Parameterized quantum circuits as machine learning models

Marcello Benedetti,[1,2,*] Erika Lloyd,[1] Stefan Sack,[1] and Mattia Fiorentini[1]

[1] *Cambridge Quantum Computing Limited, CB2 1UB Cambridge, United Kingdom*
[2] *Department of Computer Science, University College London, WC1E 6BT London, United Kingdom*

(Dated: October 10, 2019)

Hybrid quantum-classical systems make it possible to utilize existing quantum computers to their fullest extent. Within this framework, parameterized quantum circuits can be regarded as machine learning models with remarkable expressive power. This Review presents the components of these models and discusses their application to a variety of data-driven tasks, such as supervised learning and generative modeling. With an increasing number of experimental demonstrations carried out on actual quantum hardware and with software being actively developed, this rapidly growing field is poised to have a broad spectrum of real-world applications.

I. INTRODUCTION

Developments in material science, hardware manufacturing, and disciplines such as error-correction and compilation, have brought us one step closer to large-scale, fault-tolerant, universal quantum computers. However, this process is incremental and may take years. In fact, existing quantum hardware implements few tens of physical qubits and can only perform short sequences of gates before being overwhelmed by noise. In such a setting, much anticipated algorithms such as Shor's remain out of reach. Nevertheless, there is a growing consensus that noisy intermediate-scale quantum (NISQ) devices may find useful applications and commercialization in the near term [1, 2]. As prototypes of quantum computers are made available to researchers for experimentation, algorithmic research is adapting to match the pace of hardware development.

Parameterized quantum circuits (PQCs) offer a concrete way to implement algorithms and demonstrate quantum supremacy in the NISQ era. PQCs are typically composed of fixed gates, e.g., controlled NOTs, and adjustable gates, e.g., qubit rotations. Even at low circuit depth, some classes of PQCs are capable of generating highly non-trivial outputs. For example, under well-believed complexity-theoretic assumptions, the class of PQCs called instantaneous quantum polynomial-time cannot be efficiently simulated by classical resources (see Lund *et al.* [3] and Harrow and Montanaro [4] for accessible Reviews of quantum supremacy proposals). The demonstration of quantum supremacy is an important milestone in the development of quantum computers. In practice, however, it is highly desirable to demonstrate a quantum advantage on applications.

The main approach taken by the community consists in formalizing problems of interest as variational optimization problems and use hybrid systems of quantum and classical hardware to find approximate solutions. The intuition is that by implementing some subroutines on classical hardware, the requirement of quantum resources is significantly reduced, particularly the number of qubits, circuit depth, and coherence time. Therefore, in the hybrid algorithmic approach NISQ hardware focuses entirely on the classically intractable part of the problem.

The hybrid approach turned out to be successful in attacking scaled-down problems in chemistry, combinatorial optimization and machine learning. For example, the variational quantum eigensolver (VQE) [5] has been used for searching the ground state of the electronic Hamiltonian of molecules [6, 7]. Similarly, the quantum approximate optimization algorithm (QAOA) [8] has been used to find approximate solutions of classical Ising models [9] and clustering problems formulated as MaxCut [10]. The focus of this Review is on hybrid approaches for machine learning. In this field, quantum circuits are seen as components of a *model* for some data-driven task. *Learning* describes the process of iteratively updating the model's parameters towards the goal.

The general hybrid approach is illustrated in Fig. 1 and is made of three main components: the human, the classical computer, and the quantum computer. The human interprets the problem information and selects an initial model to represent it. The data is pre-processed on a classical computer to determine a set of parameters for the PQC. The quantum hardware prepares a quantum state as prescribed by a PQC and performs measurements. Measurement outcomes are post-processed by the classical computer to generate a forecast. To improve the forecast, the classical computer implements a learning algorithm that updates the model's parameters. The overall algorithm is run in a closed loop between the classical and quantum hardware. The human supervises the process and uses forecasts towards the goal.

