Typically, the activation function φ contains no free parameters and acts elementwise on its inputs.

The "deep" in deep learning comes from stacking multiple layers of this type together, so that the output of one layer is used as an input for the next. In general, each layer $\mathcal{L}_i$ will have its own independent weight and bias parameters. Summarizing all model parameters by the parameter set $\boldsymbol{\theta}$, an N-layer neural network model is given by

$$\mathbf{y} = f_{\boldsymbol{\theta}}(\mathbf{x}) = \mathcal{L}_N \circ \cdots \circ \mathcal{L}_1(\mathbf{x}), \tag{2}$$

and maps an input $\mathbf{x}$ to a final output $\mathbf{y}$.

Building machine learning models with multilayer neural networks is well-motivated because of various universality theorems [54–56]. These theorems guarantee that, provided enough free parameters, feedforward neural networks can approximate any continuous function on a closed and bounded subset of $\mathbb{R}^n$ to an arbitrary degree of accuracy. While the original theorems showed that two layers were sufficient for universal function approximation, deeper networks can be more powerful and more efficient than shallower networks with the same number of parameters [57–59].

The universality theorems prove the power of the neural network model for approximating functions, but those theorems do not say anything about how to actually find this approximation. Typically, the function to be fitted is not explicitly known, but rather its input-output relation is to be inferred from data. How can we adjust the network parameters so that it fits the given data? For this task, the workhorse is the stochastic gradient descent algorithm [60], which fits a neural network model to data by estimating derivatives of the model's parameters—the weights and biases—and using gradient descent to minimize some relevant objective function. Combined with a sufficiently large dataset, neural networks trained via stochastic gradient descent have shown remarkable performance for a variety of tasks across many application areas [14,16].

B. Quantum neural networks

While many attempts have been made over the years to encode neural-network-like models[1] into quantum systems, none has so far claimed the term "quantum neural network" univocally for itself. A major goal has been to "quantize" the transformation in Eq. (2) in order to gain advantages from quantum information processing [33,64–68]. Especially in earlier proposals, the aim was to build a fully coherent model in which both training and inference is implemented on a quantum computer [69]. Recently, the term "quantum neural network" is increasingly being used more generally to refer to parametrized quantum and hybrid algorithms which can be optimized or trained by a classical coprocessor [35,37,39,70]. In those models, faithfulness to the neural network structure [Eq. (2)] is loosened, in particular to better fit the hardware constraints of near-term devices. The name "neural networks" highlights the fact that the circuits contain many trainable parameters, and—in some cases—the use of repeated (i.e., "layered") quantum circuit building blocks to form a larger computation [36,71].

[1]We focus here on feed-forward neural networks, although Hopfield nets [61,62] and Boltzmann machines [12,63] have also been studied under the lens of quantum mechanics.

While to date almost all[2] quantum neural network proposals have used a computational model based on qubits, we propose here a natural encoding of information into continuous-variable systems, in particular using the quantum properties of light. The photonic approach straightforwardly matches both notions of quantum neural networks discussed above, i.e., it is a faithful quantization of Eq. (2), and it efficiently uses its native hardware's gate set without any additional compilation or conversion. This scheme can be used to encode classical photonic neural networks and universal quantum computations without any changes to the hardware gate layout. To interpolate between the classical and quantum realms, we need simply to activate the phase degrees of freedom within the circuit's gates; this enables the network to create superpositions and entanglement that are not present in the classical variant.

There are several advantages to our approach. The hardware-efficient design reduces the spatial and runtime resources to a minimum. Constant factors are important for classical machine learning due to scalability issues, while low-space, low-precision, and low-power setups are in increasing demand [73–75]. Furthermore, since modern communication is largely optical, photonic neural networks are uniquely suited to process directly on quantum data, such as might be distributed over a future quantum internet. Moreover, quantum photonic circuits provide a simple mechanism for carrying out the nonlinear transformations that are crucial for neural networks, while using only linear computing elements. Though the gates of a photonic quantum computer carry out linear transformations in an underlying Hilbert space, those operations appear as nonlinear transformations when we use a continuous representations of quantum states, such as the wave function or phase space pictures. Enacting nonlinear functions has otherwise been a dominant challenge for qubit-based quantum neural networks [64].

Beyond these considerations, the equivalence of the algorithm with physical operations gives rise to natural quantum extensions of classical neural networks. This makes it an excellent candidate to investigate transitions from classical to quantum machine learning, getting us closer to unveiling the power of quantum computing for machine learning tasks. It also allows for a very different approach to model design: instead of trying to artificially "quantize" a mathematical operation in search for quantum advantages, we ask what models a specific quantum hardware gives naturally rise to. The final model of the CV quantum neural network is indeed very different to any discrete-variable quantum model, extending classical neural networks in significant ways, and may give rise to very different representational capabilities.

C. The CV model

The CV formalism has a long history, and can be physically realized using optical systems [76,77], in the microwave regime [78–80], and using ion traps [81–84]. In the CV model, information is carried in the quantum states of bosonic modes, often called *qumodes*, which form the "wires" of a quantum circuit. Continuous-variable quantum information can be encoded using two related pictures: the *wave-function representation* [85,86] and the *phase space formulation* of quantum mechanics [87–90]. In the former, we specify a single continuous variable, say x, and represent the state of the qumode through a complex-valued function of this variable called the wave function $\psi(x)$, which is a vector in the space of square-integrable functions $L^2(\mathbb{C})$. Concretely, we can interpret x as a position coordinate, and $|\psi(x)|^2$ as the probability density of a particle being located at x. From elementary quantum theory, we can also use a wave function based on a conjugate momentum variable, $\phi(p)$. Instead of position and momentum, x and p can equivalently be pictured as the real and imaginary parts of a quantum field, such as light.

In the phase space picture, we treat the conjugate variables x and p on equal footing, giving a connection to classical Hamiltonian mechanics. Thus the state of a single qumode is encoded with two real-valued variables $(x, p) \in \mathbb{R}^2$. For N qumodes, the phase space employs $2N$ real variables $(\mathbf{x}, \mathbf{p}) \in \mathbb{R}^{2N}$. Qumode states are represented as real-valued functions $F(\mathbf{x}, \mathbf{p})$ in phase space called *quasiprobability distributions*. "Quasi" refers to the fact that these functions share some, but not all, properties with classical probability distributions. Specifically, quasiprobability functions can be negative. While normalization forces qubit systems to have a unitary geometry, normalization gives a much looser constraint in the CV picture, namely that the function $F(\mathbf{x}, \mathbf{p})$ has unit integral over the phase space. Qumode states also have a representation as vectors or density matrices in the countably infinite Hilbert space spanned by the *Fock states* $\{|n\rangle\}_{n=0}^{\infty}$, which are the eigenstates of the photon number operator $\hat{n} = (\hat{x}^2 + \hat{p}^2 - 1)/2$, where $\hat{x}$ and $\hat{p}$ are the position and momentum operators, respectively.[3] These basis states represent the particlelike nature of qumode systems, with n denoting the number of particles. This is analogous to how square-integrable functions can be expanded using a countable basis set like sines or cosines.

The phase space and Hilbert space formulations give equivalent predictions. Thus CV quantum systems can be explored from both a wavelike and a particlelike perspective. We will mainly concentrate on the former.

1. Gaussian operations

There is a key distinction in the CV model between the quantum gates which are *Gaussian* and those which are not. In many ways, the Gaussian gates are the "easy" operations for a CV quantum computer. The simplest single-mode Gaussian gates are *rotation* $\hat{R}(\phi)$, *displacement* $\hat{D}(\alpha)$, and *squeezing* $\hat{S}(r)$. The basic two-mode Gaussian gate is the (phaseless) *beamsplitter* $\hat{BS}(\theta)$, which can be understood as a rotation

[2]To our knowledge, the only exception is Ref. [72] who propose a quantum optics setup based on a Kerr amplifier to implement a perceptron model into photonics.

[3]We use the convention $\hbar = 1$.

between two qumodes. More explicitly, these Gaussian gates produce the following transformations on phase space:

$$\hat{R}(\phi): \begin{bmatrix} x \\ p \end{bmatrix} \mapsto \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} x \\ p \end{bmatrix}, \tag{3}$$

$$\hat{D}(\alpha): \begin{bmatrix} x \\ p \end{bmatrix} \mapsto \begin{bmatrix} x + \sqrt{2}\mathrm{Re}(\alpha) \\ p + \sqrt{2}\mathrm{Im}(\alpha) \end{bmatrix}, \tag{4}$$

$$\hat{S}(r): \begin{bmatrix} x \\ p \end{bmatrix} \mapsto \begin{bmatrix} e^{-r} & 0 \\ 0 & e^{r} \end{bmatrix} \begin{bmatrix} x \\ p \end{bmatrix}, \tag{5}$$

$$\hat{BS}(\theta): \begin{bmatrix} x_1 \\ x_2 \\ p_1 \\ p_2 \end{bmatrix} \mapsto \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & \cos\theta & -\sin\theta \\ 0 & 0 & \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ p_1 \\ p_2 \end{bmatrix}. \tag{6}$$

The ranges for the parameter values are $\phi, \theta \in [0, 2\pi]$, $\alpha \in \mathbb{C} \cong \mathbb{R}^2$, and $r \in \mathbb{R}$.

Notice that most of these Gaussian operations have names suggestive of a linear character. Indeed, there is a natural correspondence between Gaussian operations and affine transformations on phase space. For a system of N modes, the most general Gaussian transformation has the effect

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{p} \end{bmatrix} \mapsto M \begin{bmatrix} \mathbf{x} \\ \mathbf{p} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\alpha}_r \\ \boldsymbol{\alpha}_i \end{bmatrix}, \tag{7}$$

where M is a real-valued *symplectic matrix* and $\boldsymbol{\alpha} \in \mathbb{C}^N \cong \mathbb{R}^{2N}$ is a complex vector with real/imaginary parts $\boldsymbol{\alpha}_r/\boldsymbol{\alpha}_i$. This native affine structure will be our key for building quantum neural networks.

A matrix M is symplectic if it satisfies the relation $M^T \Omega M = \Omega$, where

$$\Omega = \begin{bmatrix} 0 & \mathbb{1} \\ -\mathbb{1} & 0 \end{bmatrix} \tag{8}$$

is the $2N \times 2N$ *symplectic form*. A generic symplectic matrix M can be split into a type of singular-value decomposition—known as the *Euler* or *Bloch-Messiah* decomposition [48,49]—of the form

$$M = K_2 \begin{bmatrix} \Sigma & 0 \\ 0 & \Sigma^{-1} \end{bmatrix} K_1, \tag{9}$$

where $\Sigma = \mathrm{diag}(c_1, \ldots, c_N)$ with $c_i > 0$, and K_1 and K_2 are real-valued matrices which are symplectic and orthogonal. A matrix K with these two properties must have the form

$$K = \begin{bmatrix} C & D \\ -D & C \end{bmatrix}, \tag{10}$$

with

$$CD^T - DC^T = 0, \tag{11}$$

$$CC^T + DD^T = \mathbb{1}. \tag{12}$$

We will also need later the fact that if C is an arbitrary orthogonal matrix, then $C \oplus C$ is both orthogonal and symplectic. Importantly, the intersection of the symplectic and orthogonal groups on $2N$ dimensions is isomorphic to the unitary group on N dimensions. This isomorphism allows us to perform the transformations K_i via the unitary action of passive linear optical interferometers.

Every Gaussian transformation on N modes [Eq. (7)] can be decomposed into a CV circuit containing only the basic gates mentioned above. Looking back to Eqs. (3)–(6), we can recognize that interferometers made up of R and BS gates are sufficient to generate the orthogonal transformations K_1 and K_2, while S gates are sufficient to give the scaling transformation $\Sigma \oplus \Sigma^{-1}$. Finally, displacement gates complete the full affine transformation. Alternatively, we could have defined the Gaussian transformations as those quantum circuits which contain only the gates given above. The Gaussian transformations are so-named because they map the set of Gaussian distributions in phase space to itself.

2. *Universality in the CV model*

Similar to neural networks, quantum computing comes with its own inherent notions of "universality." To define universality in the CV model, we need to first introduce operator versions of the phase space variables, namely, $\hat{x}$ and $\hat{p}$. The $\hat{x}$ operator has a spectrum consisting of the entire real line:

$$\hat{x} = \int_{-\infty}^{\infty} x|x\rangle\langle x| dx, \tag{13}$$

where the vectors $|x\rangle$ are orthogonal, $\langle x|x'\rangle = \delta(x - x')$. This operator is not trace-class, and the vectors $|x\rangle$ are not normalizable. In the phase space representation, the eigenstates $|x'\rangle$ correspond to ellipses centered at $x = x'$ which are infinitely squeezed, i.e., infinitesimal along the x axis and correspondingly infinite in extent on the p axis. The conjugate operator $\hat{p}$ has a similar structure:

$$\hat{p} = \int_{-\infty}^{\infty} p|p\rangle\langle p| dp, \tag{14}$$

where $\langle p|p'\rangle = \delta(p - p')$ and $\langle p|x\rangle = e^{-ipx}/\sqrt{2\pi}$. Each qumode of a CV quantum computer is associated with a pair of operators $(\hat{x}_i, \hat{p}_i)$. For multiple modes, we combine the associated operators together into vectors $\hat{\mathbf{r}} := (\hat{\mathbf{x}}, \hat{\mathbf{p}})$.

These operators have the commutator $[\hat{r}_j, \hat{r}_k] = i\Omega_{jk}$, which leads to the famous uncertainty relation for simultaneous measurements of $\hat{x}$ and $\hat{p}$. Connecting to Eq. (3), we can associate $\hat{p}$ with a rotation of the operator $\hat{x}$; more concretely, $\hat{p}$ is the Fourier transform of $\hat{x}$. Indeed, we can transform between $\hat{x}$ and $\hat{p}$ with the special rotation gate $\hat{F} := \hat{R}(\frac{\pi}{2})$. Using a functional representation, the $\hat{x}$ operator has the effect of multiplication $\hat{x}|\psi\rangle = \hat{x}\int \psi(x)|x\rangle dx = \int [x\psi(x)]|x\rangle dx$. In this same representation, $\hat{p}$ is proportional to the derivative operator, $\hat{p}|\psi\rangle = \int [-i\frac{\partial}{\partial x}\psi(x)]|x\rangle dx$, as expected from the theory of Fourier transforms.

Universality of the CV model is defined as the ability to approximate arbitrary transformations of the form

$$\hat{U}_H = \exp(-it\hat{H}), \tag{15}$$

where the generator $\hat{H} = H(\hat{\mathbf{x}}, \hat{\mathbf{p}})$ is a polynomial function of $(\hat{\mathbf{x}}, \hat{\mathbf{p}})$ with arbitrary but fixed degree [91]. Crucially, such transformations are unitary in the Hilbert space picture, but can have complex nonlinear effects in the phase space picture,

a fact that we later make use of for designing quantum neural networks. A set of gates is universal if it can be used to build any $\hat{U}_H$ through a polynomial-depth quantum circuit. In fact, a universal gate set for CV quantum computing consists of the following ingredients: all the Gaussian transformations from Eqs. (3)–(6), combined with any single non-Gaussian transformation, which corresponds to a nonlinear function on the phase space variables $(\mathbf{x}, \mathbf{p})$. This is analogous to classical neural networks, where affine transformations combined with a single class of nonlinearity are sufficient to universally approximate functions. Commonly encountered non-Gaussian gates are the cubic phase gate $\hat{V}(\gamma) = \exp(i\frac{\gamma}{3}\hat{x}^3)$ and the Kerr gate $\hat{K}(\kappa) = \exp(i\kappa\hat{n}^2)$.