To the best of our knowledge, the earliest hybrid systems were proposed in the context of quantum algorithm learning. Bang *et al.* [11] described a method where a classical computer controls the unitary operation implemented by a quantum device. Each execution of the quantum device is deemed as either a 'success' or 'failure', and the classical algorithm adjusts the unitary operation towards its target. Starting from a dataset of input-

* marcello.benedetti@cambridgequantum.com

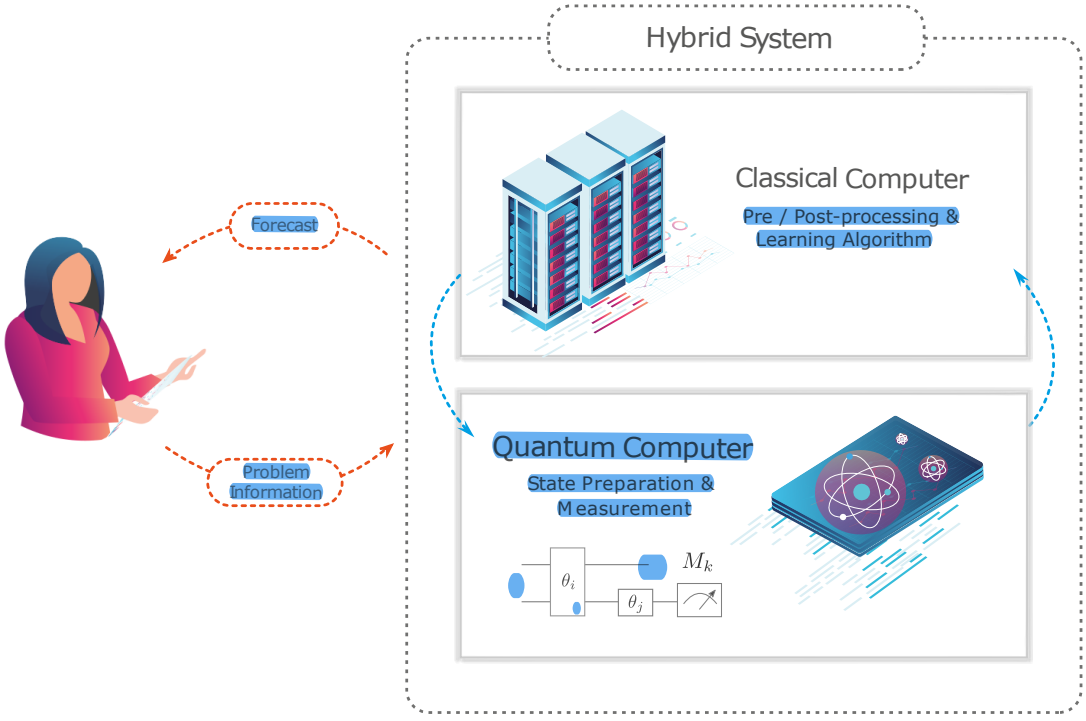

FIG. 1. High-level depiction of hybrid algorithms used for machine learning. The role of the human is to set up the model using prior information, assess the learning process, and exploit the forecasts. Within the hybrid system, the quantum computer prepares quantum states according to a set of parameters. Using the measurement outcomes, the classical learning algorithm adjusts the parameters in order to minimize an objective function. The updated parameters, now defining a new quantum circuit, are fed back to the quantum hardware in a closed loop.

output pairs their simulated system learns an equivalent of Deutsch's algorithm for finding whether a function is constant or balanced. Gammelmark and Mølmer [12] took a more general approach in which the parameters of the quantum system are quantized as well. In their simulations they successfully learn Grover's search and Shor's integer factorization algorithms.

These early proposals attacked problems that are well known within the quantum computing community but much less known among machine learning researchers. More recently though, the hybrid approach based on PQCs has been shown to perform well on machine learning tasks such as classification, regression, and generative modeling. The success is in part due to similarities between PQCs and celebrated classical models such as kernel methods and neural networks.

In the following Sections we introduce many of these multidisciplinary ideas, and we direct the Readers towards the relevant literature. Our style of exposition is pedagogical and not overly-technical, although we assume familiarity with basic machine learning definitions and methods (see Mehta *et al.* [13] for a physics-oriented introduction to machine learning), and basic working knowledge on quantum computing (see Nielsen and Chuang [14], Chapter 2, for an introduction). We make use of several acronyms when referring to models and algorithms; to help the Reader we summarize all the acronyms in Table I.

The structure of the Review is as follows: in Section II we describe the components of machine learning models based on PQCs and their learning algorithms; in Section III we describe their applications to classical and quantum tasks; and in Section IV we summarize the advantages of this approach and give an outlook of the field.

MERA	multi-scale entanglement renormalization ansatz
NISQ	noisy intermediate-scale quantum
PAC	probably approximately correct
PQC	parameterized quantum circuit
QAE	quantum autoencoder
QAOA	quantum approximate optimization algorithm
QCBM	quantum circuit Born machine
QKE	quantum kernel estimator
QGAN	quantum generative adversarial network
SPSA	simultaneous perturbation stochastic approximation
TTN	tree tensor network
VQM	variational quantum model
VQE	variational quantum eigensolver

TABLE I. Acronyms used in this Review.

II. FRAMEWORK

We assume the computer to be a closed quantum system. With n qubits, its state can be described as a unit vector living in a complex inner product vector space $\mathbb{C}^{2^n}$. The computation always starts with a state of simple preparation in the computational basis, for example the product state $|0\rangle^{\otimes n}$ (when clear from the context we often drop the tensor notation and refer to this state simply as $|0\rangle$). A unitary operator U is applied to the initial state producing a new state $U|0\rangle$. Here, the value of an observable quantity can be measured. Physical observables are associated with Hermitian operators. Let $M = \sum_i \lambda_i P_i$ be the Hermitian operator of interest, where λ_i is the i-th eigenvalue and P_i is the projector on the corresponding eigenspace. The *Born rule* states that the outcome of the measurement corresponds to one of the eigenvalues and follows probability distribution $p(\lambda_i) = \mathrm{tr}\big(P_i U|0\rangle\langle 0|U^\dagger\big)$. Plugging this in the definition of expectation values we obtain

$$\langle M\rangle = \sum_i \lambda_i p(\lambda_i) = \mathrm{tr}\big(MU|0\rangle\langle 0|U^\dagger\big). \qquad (1)$$

As we will see, one can exploit the probabilistic nature of quantum measurements to define a variety of machine learning models, and PQCs offer a concrete way to implement adjustable unitary operators U.

Figure 2 shows the components of a supervised learning model based on a PQC. First, a data vector is sampled from the training set and transformed by classical pre-processing, for example with de-correlation and standardization functions. Second, the transformed data point is mapped to the parameters of an *encoder circuit* $U_{\phi(\boldsymbol{x})}$. Third, a *variational circuit* $U_{\boldsymbol{\theta}}$, which possibly acts on an extended qubit register, implements the core operation of the model. This is followed by the estimation of a set of expectation values $\{\langle M_k\rangle_{\boldsymbol{x},\boldsymbol{\theta}}\}_{k=1}^K$ from measurements [1].

A post-processing function f is then applied to this set in order to provide a suitable output for the task. As an example, if we were to perform regression, f could be a linear combination of the kind $\sum_k w_k \langle M_k\rangle_{\boldsymbol{x},\boldsymbol{\theta}}$, with additional parameters w_k. Note that it is possible to parameterize and train all the components of the model, including pre- and post-processing functions.

Many of the proposals found in the literature fit within this framework with very small adaptation. We now describe the encoder and variational circuits in detail and explain their links to other well-known machine learning models.

[1] The number of repetitions required for the estimation of each term is determined by the desired precision as well as by the variance $\mathrm{Var}(M_k) = \langle M_k^2\rangle - \langle M_k\rangle^2$. In this Review we won't discuss estimation methods.

A. The encoder circuit $U_{\phi(\boldsymbol{x})}$

There are several ways to encode data into qubits and each one provides different expressive power. This choice of encoding is related to kernel methods, a well-established field whose goal is to embed data into a higher dimensional feature space where a specific problem may be easier to solve. For example, non-linear feature maps change the relative position between data points such that a dataset may become easier to classify in the feature space. In a similar way, the process of encoding classical data into a quantum state can be interpreted as a feature map $\boldsymbol{x} \to U_{\phi(\boldsymbol{x})}|0\rangle^{\otimes n}$ to the high-dimensional vector space of the states of n qubits. Here, ϕ is a user-defined pre-processing function which transforms the data vector into circuit parameters.

The inner product of two data points in this space defines a similarity function, or kernel, $k(\boldsymbol{x},\boldsymbol{x}') = \left|\langle 0|U^\dagger_{\phi(\boldsymbol{x}')}U_{\phi(\boldsymbol{x})}|0\rangle\right|^2$. This quantity can be evaluated using the SWAP test shown in Fig. 3, and readily used in kernel-based models such as the support vector machine, the Gaussian process, and the principal component analysis.

Let us now discuss some examples. Stoudenmire and Schwab [15] encode data as products of local kernels, one for each component of the input vector, which results in a product quantum state (i.e., disentangled). This approach is often referred to as *qubit encoding* and can produce highly non-linear kernels. As an example, for input vectors $\boldsymbol{x} \in [0,1]^n$ one can realize the feature map $\boldsymbol{x} \to \otimes_{i=1}^n \begin{pmatrix}\cos(x_i\pi/2)\\ \sin(x_i\pi/2)\end{pmatrix}$ by applying suitable single-qubit rotations. Mitarai *et al.* [16] use a similar approach, but encode each component of the data vector into multiple qubits. This redundancy populates the wave function with higher-order terms that can be exploited to fit non-linear functions of the data. Vidal and Theis [17] investigate how this redundancy helps the task of data fitting. They found lower bounds of the redundancy that are logarithmic in the complexity of the function to be fit, using a linear-algebraic complexity measure.