III. CONTINUOUS-VARIABLE QUANTUM NEURAL NETWORKS

In this section, we present a scheme for quantum neural networks using the CV framework. It is inspired from two sides. First, from the structure of classical neural networks, which are universal function approximators and have demonstrated impressive performance on many practical problems. In particular, we draw some inspiration from recent work on photonics-based classical neural networks [31]. Second, we leverage ideas from variational quantum circuits, which have recently become the predominant way of thinking about algorithms on near-term quantum devices [10,34,35,37,92–96]. The main idea is the following: the fully connected neural network architecture provides a powerful and intuitive ansatz for designing variational circuits in the CV model.

We will first introduce the most general form of the quantum neural network, which is the analog of a classical fully connected network. We then show how a classical neural network can be embedded into the quantum formalism as a special case—where no superposition or entanglement is created with respect to the position basis states $|x\rangle$—and discuss the universality and computational complexity of the fully quantum network. As modern deep learning has moved beyond the basic feedforward architecture, considering ever more specialized models, we will also discuss how to extend or specialize the quantum neural network to various other cases, specifically recurrent, convolutional, and residual networks. In Table I, we give a high-level matching between neural network concepts and their CV analogs.

TABLE I. Conceptual correspondences between classical neural networks and CV quantum computing. Some concepts from the quantum side have no classical analog.

Classical	CV quantum computing
feedforward neural network	CV variational circuit
weight matrix W	symplectic matrix M
bias vector $\boldsymbol{b}$	displacement vector $\boldsymbol{\alpha}$
affine transformations	Gaussian gates
nonlinear function	non-Gaussian gate
weight/bias parameters	gate parameters
variable x	operator $\hat{x}$
derivative $\frac{\partial}{\partial x}$	conjugate operator $\hat{p}$
no classical analog	superposition
no classical analog	entanglement

A. Fully connected quantum layers

A general CV quantum neural network is built up as a sequence of layers, with each layer containing every gate from the universal gate set. Specifically, a layer $\mathcal{L}$ consists of the successive gate sequence shown in Fig. 1:

$$\mathcal{L} := \hat{\Phi} \circ \hat{\mathcal{D}} \circ \hat{\mathcal{U}}_2 \circ \hat{\mathcal{S}} \circ \hat{\mathcal{U}}_1, \tag{16}$$

where $\hat{\mathcal{U}}_i = \hat{\mathcal{U}}_i(\boldsymbol{\theta}_i, \boldsymbol{\phi}_i)$ are general N-port linear optical interferometers containing beamsplitter and rotation gates, $\hat{\mathcal{D}} = \otimes_{i=1}^{N}\hat{D}(\alpha_i)$ and $\hat{\mathcal{S}} = \otimes_{i=1}^{N}\hat{S}(r_i)$ are collective displacement and squeezing operators (acting independently on each mode) and $\hat{\Phi} = \hat{\Phi}(\boldsymbol{\lambda})$ is some non-Gaussian gate, e.g., a cubic phase or Kerr gate. The collective gate variables $(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{r}, \boldsymbol{\alpha}, \boldsymbol{\lambda})$ form the free parameters of the network, where $\boldsymbol{\lambda}$ can be optionally kept fixed.

The sequence of Gaussian transformations $\hat{\mathcal{D}} \circ \hat{\mathcal{U}}_2 \circ \hat{\mathcal{S}} \circ \hat{\mathcal{U}}_1$ is sufficient to parametrize every possible unitary affine transformation on N qumodes. In the Heisenberg picture or in phase space, this corresponds to the transformation of Eq. (7). This sequence thus has the role of a "fully connected" matrix transformation. Interestingly, adding a nonlinearity uses the same component that adds universality: a non-Gaussian gate $\hat{\Phi}$. Using $\hat{\mathbf{r}} = (\hat{\mathbf{x}}, \hat{\mathbf{p}})$ for the vector of canonical operators, we can write the combined transformation in a form reminiscent of Eq. (1), namely,

$$\mathcal{L}(\hat{\mathbf{r}}) = \varphi(M\hat{\mathbf{r}} + \boldsymbol{\alpha}). \tag{17}$$

Thanks to the CV encoding, we get a nonlinear functional transformation while still keeping the quantum circuit unitary. A simple example of the nonlinear function φ is obtained by setting $\hat{\Phi}(\lambda)$ to be the single-mode cubic-phase gate $\hat{V}(\lambda) = \exp(i\frac{\lambda}{3}\hat{x}^3)$ which effects the following nonlinear transformation on the single mode quadratures

$$\hat{V}(\lambda) : \begin{bmatrix}\hat{x}\\ \hat{p}\end{bmatrix} \mapsto \begin{bmatrix}\hat{x}\\ \hat{p} + \lambda\hat{x}^2\end{bmatrix}. \tag{18}$$

Similar to the classical setup, we can stack multiple layers of this type end-to-end to form a deeper network (Fig. 2). The quantum state output from one layer is used as the input for the next. Different layers can be made to have

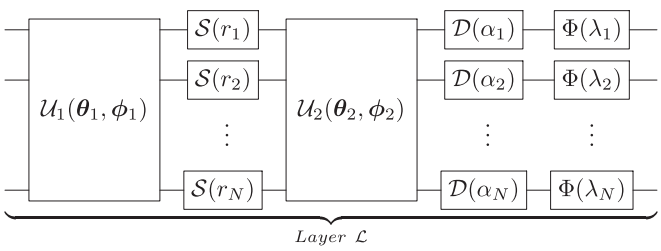

FIG. 1. The circuit structure for a single layer of a CV quantum neural network: an interferometer, local squeeze gates, a second interferometer, local displacements, and finally local non-Gaussian gates. The first four components carry out an affine transformation, followed by a final nonlinear transformation.

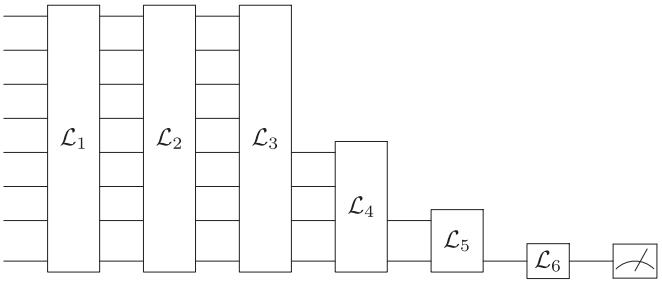

FIG. 2. An example multilayer continuous-variable quantum neural network. In this example, the later layers are progressively decreased in size. Qumodes can be removed either by explicitly measuring them or by tracing them out. The network input can be classical, e.g., by displacing each qumode according to data, or quantum. The network output is retrieved via measurements on the final qumode(s).

different widths by adding or removing qumodes between layers. Removal can be accomplished by measuring or tracing out the extra qumodes. In fact, conditioning on measurements of the removed qumodes is another method for performing non-Gaussian transformations [76]. This architecture can also accept classical inputs. We can do this by fixing some of the gate arguments to be set by classical data rather than free parameters, for example by applying a displacement $\hat{\mathcal{D}}(\mathbf{x})$ to the vacuum state to prepare the state $\hat{\mathcal{D}}(\mathbf{x})|\mathbf{0}\rangle$. This scheme can be thought of as an embedding of classical data into a quantum feature space [10]. The output of the network can be obtained by performing measurements and/or computing expectation values. The choice of measurement operators is flexible; different choices (homodyne, heterodyne, photon-counting, etc.) may be better suited for different situations. As is convention for variational circuit models, a final cost function is built using the expectation values from these measurements.

It is important to emphasize that, unlike many previous approaches using discrete variables, the CV formalism allows us to enact *nonlinear transformations* between functions, similar to classical neural networks, using purely *linear operations*, i.e., the unitary gates in a quantum circuit. This duality is due to the fact that the native domain of the CV model is the vector space $L^2(\mathbb{C})$. Vectors in this space correspond to square-integrable functions, and thus linear transformations on this space will take a function f to another function g, even if the functional relation between these vectors is nonlinear from the perspective of classical neural networks, i.e., $f(\mathbf{x}) = \varphi(W g(\mathbf{x}) + \mathbf{b})$.

In the following sections, we will explore different aspects of the CV quantum neural network architecture introduced in this section. Note that most of the discussion so far has been presented using either the Heisenberg picture or the phase-space representation. As is usual with quantum mechanics, though one particular representation may be more convenient to explain or understand certain features, no representation is fundamental. In the next sections, we sometimes examine our model using different representations, e.g., by looking at states in a given orthonormal basis, such as the Fock or quadrature basis. Using a different representation, of course, does not change the underlying model we are studying, just the way in which we look at it and understand its features.

B. Embedding classical neural networks

The above scheme for a CV quantum neural network is quite flexible and general. In fact, it includes classical neural networks as a special case, where we don't create any superposition or entanglement in the position basis. We now present a mathematical recipe for embedding a classical neural network into the quantum CV formalism. We note that this embedding is distinct from previous photonics-based proposals for classical neural networks [31], in particular with the use of the position eigenstates, squeezing, and quantum optical nonlinearities. We give the recipe for a single feed-forward layer; multilayer networks follow straightforwardly. Throughout this part, we will represent N-dimensional real-valued vectors $\mathbf{x}$ using N-mode quantum optical states built from the eigenstates $|x_i\rangle$ of the operators $\hat{x}_i$:

$$\mathbf{x} \leftrightarrow |\mathbf{x}\rangle := |x_1\rangle \otimes \cdots \otimes |x_N\rangle. \tag{19}$$

For the first layer in a network, we create the input $\mathbf{x}$ by applying the displacement operator $\hat{\mathcal{D}}(\mathbf{x})$ to the state $|\mathbf{x} = \mathbf{0}\rangle$. Subsequent layers will use the output of the previous layer as input. To read out the output from the final layer, we can use ideal homodyne detection in each qumode, which projects onto the states $|x_i\rangle$ [49].

We would like to enact a fully connected layer [Eq. (1)] completely within this encoding, i.e.,

$$|\mathbf{x}\rangle \mapsto |\varphi(W\mathbf{x} + \mathbf{b})\rangle. \tag{20}$$

This transformation will take place entirely within the $\mathbf{x}$ coordinates; we will not use the momentum variables. We thus want to restrict our quantum network to never mix between $\hat{\mathbf{x}}$ and $\hat{\mathbf{p}}$. To proceed, we will break the overall computation into separate pieces. Specifically, we split up the weight matrix using a singular value decomposition, $W = O_2 \Sigma O_1$, where the O_k are orthogonal matrices and Σ is a positive diagonal matrix. For simplicity, we assume that W is full rank. Rank-deficient matrices form a measure-zero subset in the space of weight matrices, which we can approximate arbitrarily closely with full-rank matrices.

Multiplication by an orthogonal matrix. The first step in Eq. (16) is to apply an interferometer $\hat{\mathcal{U}}_1$, which corresponds to the rightmost orthogonal matrix K_1 in Eq. (9). In order not to mix $\hat{\mathbf{x}}$ and $\hat{\mathbf{p}}$, we must restrict to block-diagonal K_1. With respect to Eqs. (10)–(12), this means that C is an orthogonal matrix and $D = 0$. This choice corresponds to an interferometer which only contains phaseless beamsplitters. With this restriction, we have

$$\begin{aligned}\hat{\mathcal{U}}_1|\mathbf{x}\rangle &= \hat{\mathcal{U}}_1\left[\bigotimes_{i=1}^{N}|x_i\rangle\right] \\ &= \bigotimes_{i=1}^{N}\left|\sum_{j=1}^{N} C_{ij}x_j\right\rangle \\ &= |C\mathbf{x}\rangle. \end{aligned} \tag{21}$$

The full derivation of this expression can be found in Appendix A. Thus the phaseless linear interferometer $\hat{\mathcal{U}}_1$ is

equivalent to multiplying the encoded data by an orthogonal matrix C. To connect to the weight matrix $W = O_1 \Sigma O_2$, we choose the interferometer which has $C = O_1$. A similar result holds for the other interferometer $\hat{\mathcal{U}}_2$.

Multiplication by a diagonal matrix. For our next element, consider the squeezing gate. The effect of squeezing on the $\hat{x}_i$ eigenstates is [97]

$$\hat{S}(r_i)|x_i\rangle = \sqrt{c_i}|c_i x_i\rangle, \tag{22}$$

where $c_i = e^{-r_i}$. An arbitrary positive scaling c_i can thus be achieved by taking $r_i = \ln(c_i)$. Note that squeezing leads to compression (positive r_i, $c_i \leqslant 1$), while antisqueezing gives expansion (negative r_i, $c_i \geqslant 1$), matching with Eq. (5). A collection of local squeezing transformations thus corresponds to an elementwise scaling of the encoded vector,

$$\hat{\mathcal{S}}(\mathbf{r})|\mathbf{x}\rangle = e^{-\frac{1}{2}\sum_i r_i}|\Sigma\mathbf{x}\rangle, \tag{23}$$

where $\Sigma := \mathrm{diag}(\{c_i\}) > 0$. We note that since the $|x_i\rangle$ eigenstates are not normalizable, the prefactor has limited formal consequence.

Addition of bias. Finally, it is well-known that the displacement operator acting locally on quadrature eigenstates has the effect

$$\hat{\mathcal{D}}(\alpha_i)|x_i\rangle = |x_i + \sqrt{2}\alpha_i\rangle, \tag{24}$$

for $\alpha_i \in \mathbb{R}$, which collectively gives

$$\hat{\mathcal{D}}(\boldsymbol{\alpha})|\mathbf{x}\rangle = |\mathbf{x} + \sqrt{2}\boldsymbol{\alpha}\rangle. \tag{25}$$

Thus, to achieve a bias translation of $\mathbf{d}$, we can simply displace by $\boldsymbol{\alpha} = \frac{1}{\sqrt{2}}\mathbf{d}$.

Affine transformation. Putting these ingredients together, we have

$$\begin{aligned}\hat{\mathcal{D}} \circ \hat{\mathcal{U}}_2 \circ \hat{\mathcal{S}} \circ \hat{\mathcal{U}}_1|\mathbf{x}\rangle &\propto |O_2 \Sigma O_1 \mathbf{x} + \boldsymbol{d}\rangle \\ &= |W\mathbf{x} + \boldsymbol{d}\rangle,\end{aligned} \tag{26}$$

where we have omitted the parameters for clarity. Hence, using only Gaussian operations which do not mix $\mathbf{x}$ and $\mathbf{p}$, we can effectively perform arbitrary full-rank affine transformations amongst the vectors $|\mathbf{x}\rangle$.

Nonlinear function. To complete the picture, we need to find a non-Gaussian transformation $\hat{\Phi}$ which has the following effect:

$$\hat{\Phi}|\mathbf{x}\rangle = |\varphi(\mathbf{x})\rangle, \tag{27}$$

where $\varphi : \mathbb{R} \to \mathbb{R}$ is some nonlinear function. We will restrict to an elementwise function, i.e., $\hat{\Phi}$ acts locally on each mode, similar to the activation function of a classical neural network. For simplicity, we will consider φ to be a polynomial of fixed degree. By allowing the degree of φ to be arbitrarily high, we can approximate any function which has convergent Taylor series. The most general form of a quantum channel consists of appending an ancilla system, performing a unitary transformation on the combined system, and tracing out the ancilla. For qumode i, we will append an ancilla i' in the $x = 0$ eigenstate, i.e.,

$$|x\rangle_i \mapsto |x\rangle_i|0\rangle_{i'}, \tag{28}$$

where, for clarity, we have made the temporary notational change $|x_i\rangle \leftrightarrow |x\rangle_i$.