A different approach is taken by Wilson *et al.* [18]; the authors pre-process the input with a random linear map $\phi(\boldsymbol{x}) = A\boldsymbol{x} + \boldsymbol{b}$, creating a quantum version of the *random kitchen sink* [19]. They show that in the limit of many realizations of random linear maps, this approach implicitly implements a kernel. Interestingly, the form of the kernel depends on the layout of the encoder circuit, and not on all layouts are capable of implementing useful kernels. Another proposal that is based on random encoder circuits, but inspired by the convolutional filters used in neural networks, is the *quanvolutional* network by Henderson *et al.* [20].

The examples discussed so far require low-depth encoder circuits and may therefore be robust depending on the noise characteristics and level. A different approach is the *amplitude encoding*, a feature map encoding 2^n-dimensional data vectors into the wave function of merely

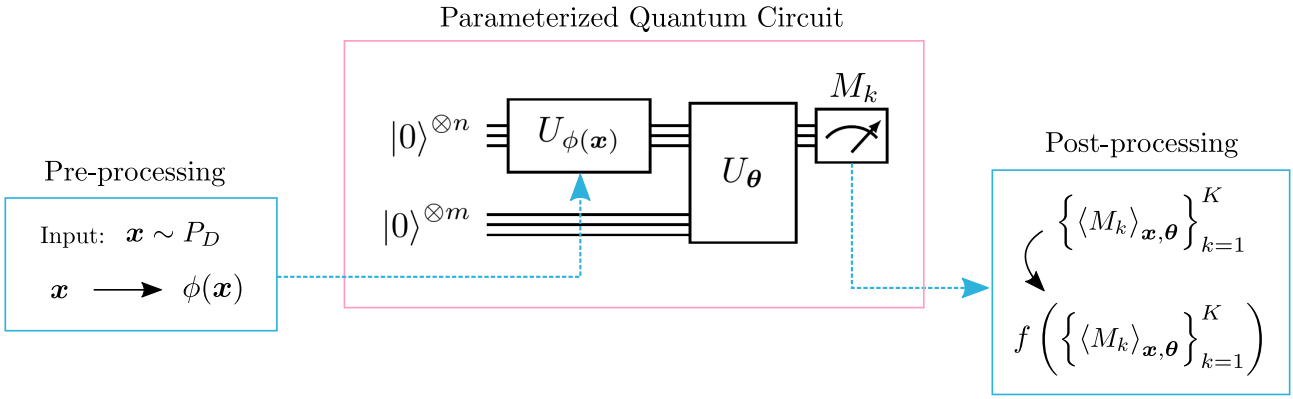

FIG. 2. A machine learning model comprised of classical pre/post-processing and parameterized quantum circuit. A data vector is sampled from the dataset distribution, $\boldsymbol{x} \sim P_D$. The pre-processing scheme maps it to the vector $\phi(\boldsymbol{x})$ that parameterizes the encoder circuit $U_{\phi(\boldsymbol{x})}$. A variational circuit $U_{\boldsymbol{\theta}}$, parameterized by a vector $\boldsymbol{\theta}$, acts on the state prepared by the encoder circuit and possibly on an additional register of ancilla qubits, producing the state $U_{\boldsymbol{\theta}}U_{\phi(\boldsymbol{x})}|0\rangle$. A set of observable quantities $\{\langle M_k\rangle_{\boldsymbol{x},\boldsymbol{\theta}}\}_{k=1}^{K}$ is estimated from the measurements. These estimates are then mapped to the output space through classical post-processing function f. For a supervised model, this output is the forecast associated to input $\boldsymbol{x}$. Generative models can be expressed in this framework with small adaptations.

n qubits. Assuming unit data vectors, the feature map $\boldsymbol{x} \to |\boldsymbol{x}\rangle$ provides an exponential advantage in terms of memory and leads to a linear kernel. It is also known that by preparing copies of this feature map one can implement arbitrary polynomial kernels [21]. Unfortunately, the depth of this encoder circuit is expected to scale exponentially with the number of qubits for generic inputs. Therefore, algorithms based on amplitude encoding could be impeded by our inability to coherently load data into quantum states.