Consider now the unitary $\hat{V}_\varphi := \exp\left(\frac{i}{\sqrt{2}}\varphi(\hat{x}_i) \otimes \hat{p}_{i'}\right)$, where $\varphi(\hat{x}_i)$ is understood as a Taylor series using powers of $\hat{x}_i$. Applying this to the above two-mode system, we get

$$\begin{aligned}\exp\left(-\frac{i}{\sqrt{2}}\varphi(\hat{x}_i) \otimes \hat{p}_{i'}\right)|x\rangle_i|0\rangle_{i'} &= \exp\left(-\frac{i}{\sqrt{2}}\varphi(x_i)\hat{p}_{i'}\right)|x\rangle_i|0\rangle_{i'} \\ &= \hat{\mathcal{D}}_{i'}(\varphi(x_i))|x\rangle_i|0\rangle_{i'} \\ &= |x\rangle_i|\varphi(x)\rangle_{i'},\end{aligned} \tag{29}$$

where we have recognized that $\hat{p}$ is the generator of displacements in x. We can now swap modes i and i' (using a perfectly reflective beamsplitter) and trace out the ancilla. The combined action of these operations leads to the overall transformation

$$|x_i\rangle \mapsto |\varphi(x_i)\rangle. \tag{30}$$

Alternatively, we are free to keep the system in the form $|x_i\rangle|\varphi(x_i)\rangle$; this can be useful for creating residual quantum neural networks.

Together, the above sequence of Gaussian operations, followed by a non-Gaussian operation, lead to the desired transformation $|\mathbf{x}\rangle \mapsto |\varphi(W\mathbf{x} + \mathbf{b})\rangle$, which is the same as a single-layer classical neural network. In this section, the states $|x\rangle$ were used in order to provide a convenient mathematical embedding; in a practical CV device, we would need to approximate the states $|x\rangle$ via finitely squeezed states. Finally, we remark that this particular classical neural network embedding strategy cannot be immediately extended to a coherent version [i.e., $\int d\mathbf{x}\psi(\mathbf{x})|\mathbf{x}\rangle \mapsto \int d\mathbf{x}\psi(\mathbf{x})|\varphi(W\mathbf{x} + \mathbf{b})\rangle$], since the nonlinearity we employed, which requires an ancilla, is not unitary. In fact, when implementing a CV neural network on real physical systems, it might be preferred to use more primitive (unitary) quantum nonlinearities, such as the cubic phase gate or the Kerr gate, rather than trying to directly mimic some particular classical activation function φ.

In summary, we have shown that the CV neural network model can duplicate all structure of a classical neural network when we use it in a particular way; specifically, a way that encodes all information into a single basis, does not leverage quantum phases, does not create superpositions, and does not generate entanglement. In practice, the general quantum neural network framework does not require any particular choice of basis or encoding. Because of this additional flexibility, the full quantum network has larger representational capacity than a conventional neural network, allows for quantum correlations like entanglement, and cannot be efficiently simulated by classical models, as we will discuss in Sec. III D.

C. Beyond the fully connected architecture

Modern deep learning techniques have expanded beyond the basic fully connected architecture. Powerful deep learning software packages [17–21] have allowed researchers to explore more specialized networks or complicated architectures. For the quantum case, we should also not feel restricted to the basic network structure presented above. Indeed, the CV model gives us flexibility to encode problems in a variety of representations. For example, we can use the phase space picture, the wave-function picture, the Hilbert space picture,

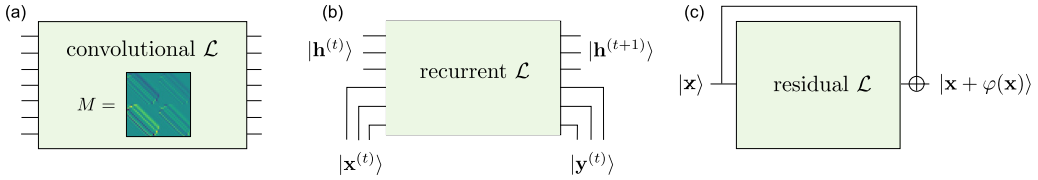

FIG. 3. Quantum adaptations of the convolutional layer, recurrent layer, and residual layer. The convolutional layer is enacted using a Gaussian unitary with translationally invariant Hamiltonian, resulting in a corresponding symplectic matrix that has a block Toeplitz structure. The recurrent layer combines an internal signal from previous layers with an external source, while the residual layer combines its input and output signals using a controlled-X gate.

or some hybrid of these. We can also encode information in coherent states, squeezed states, Fock states, or superpositions of these states. Furthermore, by choosing the gates and parameters to have particular structure, we can specialize our network ansatz to more closely match a particular class of problems. This can often lead to more efficient use of parameters and better overall models. In the rest of this section, we will highlight potential quantum versions of various special neural network architectures; see Fig. 3 for a visualization.

Convolutional network. A common architecture in classical neural networks is the convolutional network, or *convnet* [50]. Convnets are particularly well-suited for computer vision and image recognition problems because they reflect a simple yet powerful observation: since the task of detecting an object is largely independent of where the object appears in an image, the network should be equivariant to translations [16]. Consequently, the linear transformation W in a convnet is not fully connected; rather, it is a specialized sparse linear transformation, namely, a convolution. In particular, for one-dimensional convolutions, the matrix W has a *Toeplitz* structure, with entries repeated along each diagonal. This is similar to the well-known principle in physics that symmetries in a physical system can lead to simplifications of our physical model for that system (e.g., Bloch's theorem [98] or Noether's theorem [99]).

We can directly enforce translation symmetry on a quantum neural network model by making each layer in the quantum circuit translationally invariant. Concretely, consider the generator $\hat{H} = \hat{H}(\hat{\mathbf{x}}, \hat{\mathbf{p}})$ of a Gaussian unitary, $\hat{U} = \exp(-it\hat{H})$. Suppose that this generator is translationally invariant, i.e., $\hat{H}$ does not change if we map $(\hat{x}_i, \hat{p}_i)$ to $(\hat{x}_{i+1}, \hat{p}_{i+1})$. Then the symplectic matrix M that results from this Gaussian unitary will have the form

$$M = \begin{bmatrix} M_{\mathbf{xx}} & M_{\mathbf{xp}} \\ M_{\mathbf{px}} & M_{\mathbf{pp}} \end{bmatrix}, \tag{31}$$

where each $M_{\mathbf{uv}}$ is itself a Toeplitz matrix, i.e., a one-dimensional convolution (see Appendix B). The matrix M can be seen as a special kind of convolution that respects the uncertainty principle: performing a convolution on the $\mathbf{x}$ coordinates naturally leads to a conjugate convolution involving $\mathbf{p}$. The connection between translationally invariant Hamiltonians and convolutional networks was also noted in Ref. [59].

Recurrent network. This is a special-purpose neural network which is used widely for problems involving sequences [100], e.g., time series or natural language. A recurrent network can be pictured as a model which takes two inputs for every time step t. One of these inputs, $\mathbf{x}^{(t)}$, is external, coming from a data source or another model. The other input is an internal state $\mathbf{h}^{(t)}$, which comes from the same network, but at a previous time step (hence the name recurrent). These inputs are processed through a neural network $f_{\boldsymbol{\theta}}(\mathbf{x}^{(t)}, \mathbf{h}^{(t)})$, and an output $\mathbf{y}^{(t)}$ is (optionally) returned. Similar to a convolutional network, the recurrent architecture encodes translation symmetry into the weights of the model. However, instead of spatial translation symmetry, recurrent models have time translation symmetry. In terms of the network architecture, this means that the model reuses the same weights matrix W and bias vector b in every layer. In general, W or b are unrestricted, though more specialized architectures could also further restrict these.

This architecture generalizes straightforwardly to quantum neural networks, with the inputs, outputs, and internal states employing any of the data-encoding schemes discussed earlier. It is particularly well-suited to an optical implementation, since we can connect the output modes of a quantum circuit back to the input using optical fibres. This allows the same quantum optical circuit to be reused several times for the same model. We can reserve a subset of the modes for the data input and output channels, with the remainder used to carry forward the internal state of the network between time steps.

Residual network. The residual network [101], or *resnet*, is a more recent innovation than the convolutional and recurrent networks. While these other models are special cases of feedforward networks, the resnet uses a modified network topology. Specifically, "shortcut connections," which perform a simple identity transformation, are introduced between layers. Using these shortcuts, the output of a layer can be added to its input. If a layer by itself would perform the transformation $\mathcal{F}$, then the corresponding residual network performs the transformation

$$\mathbf{x} \mapsto \mathbf{x} + \mathcal{F}(\mathbf{x}). \tag{32}$$

To perform residual-type computation in a quantum neural network, we look back to Eq. (29), where a two-mode unitary was given which carries out the transformation

$$|x\rangle|0\rangle \mapsto |x\rangle|\varphi(x)\rangle, \tag{33}$$

where φ is some desired non-Gaussian function. To complete the residual computation, we need to sum these two values together. This can be accomplished using the controlled-X (or *SUM*) gate $\hat{C}_X$ [43], which can be carried out with purely Gaussian operations, namely, squeezing and beamsplitters [102]. Adding a $\hat{C}_X$ gate after the transformation in Eq. (33),

we obtain

$$|x\rangle|0\rangle \mapsto |x\rangle|x+\varphi(x)\rangle, \tag{34}$$

which is a residual transformation. This residual transformation can also be carried out on arbitrary wave functions $\psi(x)$ in superposition over position eigenstates, giving the general mapping

$$\int \psi(x)|x\rangle dx \mapsto \int \psi(x)|x\rangle|x+\varphi(x)\rangle dx. \tag{35}$$

D. The advantages of CV quantum neural networks

In this section, we identify concrete properties of CV quantum neural networks that differentiate them from their classical counterparts and discuss how these properties could potentially improve performance in machine learning. Specifically, these are the following. (1) CV quantum neural networks can create superpositions and entanglement by allowing us to mix both the x and p representations, which are Fourier transforms of each other. (2) CV quantum neural networks are universal for photonic quantum computing: they can in principle be configured to perform any CV quantum algorithm, including those with known hardness results. (3) CV quantum neural networks are able to perform nonlinear transformations on probability distributions by making use of quantum interference effects.

Superposition and entanglement. When embedding classical neural networks, restrictions were imposed on the elementary gates constituting the quantum neural network: displacement and squeezing parameters were assumed to be real, interferometers contained only phaseless beamsplitters, and non-Gaussian transformations mapped position eigenstates to position eigenstates. A distinguishing feature of quantum physics is the possibility to operate not only on some fixed basis states, e.g., the states $|\mathbf{x}\rangle$, but also on superpositions of those basis states $|\psi\rangle = \int \psi(\mathbf{x})|\mathbf{x}\rangle d\mathbf{x}$, where $\psi(\mathbf{x})$ is a multimode wave function. Additionally, it is possible to apply gates that generate superpositions and entanglement, as well as to perform measurements in different bases. By relaxing the restrictions used in the classical embedding, it is possible to unlock the full capabilities of CV quantum neural networks.

For example, by allowing phases in the interferometers, single-mode Fourier gates $\hat{F} = e^{i\pi(\hat{x}^2+\hat{p}^2)/4}$ can be applied. A multimode Fourier transform $\hat{F}^{\otimes N}$ can generate superpositions in the position basis starting from eigenstates, $\hat{F}^{\otimes N}|\mathbf{x}\rangle = \frac{1}{(\sqrt{2\pi})^N}\int e^{i\mathbf{x}\cdot\mathbf{x}'}|\mathbf{x}'\rangle d\mathbf{x}'$, and natively map input wave functions $\psi(\mathbf{x})$ to their Fourier transforms $\tilde{\psi}(\mathbf{x})$:

$$\begin{aligned}\hat{F}^{\otimes N}\int \psi(\mathbf{x})|\mathbf{x}\rangle d\mathbf{x} &= \frac{1}{(\sqrt{2\pi})^N}\iint \psi(\mathbf{x})e^{i\mathbf{x}\cdot\mathbf{x}'}|\mathbf{x}'\rangle d\mathbf{x}d\mathbf{x}' \\ &= \int \tilde{\psi}(\mathbf{x}')|\mathbf{x}'\rangle d\mathbf{x}'. \end{aligned}\tag{36}$$

A nonsuperposition state in the position basis (e.g., $|\mathbf{x}\rangle$) can become highly superposed after applying a Fourier transform $\hat{F}^{\otimes N}$. By leveraging superposition, CV quantum neural networks can work in the position or momentum representations (alternatively, time or frequency domains) with equal ease.

As a second example, by allowing phases in the interferometers, entanglement will always be generated by the CV neural network, as long as the inputs are not all coherent states. This is a corollary of the result in Ref. [103]: "Given a nonclassical pure-product-state input to an N-port linear-optical network, the output is almost always mode entangled; the only exception is a product of squeezed states, all with the same squeezing strength, input to a network that does not mix the squeezed and antisqueezed quadratures." Nonclassical in their notation means precisely coherent states; for quadratures to not be mixed it has to be the case that the beamsplitters in the interferometers are phaseless, which was the case considered before when embedding classical neural networks.

Universality and hardness. In general, CV quantum neural networks are capable of leveraging the full power of universal quantum computation. Indeed, the quantum gates in a single layer form a universal gate set, and therefore a CV quantum neural network with sufficient layers can carry out any algorithm implementable on a universal CV quantum computer. In this sense, quantum neural networks are not only appealing because of their potential applications to machine learning and artificial intelligence: they are also a new framework for quantum computing that can enable the discovery of new quantum algorithms [71,93,104,105].

We will now overview how specific models of photonic quantum computing can be realized using CV quantum neural networks. First, Gaussian boson sampling (GBS) [106] is a model of photonic quantum computing where a multimode Gaussian state is prepared and subsequently measured in the photon-number basis, a procedure which is believed to be hard to simulate classically. GBS includes conventional Boson Sampling as a special case [107,108]. Any GBS configuration can be encoded in a CV quantum neural network by (i) turning off the non-Gaussian gates and (ii) measuring the outputs using photon detectors.

Additionally, CV quantum neural networks can be used to reproduce continuous-variable instantaneous quantum polynomial (CV-IQP) circuits [109], which consist of (i) input momentum eigenstates, (ii) a unitary transformation that is diagonal in the position basis, and (iii) momentum homodyne measurements. To realize CV-IQP circuits with quantum neural networks, it suffices to use Gaussian and non-Gaussian gates that are diagonal in the position basis. Stacked together, these gates allow the implementation of arbitrary unitaries that are diagonal in the position basis. Together with Fourier transform gates to create the input states and perform the final measurements, this leads to the implementation of general CV-IQP circuits. Both GBS [106,110] and CV-IQP circuits [109,111] have been shown to be intractable to simulate classically.

These first points establish that CV quantum neural networks can behave differently than their classical counterparts. On the one hand, this motivates the use of CV quantum neural networks for quantum machine learning tasks, such as developing quantum algorithms or for learning with quantum data like in quantum chemistry [112]. On the other hand, the additional computational power of quantum computing suggests potential for improvements to machine learning with classical data. Our next point helps to make concrete the improvements that are possible.

Nonlinear transformations of probability distributions. Transforming between probability distributions is an

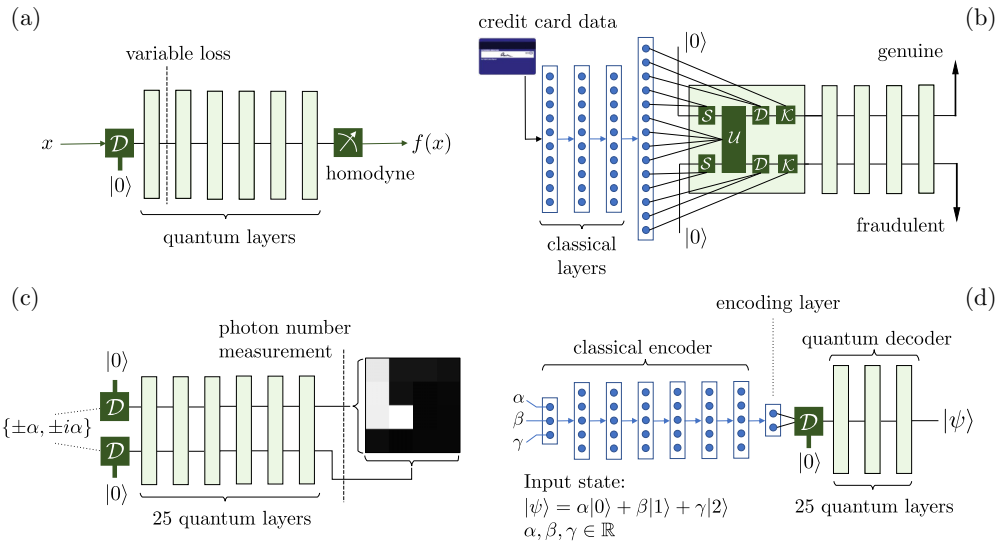

FIG. 4. Machine learning problems and architectures explored in this work: (a) curve fitting of functions $f(x)$ is achieved through a multilayer network, with x encoded through a position displacement on the vacuum and $f(x)$ through a position homodyne measurement at output; (b) credit card fraud detection using a hybrid classical-quantum classifier, with the classical network controlling the parameters of an input layer; (c) image generation of the TETRIS dataset from input displacements to the vacuum, with output image encoded in photon number measurements at the output mode; and (d) hybrid classical-quantum autoencoder for finding a continuous phase-space encoding for the first three Fock states.

important capability for generative machine learning, e.g., in the popular "normalizing flows" technique [113]. Functionality for manipulating distributions is already built into widely-used machine learning software such as TENSORFLOW [114]. The approach is to begin with a simple prior distribution $p(\mathbf{x})$, such as a Gaussian, which is easily sampled from. The sampled value $\mathbf{x}$ is then subject to a series of invertible transformations, parameterized via neural networks, leading to a final value $\mathbf{y} = f(\mathbf{x})$. This value is then a sample from the transformed probability distribution $\widetilde{p}(\mathbf{y}) = p(\mathbf{x})|\det \frac{\partial f}{\partial \mathbf{x}}|^{-1}$. This approach, common in generative neural networks, provides the ability to sample from a rich class of posterior distributions. However, this approach is still limited by the constraints of classical computing. For example, while the function f may be nonlinear with respect to the input $\mathbf{x}$, it is easily verified that the mapping is always a linear transformation with respect to the probability distribution $p(\mathbf{x})$ [115].

Given the results of Sec. III B, such transformations can also be performed using a CV quantum neural network. Yet the quantum model can carry out a richer class of transformations amongst probability distributions—including nonlinear transformations—since quantum mechanics works on the level of probability amplitudes. For example, for a given single mode state $|\psi\rangle = \int \psi(x)|x\rangle\, dx$, consider the probability distribution $p(x) = |\psi(x)|^2$ over measurements of the variable x. This probability distribution can be (invertibly) transformed by passing through one or more layers of a CV quantum neural network. If $\mathcal{W}$ is the unitary transformation corresponding to the network, the output state $|\widetilde{\psi}\rangle$ will have a transformed wave function

$$\widetilde{\psi}(x) = \int \mathcal{W}(x, x')\psi(x')dx' \tag{37}$$

and probability distribution $\widetilde{p}(x) = |\widetilde{\psi}(x)|^2$. Note that the output probability distribution is not related in a simple manner to the input probability distribution. Only the underlying probability amplitudes are connected via an integral transform with the complex kernel $\mathcal{W}(x, x')$.

As we have already discussed, the CV quantum neural network is a model of universal quantum computing, which allows for arbitrary transformations $\mathcal{W}$ between square-integrable wave functions, and by extension a richer class transformations between probability distributions $p(x)$ and $\widetilde{p}(x)$ than is possible via classical models. This capacity to carry out more complex manipulations of probability distributions by leveraging quantum interference effects suggests that CV quantum neural networks can enhance generative machine learning techniques.

IV. NUMERICAL EXPERIMENTS

We study several tasks in both supervised and unsupervised settings, with varying degrees of hybridization between quantum and classical neural networks. Some cases employ both classical and quantum networks whereas others are fully quantum. The architectures used are illustrated in Fig. 4. Unless otherwise stated, we employ the Adam optimizer [116] to train the networks and we choose the Kerr gate $\hat{K}(\kappa) = \exp(i\kappa\hat{n}^2)$ as the non-Gaussian gate in the quantum networks. Supplemental discussions, including a comparison of different optimization methods and regularization strategies, can be found in Appendix C. Our results highlight the wide range of potential applications of CV quantum neural networks, which will be further enhanced when deployed on dedicated hardware which exceeds the current limitations imposed by classical simulations.

A. Training quantum neural networks

We distinguish two regimes for the training of quantum neural networks. If they consist of a sufficiently small number of qumodes such that they can be simulated classically without prohibitive overheads, training can be in principle performed as with any classical model, namely, defining a cost function that depends on the output state of the circuit and optimizing the circuit parameters with respect to this cost function. This regime is particularly relevant in cases where the goal is to find configurations of small quantum circuits to perform specific tasks, for instance preparing complex quantum states [105].

Since quantum computations are hard to efficiently simulate, the simulator-based approach cannot be used for quantum neural networks larger than a few qumodes, and an alternative strategy is required for scalable hardware-based training. One approach is to treat the network as a black box, controlled by a set of gate parameters $(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{r}, \boldsymbol{\alpha}, \boldsymbol{\lambda})$, which is used to compute expectation values. Training could then be performed using numerical techniques, such as the finite-difference method or Nelder-Mead, to optimize the cost function under this black-box approach.

In this work, the networks are simulated using the STRAWBERRY FIELDS software platform [117] and the QUANTUM MACHINE LEARNING TOOLBOX app which is built on top of it. We use both automatic differentiation with respect to the quantum gate parameters—which is built into STRAWBERRY FIELDS' TENSORFLOW [20] quantum circuit simulator—as well as numerical optimization algorithms to train these networks. More concretely, STRAWBERRY FIELDS can express the output state of a quantum neural network as a TENSORFLOW Tensor object. TENSORFLOW's built-in tools then allow use of established optimization algorithms, based on stochastic gradient descent, for optimization of arbitrary cost functions which depend on the output state.

A more advanced training strategy could make use of circuit learning techniques [36] to implement new "quantum-aware" variants of the backpropagation algorithm. This method does not require the use of a simulator; instead, gradients of the cost function can be calculated using the same quantum neural network hardware that is being trained [118]. The main idea is to analytically differentiate the equation describing a circuit, e.g., Eq. (16), and rewrite the result as a difference of two circuits. This provides a "parameter shift rule" for how to obtain a gradient using the same hardware, but with different settings. This method can be combined seamlessly with conventional backpropagation algorithms, allowing hybrid computational models to be trained end-to-end [119]. In cases where an analytic gradient rule is not yet known, the finite difference method can be used as a fallback. This approach is efficient in the sense that the number of network parameters grows polynomially with the depth and width of the network.

In several of the examples we study in the following sections, cost functions are expressed in terms of a fidelity with respect to a target pure state. If the output state of the network is $\hat{\rho}$ and the target state is $|\varphi\rangle$, the fidelity can be expressed as the expectation value $\mathrm{Tr}(\hat{\rho}|\varphi\rangle\langle\varphi|)$ and the aforementioned training methods can be applied. For experimental implementation, it suffices to decompose the projector $|\varphi\rangle\langle\varphi|$ in terms of an experimentally-accessible basis of operators. These expectation values cannot be calculated exactly; rather, repeated measurements must be performed to estimate them with sufficient precision. Once gradients have been estimated in this manner, we can again employ established optimization algorithms based on stochastic gradient descent.

B. Curve fitting

A prototypical problem in machine learning is curve fitting: learning a given relationship between inputs and outputs. We will use this simple setting to analyze the behavior of CV quantum neural networks with respect to different choices for the model architecture, cost function, and optimization algorithm. We consider the simple case of training a quantum neural network to reproduce the action of a function $f(x)$ on one-dimensional inputs x, when given a training set of noisy data. This is summarized in Fig. 4(a). We encode the classical inputs as position-displaced vacuum states $\hat{\mathcal{D}}(x)|0\rangle$, where $\hat{\mathcal{D}}(x)$ is the displacement operator and $|0\rangle$ is the single-mode vacuum. Let $|\psi_x\rangle$ be the output state of the circuit given input $\hat{\mathcal{D}}(x)|0\rangle$. The goal is to train the network to produce output states whose expectation value for the quadrature operator $\hat{x}$ is equal to $f(x)$, i.e., to satisfy the relation $\langle\psi_x|\hat{x}|\psi_x\rangle = f(x)$ for all x.

To train the circuits, we use a supervised learning setting where the training and test data are tuples $(x_i, f(x_i))$ for values of x_i chosen uniformly at random in some interval. We define the loss function as the mean square error (MSE) between the circuit outputs and the desired function values

$$L = \frac{1}{N}\sum_{i=1}^{N}\left[f(x_i) - \langle\psi_{x_i}|\hat{y}|\psi_{x_i}\rangle\right]^2. \tag{38}$$

To test this approach in the presence of noise in the data, we consider functions of the form $\tilde{f}(x) = f(x) + \Delta f$ where Δf is drawn from a normal distribution with zero mean and standard deviation ϵ. The results of curve fitting on three noisy functions are illustrated in Fig. 5.

Avoiding overfitting. Ideally, the circuits will produce outputs that are smooth and do not overfit the noise in the data. CV quantum neural networks are inherently adept at achieving smoothness because quantum states that are close to each other cannot differ significantly in their expectation value with respect to observables. Quantitatively, Hölder's inequality states that for any two states $\hat{\rho}$ and $\hat{\sigma}$ it holds that

$$|\mathrm{Tr}[(\hat{\rho} - \hat{\sigma})\hat{X}]| \leqslant \|\hat{\rho} - \hat{\sigma}\|_1 \|\hat{X}\|_\infty \tag{39}$$

for any operator X. This smoothness property of quantum neural networks is clearly seen in Fig. 5, where the input/output relationship of quantum circuits gives rise to smooth functions that are largely immune to the presence of noise, while still being able to generalize from training to test data. We found that no regularization mechanism was needed to prevent overfitting of the problems explored here. Of course, the output of the function is an expectation, which can be calculated exactly when the circuits can be simulated, but only estimated to finite precision in an experimental setting. The error in

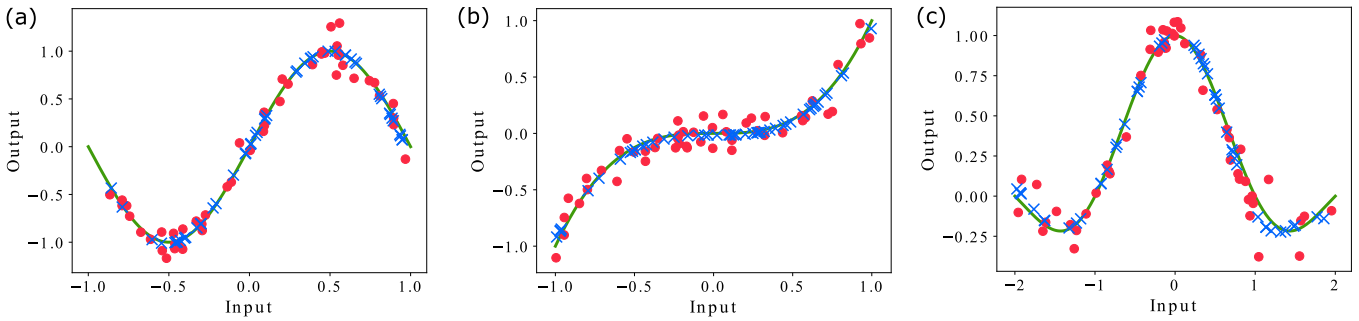

FIG. 5. Numerical experiment A. Curve fitting with continuous-variable quantum neural networks. The networks consist of a single mode and six layers, and was trained for 2000 steps with a Hilbert-space cutoff dimension of 10. As examples, we consider noisy versions of the functions $\sin(\pi x)$, x^3, and $\mathrm{sinc}(\pi x)$, displayed respectively from left to right. We set a standard deviation of $\epsilon = 0.1$ for the noise. The training data are shown as red circles. The output expectation values of the quantum neural network for the test inputs are shown as blue crosses. The outputs of the circuit very closely resemble the noiseless ground truth curves, shown in green. The scale is set such that $\hbar = 1$. In our simulated experiments, the expectation values are calculated numerically. In a real-world experiment, expectation values can be determined using empirical averages, and the precision of the estimate can be increased by repeated experiments.

the estimation can be lowered by increasing the number of measurements.

Improvement with depth. The circuit architecture is defined by the number of layers, i.e., the circuit depth. Figure 6 (top) studies the effect of the number of layers on the final value of the MSE. A clear improvement for the curve fitting task is seen for up to six layers, at which point the improvements saturate. The MSE approaches the square of the standard deviation of the noise, $\epsilon^2 = 0.01$, as expected when the circuit is in fact reproducing the input-output relationship of the noiseless curve.

Quantum device imperfections. We also study the effect of imperfections in the circuit, which for photonic quantum computers is dominated by photon loss. We model this using a lossy bosonic channel, with a loss parameter η. Here $\eta = 0\%$ stands for perfect transmission (no photon loss). The lossy channel acts at the end of each individual layer, ensuring that the effect of photon loss increases with circuit depth. For example, a circuit with six layers and loss coefficient $\eta = 10\%$ experiences a total loss of 46.9%. The effect of loss is illustrated in Fig. 6 (bottom) where we plot the MSE as a function of η. The quality of the fit exhibits resilience to this imperfection, indicating that the circuit learns to compensate for the effect of losses.

C. Supervised learning with hybrid networks

Classification of data is a canonical problem in machine learning. We construct a hybrid classical-quantum neural network as a classifier to detect fraudulent transactions in credit card purchases. In this hybrid approach, a classical neural network is used to control the gate parameters of the quantum network, the output of which determines whether the transactions are classified as genuine or fraudulent. This is illustrated in Fig. 4(b).

Data preparation. For the experiment, data were taken from a publicly available database of labeled historical credit card transactions which are flagged as either *fraudulent* or *genuine* [120]. The data are composed of 28 features derived through a principal component analysis of the raw data, providing an anonymization of the transactions. Of the 284,807 provided transactions, only 0.172% are fraudulent. We create training and test datasets by splitting the fraudulent transactions in two and combining each subset with genuine

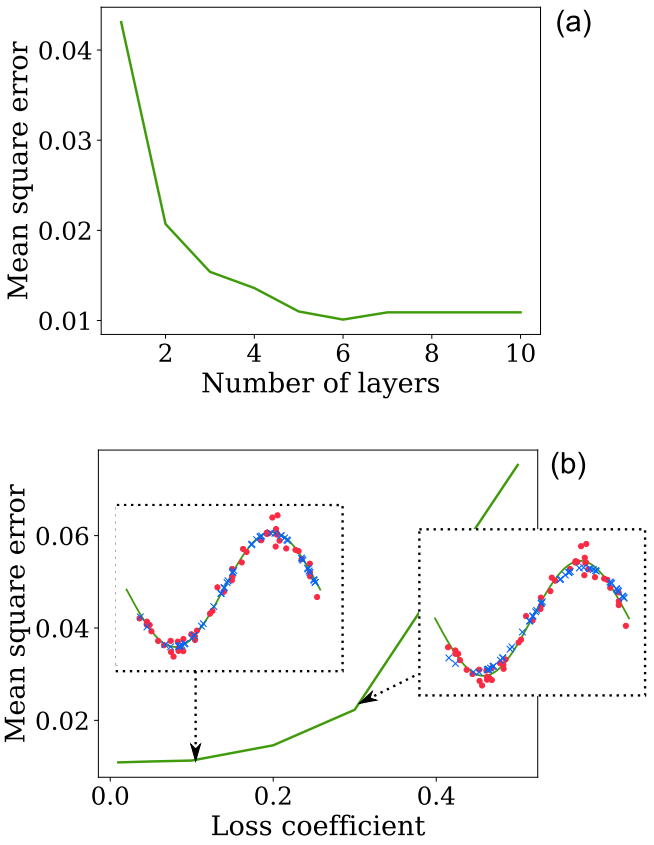

FIG. 6. MSE as a function of the number of layers and as a function of photon loss. The plots correspond to the task of fitting the function $\sin(\pi x)$ in the interval $x \in [-1, 1]$. (Top) Increasing the number of layers is helpful until a saturation point is reached with six layers, after which little improvement is observed. (Bottom) The networks can be resilient to imperfections, as seen by the fact that only a slight deviation in the mean square error appears for losses of 10% in each layer. The fits with a photon loss coefficient of 10% and 30% are shown in the inset.

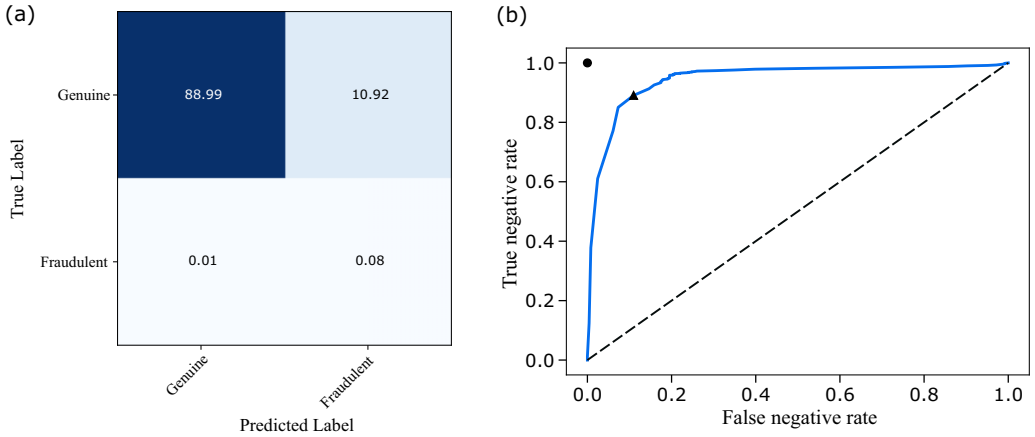

FIG. 7. Numerical experiment B. (Left) Confusion matrix for the test dataset with a threshold probability of $p_{\text{th}} = 0.9$. (Right) Receiver operating characteristic (ROC) curve for the test dataset, showing the true negative rate against the false negative rate as a parametric plot of the threshold probability. Here, the ideal point is given by the circle in the top-left corner, while the triangle denotes the closest point to optimal among chosen thresholds. This point corresponds to the confusion matrix given here, with threshold $p_{\text{th}} = 0.9$.

transactions. For the training dataset, we undersample the genuine transactions by randomly selecting them so that they outnumber the fraudulent transactions by a ratio of 3 : 1. This undersampling is used to address the notable asymmetry in the number of fraudulent and genuine transactions in the original dataset. The test dataset is then completed by adding all the remaining genuine transactions.

Hybrid network architecture. The first section of the network is composed of a series of classical fully connected feedforward layers. Here, an input layer accepts the first 10 features. This is followed by two hidden layers of the same size and the result is output on a layer of size 14. An exponential linear unit (ELU) was used as the nonlinearity. The second section of our architecture is a quantum neural network consisting of two modes initially in the vacuum. An input layer first operates on the two modes. The input layer omits the first interferometer as this has no effect on the vacuum qumodes. This results in the layer being described by 14 free parameters, which are set to be directly controlled by the output layer of the classical neural network. The input layer then feeds onto four hidden layers with fully controllable parameters, followed by an output layer in the form of a photon number measurement. An output encoding is fixed in the Fock basis by post-selecting on single-photon outputs and associating a photon in the first mode with a genuine transaction and a photon in the second mode with a fraudulent transaction.

Training. To train the hybrid network, we perform stochastic gradient descent (SGD) with a batch size of 24. Let p be the probability that a single photon is observed in the mode corresponding to the correct label for the input transaction. The cost function to minimize is

$$C = \sum_{i \in \text{data}} (1 - p_i)^2, \tag{40}$$

where p_i is the probability of the single photon being detected in the correct mode on input i. The probability included in the cost function is not post-selected on single photon outputs, meaning that training learns to output a useful classification as often as possible. We perform training with a cutoff dimension of 10 in each mode for 3×10^4 batches. Once trained, we use the probabilities post-selected on single photon events as classification, which could be estimated experimentally by averaging the number of single-photon events occurring across a sequence of runs.

Model performance. We test the model by choosing a threshold probability required for transactions to be classified as genuine. The confusion matrix for a threshold of $p_{\text{th}} = 0.9$ is given in Fig. 7. By varying the classification threshold, a receiver operating characteristic (ROC) curve can be constructed, where each point in the curve is parametrized by a value of the threshold. This is shown in Fig. 7, where the true negative rate is plotted against the false negative rate. An ideal classifier has a true negative rate of 1 and a false negative rate of 0, as illustrated by the circle in the figure. Conversely, randomly guessing at a given threshold probability results in the dashed line in the figure. Our classifier has an area under the ROC curve of 0.945, compared to the optimal value of 1.

For detection of fraudulent credit card transactions, it is imperative to minimize the false negative rate (bottom left square in the confusion matrix of Fig. 7), i.e., the rate of misclassifying a fraudulent transaction as genuine. Conversely, it is less important to minimize the false positive rate (top right square)—these are the cases of genuine transactions being classed as fraudulent. Such cases can typically be addressed by sending verification messages to cardholders. The larger false positive rate in Fig. 7 can also be attributed to the large asymmetry between the number of genuine and fraudulent data points.

The results here illustrate a proof-of-principle hybrid classical-quantum neural network able to perform classification for a problem of genuine practical interest. While it is simple to construct a classical neural network to outperform this hybrid model, our network is restricted in both width and depth due to the need to simulate the quantum network on a classical device. It would be interesting to further explore the performance of hybrid networks in conjunction with a physical quantum computer.

D. Generating images from labeled data

Next, we study the problem of training a quantum neural network to generate quantum states that encode grayscale images. We consider images of $N \times N$ pixels specified by a matrix A whose entries $a_{ij} \in [0, 1]$ indicate the intensity of the pixel on the ith row and jth column of the picture. These images can be encoded into two-mode quantum states $|A\rangle$ by associating each entry of the matrix with the coefficients of the state in the Fock basis:

$$|A\rangle = \frac{1}{\sqrt{\mathcal{N}}} \sum_{i,j=0}^{N-1} \sqrt{a_{ij}} |i\rangle|j\rangle, \tag{41}$$

where $\mathcal{N} = \sum_{i,j=0}^{N-1} |a_{ij}|^2$ is a normalization constant. We refer to these as *image states*. The matrix coefficients a_{ij} are the probability amplitude of observing i photons in the first mode and j photons in the second mode. Therefore, given many copies of a state $|A\rangle$, the image can be statistically reconstructed by averaging photon detection events at the output modes. This architecture is illustrated in Fig. 4(c).

Image encoding strategy. Given a collection of images $A_1, A_2, \ldots, A_n$, we fix a set of input two-mode coherent states $|\alpha_1\rangle|\beta_1\rangle, |\alpha_2\rangle|\beta_2\rangle, \ldots, |\alpha_n\rangle|\beta_n\rangle$. The goal is to train the quantum neural network to perform the transformation $|\alpha_i\rangle|\beta_i\rangle \to |A_i\rangle$ for all $i = 1, 2, \ldots, n$. Since the transformation is unitary, the Gram matrix of input and output states must be equal, i.e., it must hold that

$$\langle\alpha_i|\alpha_j\rangle\langle\beta_i|\beta_j\rangle = \langle A_i|A_j\rangle \tag{42}$$

for all i, j.

In general, it is not possible to find coherent states that satisfy this condition for arbitrary collections of output states. To address this, we consider output states with support in regions of larger photon number and demand that their projection onto the image Hilbert space of at most $N - 1$ photons in each mode coincides, modulo normalization, with the desired output states. Mathematically, if $\hat{\mathcal{V}}$ is the unitary transformation performed by the quantum neural network, the goal is to train the circuit to produce output states $\hat{\mathcal{V}}|\alpha_i\rangle|\beta_i\rangle$ such that

$$\hat{\Pi}_N \hat{\mathcal{V}}|\alpha_i\rangle|\beta_i\rangle = \sqrt{p_i}|A_i\rangle, \tag{43}$$

where $\hat{\Pi}_N = \sum_{i,j=0}^{N-1} |i\rangle\langle i| \otimes |j\rangle\langle j|$ is a projector onto the Hilbert space of at most $N - 1$ photons in each mode and $p_i = \mathrm{Tr}[\hat{\Pi}_N \hat{\mathcal{V}}|\alpha_i\rangle\langle\alpha_i| \otimes |\beta_i\rangle\langle\beta_i|\hat{\mathcal{V}}^\dagger]$ is the probability of observing the state in the subspace defined by this projector. The quantum neural network therefore needs to learn not only how to transform input coherent states into image states, it must also learn to employ the additional dimensions in Hilbert space to satisfy the constraints imposed by unitarity. This approach still allows us to retrieve the encoded image by performing photon counting, albeit with a penalty of p_i in the sampling rate.

As an example problem, we select a database of 4×4 images corresponding to the seven standard configurations of four blocks used in the digital game TETRIS. These configurations are known as tetrominos. For a fixed value of the parameter $\alpha > 0$, the seven input states are set to

$$\begin{aligned}
|\varphi_1\rangle &= |\alpha\rangle|\alpha\rangle, \\
|\varphi_2\rangle &= |-\alpha\rangle|-\alpha\rangle, \\
|\varphi_3\rangle &= |\alpha\rangle|-\alpha\rangle, \\
|\varphi_4\rangle &= |-\alpha\rangle|\alpha\rangle, \\
|\varphi_5\rangle &= |i\alpha\rangle|i\alpha\rangle, \\
|\varphi_6\rangle &= |-i\alpha\rangle|-i\alpha\rangle, \\
|\varphi_7\rangle &= |i\alpha\rangle|\alpha\rangle,
\end{aligned}$$

each of which must be mapped to the image state of a corresponding tetromino.

Training. We define the states

$$|\Psi_i\rangle := \hat{\mathcal{V}}|\varphi_i\rangle, \tag{44}$$

$$|\psi_i\rangle := \frac{\hat{\Pi}_4|\Psi_i\rangle}{\|\hat{\Pi}_4|\Psi_i\rangle\|}, \tag{45}$$

i.e., $|\Psi_i\rangle$ is the output state of the network and $|\psi_i\rangle$ is the normalized projection of the output state onto the image Hilbert space of at most three photons in each mode. To train the quantum neural network, we define the cost function

$$C = \sum_{i=1}^{7} |\langle\psi_i|A_i\rangle|^2 + \gamma P(\{|\Psi_i\rangle\}), \tag{46}$$

where $|A_1\rangle, |A_2\rangle, \ldots, |A_7\rangle$ are the image states of the seven tetrominos, P is the trace penalty as in Eq. (C1) and we set $\gamma = 100$. By choosing this cost function we are forcing each input to be mapped to a specific image of our choice. In this sense, we can view the images as labeled data of the form $(|\varphi_i\rangle, |A_i\rangle)$ where the label specifies which input state they correspond to. We employed a network with 25 layers [see Fig. 4(c)] and fixed a cutoff of 11 photons in the numerical simulation, setting the displacement parameter of the input states to $\alpha = 1.4$.

Model performance. The resulting image states are illustrated in Fig. 8, where we plot the absolute value squared of the coefficients in the Fock basis as grayscale pixels in an image. Tetrominos are referred to in terms of the letter of they alphabet they resemble. We fixed the desired output images according to the sequence "LOTISJZ" such that the first input state is mapped to the tetromino "L," the second to "O," and so forth.

Figure 8 clearly illustrates the role of the higher-dimensional components of the output states in satisfying the constraints imposed by unitarity: the network learns not only how to reproduce the images in the smaller Hilbert space but also how to populate the remaining regions in order to preserve the pairwise overlaps between states. For instance, the input states $|\varphi_1\rangle$ and $|\varphi_2\rangle$ are nearly orthogonal, but the images of the L and O tetrominos have a significant overlap. Consequently, the network learns to assign a relatively small probability of projecting onto the image space while populating the higher photon sectors in orthogonal subspaces. Overall, the network is successful in reproducing the images in the space of a few photons, precisely as it was intended to do.

E. Hybrid quantum-classical autoencoder

In this example, we build a joint quantum-classical autoencoder [see Fig. 4(d)]. Conventional autoencoders are neural

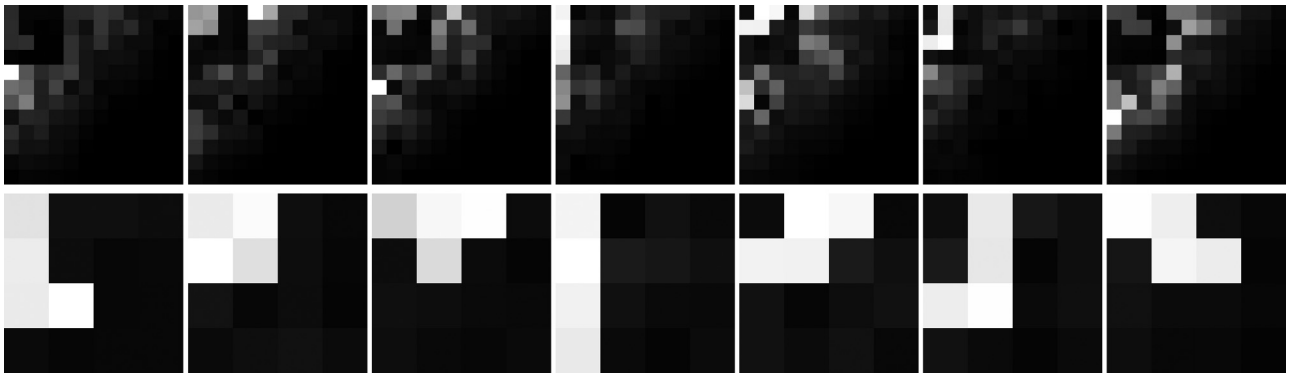

FIG. 8. Numerical experiment C. Output images for the "LOTISJZ" tetromino image data. The top row shows the output two-mode states where the intensity of the pixel in the ith row and jth column is proportional to the probability of finding i photons in the first mode and j photons in the second mode. The bottom row is a close-up in the image Hilbert space of up to three photons, renormalized with respect to the probability of projecting the state onto that subspace. In other words, this row illustrates the states $|\psi_i\rangle$ of Eq. (45). The fidelities of the output states $|\psi_i\rangle$ with respect to the desired image states are respectively 99.0%, 98.6%, 98.6%, 98.1%, 98.0%, 97.8%, and 98.8% for an average fidelity of 98.4%. The probabilities p_i of projecting the state onto the image space of at most three photons are respectively 5.8%, 36.0%, 21.7%, 62.1%, 40.7%, 71.3%, and 5.6%.

networks consisting of an encoder network followed by a decoder network. The objective is to train the network to act as an identity operation on input data. During training, the network learns a restricted encoding of the input data—which can be found by inspecting the small middle layer which links the encoder and decoder. For the hybrid autoencoder, our goal is to find a continuous phase-space encoding of the first three Fock states $|0\rangle$, $|1\rangle$, and $|2\rangle$. Each of these states will be encoded into the form of displaced vacuum states, then decoded back to the correct Fock state form.

Model architecture. For the hybrid autoencoder, we fix a classical feedforward architecture as an encoder and a sequence of layers on one qumode as a decoder, as shown in Fig. 4(d). The classical encoder begins with an input layer with three dimensions, allowing for any real linear combination in the $\{|0\rangle, |1\rangle, |2\rangle\}$ subspace to be input into the network. The input layer is followed by six hidden layers of dimension five and a two-dimensional output layer. We use a fully connected model with an ELU nonlinearity.

The two output units of the classical network are used to set the x and p components of a displacement gate acting on the vacuum in one qumode. This serves as a continuous encoding of the Fock states as displaced vacuum states. In fact, displaced vacuum states have Gaussian distributions in phase space, so the network has a resemblance to a variational autoencoder [121]. We employ a total of 25 layers with controllable parameters. The goal of the composite autoencoder is to physically generate the Fock state originally input into the network. Once the autoencoder has been trained, by removing the classical encoder we are left with a method to generate Fock states by varying the displacement of the vacuum. Notably, there is no need to specify which displacement should be mapped to each Fock state: this is automatically taken care of by the autoencoder.

Training. Our hybrid network is trained in the following way. For each of the Fock states $|0\rangle$, $|1\rangle$, and $|2\rangle$, we input the corresponding one-hot vectors $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$ into the classical encoder. Suppose that for an input $|i\rangle$ the encoder outputs the vector (x_i, p_i). This is used to displace the vacuum in one mode, i.e., enacting $\hat{\mathcal{D}}(\alpha_i)|0\rangle$ with $\alpha_i = (x_i, y_i)$. The output of the quantum decoder is the quantum state $|\Psi_i\rangle = \hat{\mathcal{V}}\hat{\mathcal{D}}(\alpha_i)|0\rangle$, with $\hat{\mathcal{V}}$ the unitary resulting from the layers. We define the normalized projection

$$|\psi_i\rangle = \frac{\hat{\Pi}_3|\Psi_i\rangle}{\|\hat{\Pi}_3|\Psi_i\rangle\|} \tag{47}$$

onto the subspace of the first three Fock states, with $\hat{\Pi}_3$ being the corresponding projector. As we have discussed previously, this allows the network to output the state $|\psi_i\rangle$ probabilistically upon a successful projection onto the subspace. The objective is to train the network so that $|\psi_i\rangle$ is close to $|i\rangle$, where closeness is measured using the fidelity $|\langle i|\psi_i\rangle|^2$. As before, we introduce a trace penalty and set a cost function given by

$$C = \sum_{i=0}^{2}(|\langle i|\psi_i\rangle|^2 - 1)^2 + \gamma P(\{|\Psi_i\rangle\}), \tag{48}$$

with $\gamma = 100$ for the regularization parameter. Additionally, we constrain the displacements in the input phase space to a circle of radius $|\alpha| = 1.5$ to make sure the encoding is as compact as possible.

Model performance. After training, the classical encoder element can be removed and we can analyze the quantum decoder by varying the displacements α applied to the vacuum. Figure 9 illustrates the resulting performance by showing the maximum fidelity between the output of the network and each of the three Fock states used for training. For the three Fock states $|0\rangle$, $|1\rangle$, and $|2\rangle$, the best matching input displacements each lead to a decoder output state with fidelity of 99.5%.

The hybrid network has learned to associate different areas of phase space with each of the three Fock states used for training. It is interesting to investigate the resultant output states from the quantum network when the vacuum is displaced to intermediate points between the three areas. These displacements can result in states that exhibit a transition between the Fock states. We use the wave function of the output

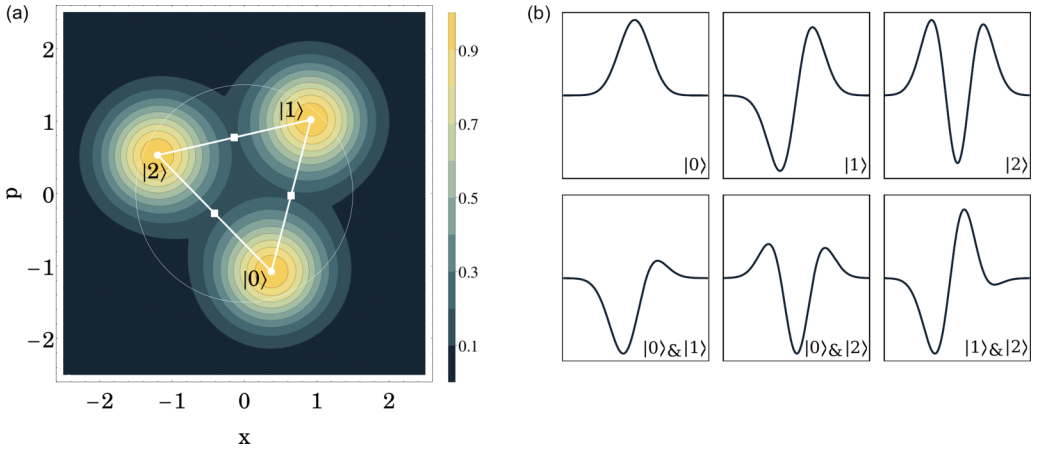

FIG. 9. Numerical experiment D. (Left) Learning a continuous phase-space encoding of the Fock states. The quantum decoder element of a trained classical-quantum autoencoder can be investigated by varying the displacement on the vacuum, which represents the chosen encoding method. The hybrid network has learned to encode the Fock states in different regions of phase space. This is illustrated by a contour plot showing, for each point in phase space, the largest fidelity between the output state for that displacement and the first three Fock states. The thin white circle represents a clipping applied to input displacements during training, i.e., so that no displacement can ever reach outside of the circle. The white circles at points $(0.37, -1.08)$, $(0.92, 1.02)$, and $(-1.20, 0.53)$ represent the input displacements leading to optimal fidelities with the $|0\rangle$, $|1\rangle$, and $|2\rangle$ Fock states, the white lines represent the lines interpolating these optimal displacements, and the white squares represent the halfway points. (Right) Visualizing the wave functions of output states. The top row represents the position wave functions of states with highest fidelity to $|0\rangle$, $|1\rangle$, and $|2\rangle$, respectively. The bottom row represents the wave functions of states with intermediate displacements between the points corresponding to $|0\rangle$ and $|1\rangle$, $|0\rangle$ and $|2\rangle$, $|1\rangle$ and $|2\rangle$, respectively. Each wave function is rescaled so that the maximum in absolute value is ± 1, while the x axis denotes positions in the range $[-4.5, 4.5]$.

states to visualize this transition. We plot on the right-hand side of Fig. 9 the output wave functions which give best fidelity to each of the three Fock states $|0\rangle$, $|1\rangle$, $|2\rangle$, respectively. Wave functions are also plotted for displacements which are the intermediate points between those corresponding to: $|0\rangle$ and $|1\rangle$; $|0\rangle$ and $|2\rangle$; and $|1\rangle$ and $|2\rangle$, respectively. These plots illustrate a smooth transition between the encoded Fock states in phase space.

V. CONCLUSIONS

We have presented a quantum neural network architecture which leverages the continuous-variable formalism of quantum computing, and explored it in detail through both theoretical exposition and numerical experiments. This scheme can be considered as an analog of recent proposals for neural networks encoded using classical light [31], with the additional ingredient that we leverage the quantum properties of the electromagnetic field. Interestingly, as light-based systems are already used in communication networks (both classical and quantum), an optical CV neural network could be wired up directly to communication channels, allowing us to avoid the costly interconversion of classical and quantum information.

Several challenges remain in the experimental implementation of quantum neural networks. The implementation of deterministic, tunable non-Gaussian gates with sufficient strength is an outstanding goal in experimental quantum optics, although several proposals and proof-of-principle experiments have been reported [122–125]. Additionally, as with all forms of quantum computing, it remains open whether quantum advantages can be realized in the presence of decoherence, or if error correction and fault tolerance are required. As experimental capabilities continue to progress, CV quantum neural networks can serve as a theoretical testbed to study quantum machine learning models, both in terms of their applications and differences compared to classical methods.

We have proposed variants for several well-known classical neural networks, specifically fully connected, convolutional, recurrent, and residual networks. We envision that in future work specialized neural networks will also be inspired purely from the quantum side. We have numerically analyzed the performance of quantum neural network models and demonstrated that they show promise in the tasks we considered. In several of these examples, we employed joint architectures, where classical and quantum networks are used together. This is another promising direction for future exploration, in particular given the current technological lead of classical computers and the expectation that near-term quantum hardware will be limited in size. The quantum part of the model can be specialized to process classically difficult parts of a larger computational to which it is naturally suited. In the longer term, as larger-scale quantum computers are built, the quantum component could take a larger role in hybrid models. Finally, it would be a fruitful research direction to explore the role that fundamental quantum physics concepts—such as symmetry, interference, entanglement, and the uncertainty principle—play in quantum neural networks more deeply.

ACKNOWLEDGMENTS

We thank Krishna Kumar Sabapathy, Haoyu Qi, Timjan Kalajdzievski, and Josh Izaac for helpful discussions. S.L. was supported by the Army Research Office under the Blue Sky program.

APPENDIX A: LINEAR INTERFEROMETERS

In this section, we derive Eq. (21) for the effect of a passive interferometer on the eigenstates $|\mathbf{x}\rangle$. A simple expression for an eigenstate of the $\hat{x}$ quadrature with eigenvalue x can be found in Appendix 4 of Ref. [126]

$$|x\rangle = \pi^{-1/4} \exp\left(-\frac{1}{2}x^2 + \sqrt{2}x\hat{a} - \frac{1}{2}a^{\dagger 2}\right)|0\rangle, \tag{A1}$$

where $\hat{a} = \frac{1}{\sqrt{2}}(\hat{x} + i\hat{p})$ is the bosonic annihilation operator, and $|0\rangle$ is the single mode vacuum state. The last expression is independent of any prefactors used to define the quadrature operator $\hat{x}$ in terms of $\hat{a}$ and $\hat{a}^\dagger$.

This can be easily generalized to N modes:

$$|\mathbf{x}\rangle = \bigotimes_{i=1}^{N} |x_i\rangle = \pi^{-\frac{N}{4}} \exp\left(-\frac{1}{2}\mathbf{x}^T\mathbf{x} + \sqrt{2}\mathbf{x}^T\hat{\boldsymbol{a}}^\dagger - \frac{1}{2}(\hat{\boldsymbol{a}}^\dagger)^T\hat{\boldsymbol{a}}^\dagger\right)|\mathbf{0}\rangle, \tag{A2}$$

where now

$$\mathbf{x} = (x_1, \ldots, x_N)^T, \tag{A3}$$

$$\hat{\boldsymbol{a}}^\dagger = (\hat{a}_1^\dagger, \ldots, \hat{a}_N^\dagger)^T, \quad (\hat{\boldsymbol{a}}^\dagger)^T = (\hat{a}_1^\dagger, \ldots, \hat{a}_N^\dagger), \tag{A4}$$

and $|\mathbf{0}\rangle$ is the multimode vacuum state. Now consider a (passive) linear optical transformation $\hat{\mathcal{U}}$

$$\hat{a}_i^\dagger \to \hat{\mathcal{U}}\hat{a}_i^\dagger\hat{\mathcal{U}}^\dagger = \sum_j U_{ij}\hat{a}_j^\dagger, \tag{A5}$$

$$\hat{\boldsymbol{a}}^\dagger \to U\hat{\boldsymbol{a}}^\dagger, \quad (\hat{\boldsymbol{a}}^\dagger)^T \to (\hat{\boldsymbol{a}}^\dagger)^T U^T. \tag{A6}$$

In general, U is an arbitrary unitary matrix, $UU^\dagger = \mathbb{1}_N$. We will however restrict U to have real entries and thus to be orthogonal. In this case, $U^\dagger = U^T$ and hence $U^T U = U U^T = \mathbb{1}_N$.

We can now examine how the multimode state $|\mathbf{x}\rangle$ transforms under such a linear interferometer $\hat{\mathcal{U}}$:

$$\begin{aligned}\hat{\mathcal{U}}|\mathbf{x}\rangle &= \hat{\mathcal{U}}\left[\pi^{-\frac{N}{4}} \exp\left(-\frac{1}{2}\mathbf{x}^T\mathbf{x} + \sqrt{2}\mathbf{x}^T\hat{\boldsymbol{a}}^\dagger - \frac{1}{2}(\hat{\boldsymbol{a}}^\dagger)^T\hat{\boldsymbol{a}}^\dagger\right)|\mathbf{0}\rangle\right] \\ &= \hat{\mathcal{U}}\frac{\exp\left(-\frac{1}{2}\mathbf{x}^T\mathbf{x} + \sqrt{2}\mathbf{x}^T\hat{\boldsymbol{a}}^\dagger - \frac{1}{2}(\hat{\boldsymbol{a}}^\dagger)^T\hat{\boldsymbol{a}}^\dagger\right)}{\pi^{\frac{N}{4}}}\hat{\mathcal{U}}^\dagger\hat{\mathcal{U}}|\mathbf{0}\rangle \\ &= \frac{\exp\left(\hat{\mathcal{U}}\left[-\frac{1}{2}\mathbf{x}^T\mathbf{x} + \sqrt{2}\mathbf{x}^T\hat{\boldsymbol{a}}^\dagger - \frac{1}{2}(\hat{\boldsymbol{a}}^\dagger)^T\hat{\boldsymbol{a}}^\dagger\right]\hat{\mathcal{U}}^\dagger\right)}{\pi^{\frac{N}{4}}}|\mathbf{0}\rangle.\end{aligned} \tag{A7}$$

We can use the transformation in Eq. (A5) to write

$$\hat{\mathcal{U}}|\mathbf{x}\rangle = \frac{\exp\left(-\frac{1}{2}\mathbf{x}^T\mathbf{x} + \sqrt{2}\mathbf{x}^T U\hat{\boldsymbol{a}}^\dagger - \frac{1}{2}(\hat{\boldsymbol{a}}^\dagger)^T U^T U\hat{\boldsymbol{a}}^\dagger\right)}{\pi^{\frac{N}{4}}}|\mathbf{0}\rangle. \tag{A8}$$

Now we use that $U^T U = UU^T = \mathbb{1}_N$ to write the last expression as

$$\hat{\mathcal{U}}|\mathbf{x}\rangle = \frac{\exp\left(-\frac{1}{2}\mathbf{x}^T UU^T\mathbf{x} + \sqrt{2}\mathbf{x}^T U\hat{\boldsymbol{a}}^\dagger - \frac{1}{2}(\hat{\boldsymbol{a}}^\dagger)^T\hat{\boldsymbol{a}}^\dagger\right)}{\pi^{\frac{N}{4}}}|\mathbf{0}\rangle. \tag{A9}$$

Let us define the vector $\boldsymbol{y} = U^T\mathbf{x}$ and, to match the notation of Eq. (21), the orthogonal matrix $C = U^T$, in terms of which we find

$$\begin{aligned}\hat{\mathcal{U}}|\mathbf{x}\rangle &= \frac{\exp\left(-\frac{1}{2}\mathbf{x}^T UU^T\mathbf{x} + \sqrt{2}\mathbf{x}^T U\hat{\boldsymbol{a}}^\dagger - \frac{1}{2}(\hat{\boldsymbol{a}}^\dagger)^T\hat{\boldsymbol{a}}^\dagger\right)}{\pi^{\frac{N}{4}}}|\mathbf{0}\rangle \\ &= \frac{\exp\left(-\frac{1}{2}\boldsymbol{y}^T\boldsymbol{y} + \sqrt{2}\boldsymbol{y}^T\hat{\boldsymbol{a}}^\dagger - \frac{1}{2}(\hat{\boldsymbol{a}}^\dagger)^T\hat{\boldsymbol{a}}^\dagger\right)}{\pi^{\frac{N}{4}}}|\mathbf{0}\rangle \\ &= |\boldsymbol{y}\rangle \\ &= |C\mathbf{x}\rangle.\end{aligned} \tag{A10}$$

Note that the output state is also a product state. This simple product transformation is a corollary of the elegant results of Ref. [103].

APPENDIX B: CONVOLUTIONAL NETWORKS

In this section, we derive the connection between a translationally invariant Hamiltonian and a Block Toeplitz symplectic transformation. The notion of translation symmetry and Toeplitz structure are both connected to one-dimensional convolutions. Two-dimensional convolutions, naturally appearing in image processing applications, are connected not with Toeplitz matrices, but with doubly block circulant matrices [16]. We will not consider this extension here, but the basic ideas are the same.

Suppose we have a Hamiltonian operator $\hat{H} = \hat{H}(\hat{\mathbf{x}}, \hat{\mathbf{p}})$ which generates a Gaussian unitary $\hat{U} = \exp(-it\hat{H})$ on N modes. We are interested only in the matrix multiplication part of an affine transformation, i.e., $\hat{H}$ does not generate displacements. Under these conditions, $\hat{H}$ has to be quadratic in the operators $(\hat{\mathbf{x}}, \hat{\mathbf{p}})$,

$$\hat{H} = [\hat{\mathbf{x}}^T \;\; \hat{\mathbf{p}}^T]\begin{bmatrix} H_{\mathbf{xx}} & H_{\mathbf{xp}} \\ H_{\mathbf{px}} & H_{\mathbf{pp}} \end{bmatrix}\begin{bmatrix} \hat{\mathbf{x}} \\ \hat{\mathbf{p}} \end{bmatrix}, \tag{B1}$$

where each $H_{\mathbf{uv}}$ is an $N \times N$ matrix. We will call the inner matrix in this equation $\widetilde{H}$. In the phase space picture, the symplectic transformation M_H generated by H is obtained via the rule [49]

$$M_H = \exp(\Omega\widetilde{H}), \tag{B2}$$

where Ω is the symplectic form from Eq. (8).

We now fix $\hat{H}$ to be translationally invariant, i.e., $\hat{H}$ does not change under the transformation

$$\begin{bmatrix} \hat{\mathbf{x}} \\ \hat{\mathbf{p}} \end{bmatrix} \mapsto \begin{bmatrix} T\hat{\mathbf{x}} \\ T\hat{\mathbf{p}} \end{bmatrix}, \tag{B3}$$

where we have introduced the shift operator T which maps $\hat{x}_i \mapsto \hat{x}_{i+1}$ and $\hat{p}_i \mapsto \hat{p}_{i+1}$. We assume periodic boundary conditions on the modes, $\hat{x}_N \mapsto \hat{x}_1$ and $\hat{p}_N \mapsto \hat{p}_1$, which allows us to represent translation as an $N \times N$ orthogonal matrix:

$$T = \sum_i |i+1\rangle\langle i|. \tag{B4}$$

The translationally-invariant condition on H translates to the statement that

$$[T, H_{\mathbf{uv}}] = 0 \tag{B5}$$

for $\mathbf{u}, \mathbf{v} \in \{\mathbf{x}, \mathbf{p}\}$.

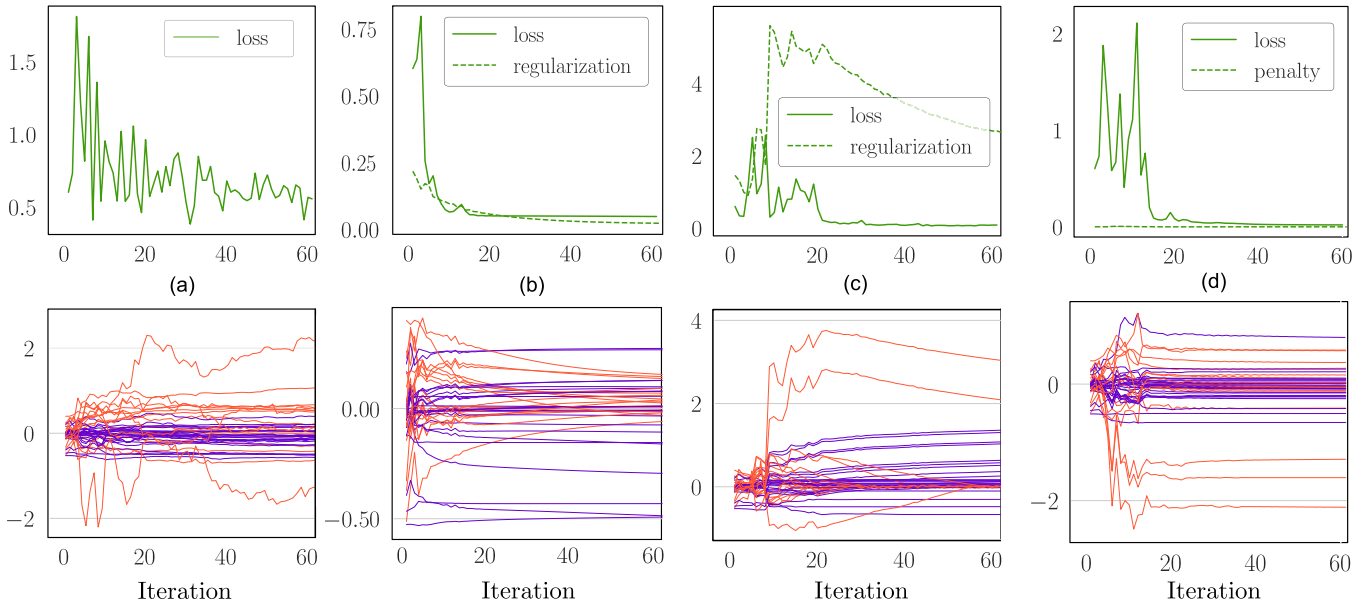

FIG. 10. Cost function and circuit parameters during 60 steps of stochastic gradient descent training for the task of fitting the sine function from Fig. 6. The active parameters are plotted in orange, while all others are plotted in purple. As hyperparameters, we used an initial learning rate of 0.1 which has an inverse decay of 0.25, a penalty strength $\gamma = 10$, a regularization strength of 0.5, batch size of 50, a cutoff of 10 for the Hilbert-space dimension, and randomly chosen but fixed initial circuit parameters.

In the $2N$-dimensional phase space, the N-dimensional translation matrix takes the form $T \oplus T$. Considering the expression

$$[\Omega\widetilde{H}, T \oplus T] = \begin{bmatrix} [M_{\mathbf{px}}, T] & [M_{\mathbf{pp}}, T] \\ -[M_{\mathbf{xx}}, T] & -[M_{\mathbf{xp}}, T] \end{bmatrix} = 0, \tag{B6}$$

we see that the symplectic matrix M_H from Eq. (B2) must also be symmetric under translations:

$$[M_H, T \oplus T] = 0. \tag{B7}$$

Writing this matrix in a block form,

$$M_H = \begin{bmatrix} M_{\mathbf{xx}} & M_{\mathbf{xp}} \\ M_{\mathbf{px}} & M_{\mathbf{pp}} \end{bmatrix}, \tag{B8}$$

we conclude that we must also have

$$[M_{\mathbf{uv}}, T] = 0 \tag{B9}$$

for each $\mathbf{u}, \mathbf{v} \in \{\mathbf{x}, \mathbf{p}\}$. Expressing this in the equivalent form

$$T^T M_{\mathbf{uv}} T = M_{\mathbf{uv}}, \tag{B10}$$

we see that the following condition must hold on the entries of each $M_{\mathbf{uv}}$:

$$[M_{\mathbf{uv}}]_{ij} = [M_{\mathbf{uv}}]_{i+1,j+1}. \tag{B11}$$

In other words, when the generating Hamiltonian is translationally invariant, each block of the corresponding symplectic matrix is a Toeplitz matrix, which implements a one-dimensional convolution.

APPENDIX C: REGULARIZATION AND OPTIMIZATION

Penalties and regularization. In the numerical simulations of quantum circuits, each qumode is truncated to a given cutoff dimension in the infinite-dimensional Hilbert space of Fock states. During training, it is possible for the gate parameters to reach values such that the output states have significant support outside of the truncated Hilbert space. In a simulation, this results in unnormalized output states and unreliable computations. To address this issue, we add a penalty to the loss function that penalizes unnormalized quantum states. Given a set of output states $\{|\psi_{x_i}\rangle\}$, we define the penalty function

$$P(\{|\psi_{x_i}\rangle\}) = \sum_i \left(|\langle\psi_{x_i}|\hat{\Pi}_{\mathcal{H}}|\psi_{x_i}\rangle|^2 - 1\right)^2, \tag{C1}$$

where $\hat{\Pi}_{\mathcal{H}}$ is a projector onto the truncated Hilbert space of the simulation. This function penalizes unnormalized states whose trace is different to one. The overall cost function to be minimized is then

$$C = L + \gamma P(\{|\psi_{x_i}\rangle\}), \tag{C2}$$

where $\gamma > 0$ is a user-defined hyperparameter.

An alternate approach to the trace penalty is to regularize the circuit parameters that can alter the energy of the state, which we refer to as the active parameters. Using the curve fitting example from Sec. IV B, Fig. 10 compares optimizing the function of Eq. (C2) without any penalty (first column from the left), imposing an L2 regularizer (second column), using an L1 regularizer (third column), and using the trace penalty (fourth column). Without any strategy to keep the parameters small, learning fails due to unstable simulations: the trace of the state drops in fact to 0.1. Both regularization strategies as well as the trace penalty manage to bring the loss function to almost zero within a few steps while maintaining the unit trace of the state. However, there are interesting differences. While L2 regularization decreases the magnitude of

the active parameters, L1 regularization dampens all but two of them. The undamped parameters turn out to be the circuit parameters for the nonlinear gates in layers 3 and 4, a hint that these nonlinearities are most essential for the task. The trace penalty induces heavy fluctuations in the loss function for the first 20 steps, but finds parameters that are larger in absolute value than those found by L2 regularization, with a lower final loss.

Optimization methods. We also analyzed different optimization algorithms for the sine curve-fitting problem. Figure 11 compares three numerical methods and two methods based on automatic differentiation in a simulator. Numerical stochastic gradient descent (SGD) approximates the gradients with a finite differences estimate. Nelder-Mead is a gradient-free technique, while the sequential least-squares programming (SLSQP) method solves quadratic subproblems with approximate gradients. These latter two converge significantly slower, but can have advantages in smoothness and speed per iteration. The Adam optimizer with adaptive learning rate performed better than vanilla SGD in this experiment.

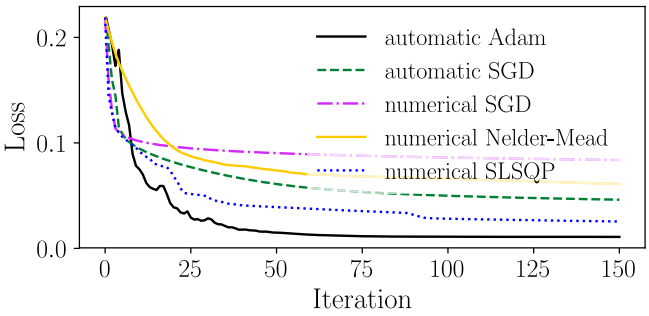

FIG. 11. Loss function of the different optimizers mentioned in the text for the curve-fitting task.

[1] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, Nature (London) **549**, 195 (2017).

[2] P. Wittek, *Quantum Machine Learning: What Quantum Computing Means to Data Mining* (Academic Press, San Diego, 2014).

[3] M. Schuld and F. Petruccione, *Supervised Learning with Quantum Computers* (Springer International Publishing, Switzerland, 2018).

[4] N. Wiebe, D. Braun, and S. Lloyd, Quantum Algorithm for Data Fitting, Phys. Rev. Lett. **109**, 050505 (2012).

[5] S. Lloyd, M. Mohseni, and P. Rebentrost, Quantum principal component analysis, Nat. Phys. **10**, 631 (2014).

[6] G. H. Low, T. J. Yoder, and I. L. Chuang, Quantum inference on Bayesian networks, Phys. Rev. A **89**, 062315 (2014).

[7] N. Wiebe and C. Granade, Can small quantum systems learn? Quantum Info. Comput. **17**, 568 (2017).

[8] A. Montanaro, Quantum speedup of monte carlo methods, Proc. R. Soc. A **471**, 20150301 (2015).

[9] P. Rebentrost, M. Mohseni, and S. Lloyd, Quantum Support Vector Machine for Big Data Classification, Phys. Rev. Lett. **113**, 130503 (2014).

[10] M. Schuld and N. Killoran, Quantum Machine Learning in Feature Hilbert Spaces, Phys. Rev. Lett. **122**, 040504 (2019).

[11] M. H. Amin, E. Andriyash, J. Rolfe, B. Kulchytskyy, and R. Melko, Quantum Boltzmann Machine, Phys. Rev. X **8**, 021050 (2018).

[12] M. Kieferova and N. Wiebe, Tomography and generative data modeling via quantum Boltzmann training, Phys. Rev. A **96**, 062327 (2017).

[13] I. Kerenidis and A. Prakash, Quantum recommendation systems, in *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)* (Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017).

[14] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, Nature (London) **521**, 436 (2015).

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

[16] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning* (MIT Press, Cambridge, 2016), Vol. 1.

[17] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, Theano: A CPU and GPU math compiler in Python, in *Proc. 9th Python in Science Conference* (2010), Vol. 1.

[18] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, Caffe: Convolutional architecture for fast feature embedding, in *Proceedings of the 22nd ACM international conference on Multimedia* (ACM, New York, 2014), pp. 675–678.

[19] D. Maclaurin, D. Duvenaud, and R. P. Adams, Autograd: Effortless gradients in Numpy, in *ICML 2015 AutoML Workshop* (2015).

[20] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, TENSORFLOW: Large-scale machine learning on heterogeneous distributed systems, arXiv:1603.04467.

[21] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, Automatic differentiation in PYTORCH (2017).

[22] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, cuDNN: Efficient primitives for deep learning, arXiv:1410.0759.

[23] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, In-datacenter performance analysis of a tensor processing unit, in *Proceedings of the 44th Annual International Symposium on Computer Architecture* (ACM, New York, 2017), pp. 1–12.

[24] C. Mead, Neuromorphic electronic systems, Proc. IEEE **78**, 1629 (1990).

[25] C.-S. Poon and K. Zhou, Neuromorphic silicon neurons and large-scale neural networks: challenges and opportunities, Front. Neurosci. **5**, 108 (2011).

[26] L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer, Information processing using a single dynamical node as complex system, Nat. Commun. **2**, 468 (2011).

[27] A. N. Tait, M. A. Nahmias, B. J. Shastri, and P. R. Prucnal, Broadcast and weight: an integrated network for scalable photonic spike processing, J. Lightwave Technol. **32**, 4029 (2014).

[28] D. Monroe, Neuromorphic computing gets ready for the (really) big time, Commun. ACM **57**, 13 (2014).

[29] A. N. Tait, M. A. Nahmias, Y. Tian, B. J. Shastri, and P. R. Prucnal, Photonic neuromorphic signal processing and computing, in *Nanophotonic Information Physics* (Springer, Berlin, Heidelberg, 2014), pp. 183–222.

[30] K. Vandoorne, P. Mechet, T. Van Vaerenbergh, M. Fiers, G. Morthier, D. Verstraeten, B. Schrauwen, J. Dambre, and P. Bienstman, Experimental demonstration of reservoir computing on a silicon photonics chip, Nat. Commun. **5**, 3541 (2014).

[31] Y. Shen, N. C. Harris, S. Skirlo, M. Prabhu, T. Baehr-Jones, M. Hochberg, X. Sun, S. Zhao, H. Larochelle, D. Englund *et al.*, Deep learning with coherent nanophotonic circuits, Nat. Photonics **11**, 441 (2017).

[32] J. Romero, J. P. Olson, and A. Aspuru-Guzik, Quantum autoencoders for efficient compression of quantum data, Quantum Sci. Technol. **2**, 045001 (2017).

[33] K. H. Wan, O. Dahlsten, H. Kristjánsson, R. Gardner, and M. Kim, Quantum generalisation of feedforward neural networks, npj Quantum Inf. **3**, 36 (2017).

[34] G. Verdon, M. Broughton, and J. Biamonte, A quantum algorithm to train neural networks using low-depth circuits, arXiv:1712.05304.

[35] E. Farhi and H. Neven, Classification with quantum neural networks on near term processors, arXiv:1802.06002.

[36] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, Quantum circuit learning, Phys. Rev. A **98**, 032309 (2018).

[37] M. Schuld, A. Bocharov, K. Svore, and N. Wiebe, Circuit-centric quantum classifiers, arXiv:1804.00633.

[38] E. Grant, M. Benedetti, S. Cao, A. Hallam, J. Lockhart, V. Stojevic, A. G. Green, and S. Severini, Hierarchical quantum classifiers, npj Quantum Inf. **4**, 65 (2018).

[39] H. Chen, L. Wossnig, S. Severini, H. Neven, and M. Mohseni, Universal discriminative quantum neural networks, arXiv:1805.08654.

[40] G. Adesso, S. Ragy, and A. R. Lee, Continuous variable quantum information: Gaussian states and beyond, Open Syst. Inf. Dynamics **21**, 1440001 (2014).

[41] A. Perdomo-Ortiz, M. Benedetti, J. Realpe-Gómez, and R. Biswas, Opportunities and challenges for quantum-assisted machine learning in near-term quantum computers, Quantum Sci. Technol. **3**, 030502 (2018).

[42] M. Benedetti, J. R. Gómez, and A. Perdomo-Ortiz, Quantum-assisted Helmholtz machines: A quantum-classical deep learning framework for industrial datasets in near-term devices, Quantum Sci. Technol. **3**, 034007 (2018).

[43] D. Gottesman, A. Kitaev, and J. Preskill, Encoding a qubit in an oscillator, Phys. Rev. A **64**, 012310 (2001).

[44] E. Knill, R. Laflamme, and G. J. Milburn, A scheme for efficient quantum computation with linear optics, Nature (London) **409**, 46 (2001).

[45] H.-K. Lau, R. Pooser, G. Siopsis, and C. Weedbrook, Quantum Machine Learning Over Infinite Dimensions, Phys. Rev. Lett. **118**, 080501 (2017).

[46] S. Das, G. Siopsis, and C. Weedbrook, Continuous-variable quantum Gaussian process regression and quantum singular value decomposition of nonsparse low-rank matrices, Phys. Rev. A **97**, 022315 (2018).

[47] A. Ferraro, S. Olivares, and M. G. A. Paris, Gaussian states in continuous variable quantum information, arXiv:quant-ph/0503237.

[48] C. Weedbrook, S. Pirandola, R. García-Patrón, N. J. Cerf, T. C. Ralph, J. H. Shapiro, and S. Lloyd, Gaussian quantum information, Rev. Mod. Phys. **84**, 621 (2012).

[49] A. Serafini, *Quantum Continuous Variables: A Primer of Theoretical Methods* (CRC Press, Boca Raton, FL, 2017).

[50] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, Backpropagation applied to handwritten zip code recognition, Neural Comput. **1**, 541 (1989).

[51] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning representations by back-propagating errors, Nature (London) **323**, 533 (1986).

[52] J. Ba, V. Mnih, and K. Kavukcuoglu, Multiple object recognition with visual attention, arXiv:1412.7755.

[53] A. Graves, G. Wayne, and I. Danihelka, Neural turing machines, arXiv:1410.5401.

[54] K. Hornik, M. Stinchcombe, and H. White, Multilayer feedforward networks are universal approximators, Neural Networks **2**, 359 (1989).

[55] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Syst. **2**, 303 (1989).

[56] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, Multilayer feedforward networks with a nonpolynomial activation function can approximate any function, Neural Networks **6**, 861 (1993).

[57] W. Maass, G. Schnitger, and E. D. Sontag, A comparison of the computational power of sigmoid and boolean threshold circuits, in *Theoretical Advances in Neural Computation and Learning* (Springer, Boston, MA, 1994), pp. 127–151.

[58] G. F. Montúfar, Universal approximation depth and errors of narrow belief networks with discrete units, Neural Comput. **26**, 1386 (2014).

[59] H. W. Lin, M. Tegmark, and D. Rolnick, Why does deep and cheap learning work so well? J. Stat. Phys. **168**, 1223 (2017).

[60] L. Bottou, On-line learning and stochastic approximations, in *On-Line Learning in Neural Networks*, edited by D. Saad (Cambridge University Press, Cambridge, 1999), Chap. 2, pp. 9–42.

[61] P. Rotondo, M. Marcuzzi, J. P. Garrahan, I. Lesanovsky, and M. Müller, Open quantum generalisation of hopfield neural networks, J. Phys. A: Math. Theor. **51**, 115301 (2018).

[62] D. Ventura and T. Martinez, Quantum associative memory, Inf. Sci. **124**, 273 (2000).

[63] N. Wiebe, A. Kapoor, and K. M. Svore, Quantum deep learning, Quantum Info. Comput. **16**, 541 (2016).

[64] M. Schuld, I. Sinayskiy, and F. Petruccione, The quest for a quantum neural network, Quantum Inf. Proc. **13**, 2567 (2014).

[65] Y. Cao, G. G. Guerreschi, and A. Aspuru-Guzik, Quantum neuron: an elementary building block for machine learning on quantum computers, arXiv:1711.11240.

[66] T Espinosa-Ortega and T. C. H. Liew, Perceptrons with Hebbian Learning Based on Wave Ensembles in Spatially Patterned Potentials, Phys. Rev. Lett. **114**, 118101 (2015).

[67] M. Schuld, I. Sinayskiy, and F. Petruccione, Simulating a perceptron on a quantum computer, Phys. Lett. A **379**, 660 (2015).
[68] E. Torrontegui and J. J. Garcia-Ripoll, Universal quantum perceptron as efficient unitary approximators, EPL **125**, 30004 (2019).
[69] B. Ricks and D. Ventura, Training a quantum neural network, in *Advances in Neural Information Processing Systems* (MIT Press, Cambridge, MA, 2003), Vol. 16, pp. 1–8.
[70] I. Cong, S. Choi, and M. D. Lukin, Quantum convolutional neural networks, arXiv:1810.03787.
[71] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, The theory of variational hybrid quantum-classical algorithms, New J. Phys. **18**, 023023 (2016).
[72] N. Tezak and H. Mabuchi, A coherent perceptron for all-optical learning, EPJ Quantum Technology **2**, 10 (2015).
[73] J. Binas, D. Neil, G. Indiveri, S.-C. Liu, and M. Pfeiffer, Precise deep neural network computation on imprecise low-power analog hardware, arXiv:1606.07786.
[74] S. Ambrogio, P. Narayanan, H. Tsai, R. M. Shelby, I. Boybat, C. Nolfo, S. Sidler, M. Giordano, M. Bodini, N. C. P. Farinha *et al.*, Equivalent-accuracy accelerated neural-network training using analogue memory, Nature (London) **558**, 60 (2018).
[75] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars, ACM SIGARCH Computer Architecture News **44**, 14 (2016).
[76] U. L. Andersen, J. S. Neergaard-Nielsen, P. Van Loock, and A. Furusawa, Hybrid discrete- and continuous-variable quantum information, Nat. Phys. **11**, 713 (2015).
[77] Jun-ichi Yoshikawa, S. Yokoyama, T. Kaji, C. Sornphiphatphong, Y. Shiozawa, K. Makino, and A. Furusawa, Invited article: Generation of one-million-mode continuous-variable cluster state by unlimited time-domain multiplexing, APL Photonics **1**, 060801 (2016).
[78] K. Moon and S. M. Girvin, Theory of Microwave Parametric Down-Conversion and Squeezing using Circuit QED, Phys. Rev. Lett. **95**, 140504 (2005).
[79] B. Peropadre, G. G. Guerreschi, J. Huh, and A. Aspuru-Guzik, Proposal for Microwave Boson Sampling, Phys. Rev. Lett. **117**, 140505 (2016).
[80] S. M. Girvin, Schrodinger cat states in circuit QED, arXiv:1710.03179.
[81] H.-K. Lau and D. F. V. James, Proposal for a scalable universal bosonic simulator using individually trapped ions, Phys. Rev. A **85**, 062329 (2012).
[82] C. Shen, Z. Zhang, and L.-M. Duan, Scalable Implementation of Boson Sampling with Trapped Ions, Phys. Rev. Lett. **112**, 050504 (2014).
[83] D. M. Meekhof, C. Monroe, B. E. King, W. M. Itano, and D. J. Wineland, Generation of Nonclassical Motional States of a Trapped Atom, Phys. Rev. Lett. **76**, 1796 (1996).
[84] Ch. Monroe, D. M. Meekhof, B. E. King, and D. J. Wineland, A "Schrödinger Cat" superposition state of an atom, Science **272**, 1131 (1996).
[85] E. Schrödinger, Quantisierung als Eigenwertproblem, Ann. Phys. **385**, 437 (1926).
[86] E. Schrödinger, An undulatory theory of the mechanics of atoms and molecules, Phys. Rev. **28**, 1049 (1926).
[87] H. Weyl, Quantenmechanik und Gruppentheorie, Z. Phys. **46**, 1 (1927).
[88] E. Wigner, On the quantum correction for thermodynamic equilibrium, Phys. Rev. **40**, 749 (1932).
[89] H. J. Groenewold, On the principles of elementary quantum mechanics, in *On the Principles of Elementary Quantum Mechanics* (Springer, Dordrecht, 1946), pp. 1–56.
[90] J. E. Moyal, Quantum mechanics as a statistical theory, in *Mathematical Proceedings of the Cambridge Philosophical Society* (Cambridge University Press, Cambridge, 1949), Vol. 45, pp. 99–124.
[91] S. Lloyd and S. L. Braunstein, Quantum Computation Over Continuous Variables, Phys. Rev. Lett. **82**, 1784 (1999).
[92] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, A variational eigenvalue solver on a photonic quantum processor, Nat. Commun. **5**, 4213 (2014).
[93] N. Moll, P. Barkoutsos, L. S. Bishop, J. M. Chow, A. Cross, D. J. Egger, S. Filipp, A. Fuhrer, J. M. Gambetta, M. Ganzhorn *et al.*, Quantum optimization using variational algorithms on near-term quantum devices, Quantum Sci. Technol. **3**, 030503 (2018).
[94] P.-L. Dallaire-Demers and N. Killoran, Quantum generative adversarial networks, Phys. Rev. A **98**, 012324 (2018).
[95] M. Benedetti, E. Grant, L. Wossnig, and S. Severini, Adversarial quantum circuit learning for pure state approximation, New J. Phys. **21**, 043023 (2019).
[96] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, Supervised learning with quantum-enhanced feature spaces, Nature (London) **567**, 209 (2019).
[97] P. Kok and B. W. Lovett, *Introduction to Optical Quantum Information Processing* (Cambridge University Press, Cambridge, 2010).
[98] F. Bloch, Über die Quantenmechanik der Elektronen in Kristallgittern, Z. Phys. **52**, 555 (1929).
[99] E. Noether, Invariante Variationsprobleme, Nachr. Ges. Wiss. Goettingen, Math. Phys. Kl. **1918**, 235 (1918).
[100] A. Graves, Supervised sequence labeling with recurrent neural networks, Ph.D. thesis, Technische Universität München, 2012.
[101] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, in *Proc. IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2016), pp. 770–778.
[102] Xanadu Quantum Technologies, STRAWBERRY FIELDS online documentation: Controlled-X gate, https://strawberryfields.readthedocs.io/en/latest/conventions/gates.html#controlled-x-gate (2018) [Online; accessed 15-June-2018].
[103] Z. Jiang, M. D. Lang, and C. M. Caves, Mixing nonclassical pure states in a linear-optical network almost always generates modal entanglement, Phys. Rev. A **88**, 044301 (2013).
[104] M. E. S. Morales, T. Tlyachev, and J. Biamonte, Variational learning of Grover's quantum search algorithm, Phys. Rev. A **98**, 062333 (2018).
[105] J. M. Arrazola, T. R. Bromley, J. Izaac, C. R. Myers, K. Brádler, and N. Killoran, Machine learning method for state

preparation and gate synthesis on photonic quantum computers, Quantum Sci. Technol. **4**, 024004 (2019).

[106] C. S. Hamilton, R. Kruse, L. Sansoni, S. Barkhofen, C. Silberhorn, and I. Jex, Gaussian Boson Sampling, Phys. Rev. Lett. **119**, 170501 (2017).

[107] A. P. Lund, A. Laing, S. Rahimi-Keshari, T Rudolph, J. L. O'Brien, and T. C. Ralph, Boson Sampling from a Gaussian State, Phys. Rev. Lett. **113**, 100502 (2014).

[108] R. Kruse, C. S. Hamilton, L. Sansoni, S. Barkhofen, C. Silberhorn, and I. Jex, A detailed study of Gaussian Boson Sampling, Phys. Rev. A **100**, 032326 (2019).

[109] T. Douce, D. Markham, E. Kashefi, E. Diamanti, T. Coudreau, P. Milman, P. van Loock, and G. Ferrini, Continuous-Variable Instantaneous Quantum Computing is Hard to Sample, Phys. Rev. Lett. **118**, 070503 (2017).

[110] S. Aaronson and A. Arkhipov, The computational complexity of linear optics, in *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing* (ACM, New York, 2011), pp. 333–342.

[111] J. M. Arrazola, P. Rebentrost, and C. Weedbrook, Quantum supremacy and high-dimensional integration, arXiv:1712.07288.

[112] S. Lloyd, Universal quantum simulators, Science **273**, 1073 (1996).

[113] D. J. Rezende and S. Mohamed, Variational inference with normalizing flows, in *International Conference on Machine Learning* (2015), pp. 1530–1538.

[114] J. V. Dillon, I. Langmore, D. Tran, E. Brevdo, S. Vasudevan, D. Moore, B. Patton, A. Alemi, M. Hoffman, and R. A. Saurous, TENSORFLOW distributions, arXiv:1711.10604.

[115] K. Kawamura, The Perron-Frobenius operators, invariant measures and representations of the Cuntz-Krieger algebras, J. Math. Phys. **46**, 083514 (2005).

[116] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.

[117] N. Killoran, J. Izaac, N. Quesada, V. Bergholm, M. Amy, and C. Weedbrook, Strawberry fields: A software platform for photonic quantum computing, Quantum **3**, 129 (2019).

[118] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, Evaluating analytic gradients on quantum hardware, Phys. Rev. A **99**, 032331 (2019).

[119] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, C. Blank, K. McKiernan, and N. Killoran, Pennylane: Automatic differentiation of hybrid quantum-classical computations, arXiv:1811.04968.

[120] A. Dal Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi, Calibrating probability with undersampling for unbalanced classification, in *2015 IEEE Symposium Series on Computational Intelligence* (IEEE, 2015), pp. 159–166.

[121] D. P. Kingma and M. Welling, Auto-encoding variational Bayes, arXiv:1312.6114.

[122] P. Marek, R. Filip, and A. Furusawa, Deterministic implementation of weak quantum cubic nonlinearity, Phys. Rev. A **84**, 053802 (2011).

[123] K. Marshall, R. Pooser, G. Siopsis, and C. Weedbrook, Repeat-until-success cubic phase gate for universal continuous-variable quantum computation, Phys. Rev. A **91**, 032321 (2015).

[124] K. Miyata, H. Ogawa, P. Marek, R. Filip, H. Yonezawa, J.-i. Yoshikawa, and A. Furusawa, Implementation of a quantum cubic gate by an adaptive non-Gaussian measurement, Phys. Rev. A **93**, 022301 (2016).

[125] P. Marek, R. Filip, H. Ogawa, A. Sakaguchi, S. Takeda, J.-i. Yoshikawa, and A. Furusawa, General implementation of arbitrary nonlinear quadrature phase gates, Phys. Rev. A **97**, 022329 (2018).

[126] S. M. Barnett and P. M. Radmore, *Methods in Theoretical Quantum Optics* (Oxford University Press, Oxford, 2002), Vol. 15.