On a different note, Havlíček *et al.* [22] argue that a feature map can be constructed so that the kernel is hard to estimate using classical resources, and that this is a form of quantum supremacy. They consider, for example, $U_{\phi(\boldsymbol{x})} = \exp(i\sum_{j,k}^{n}\phi_{j,k}(\boldsymbol{x})Z_jZ_k)H^{\otimes n}$ where Z_j is the Pauli-Z operator for the j-th qubit, $\phi_{j,k}$ are real functions, and H is the Hadamard gate. They conjecture that two layers of such an encoder circuit make the estimation of the kernel $k(\boldsymbol{x},\boldsymbol{x'}) = |\langle 0|U^{\dagger}_{\phi(\boldsymbol{x'})}U^{\dagger}_{\phi(\boldsymbol{x'})}U_{\phi(\boldsymbol{x})}U_{\phi(\boldsymbol{x})}|0\rangle|^2$ classically intractable. This is due to its similarity to the circuits used in the hidden shift problem of Boolean bent functions, which are known to be classically hard to simulate [23].

The design of feature maps inspired by quantum supremacy proposals is an interesting research direction. Whether this leads to an advantage in practical machine learning is an open question and should be tested empirically on existing computers. Ultimately, the form of the kernel and its parameters could be learned from data; this is a largely unexplored area in PQCs and has the potential to reduce the bias in kernel selection, and to automatically discover unknown feature maps that exhibit quantum supremacy.

B. The variational circuit $U_{\boldsymbol{\theta}}$

Similar to the universal approximation theorem in neural networks [24], there always exists a quantum circuit that can represent a target function within an arbitrary small error. The caveat is that such a circuit may be exponentially deep and therefore impractical. Lin *et al.* [25] argue that since real datasets arise from physical systems, they exhibit symmetry and locality; this suggests that it is possible to use 'cheap' models, rather than exponentially costly ones, and still obtain a satisfactory result. With this in mind, the variational circuit aims to implement a function that can approximate the task at hand while remaining scalable in the number of parameters and depth.

In practice, the circuit design follows a fixed structure of gates. Despite the dimension of the vector space growing exponentially with the number of qubits, the fixed structure reduces the model complexity, resulting in the number of free parameters to scale as a polynomial of the qubit count.

The first strategy to circuit design aims to comply with the fact that NISQ hardware has few qubits and usually operates on a sparse qubit-to-qubit connectivity graph with rather simple gates. *Hardware-efficient* circuits alternate layers of native entangling gates and single-qubit rotations [7]. Examples of these layers are shown in Fig. 4, where (a) and (b) are designed around the connectivity and gate set of superconducting and trapped ion computers, respectively. Heuristics can be used to strategically reduce the number of costly entangling gates. For example, Liu and Wang [26] use the Chow-Liu tree graph [27] to setup the entangling layers. First, the mutual information between all pairs of variables is estimated form the dataset. Then, entangling

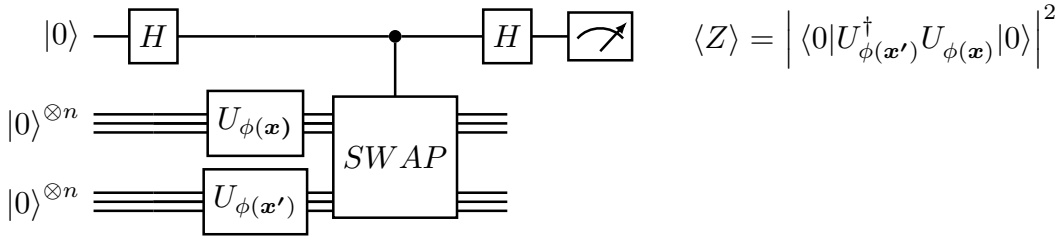

FIG. 3. The SWAP test can be used to estimate the implicit kernel implemented by an encoder circuit. Measurements of the Z Pauli observable on the ancilla qubit yield the absolute value squared of the inner product between $U_{\phi(\boldsymbol{x})}|0\rangle$ and $U_{\phi(\boldsymbol{x'})}|0\rangle$, respectively encoding data points $\boldsymbol{x}$ and $\boldsymbol{x}'$. The SWAP test finds several applications in machine learning and is a ubiquitous routine in quantum computing in general.

gates are placed between qubits so that most of the mutual information is represented.

Another principled approach to circuit design is inspired by quantum many-body physics. *Tensor networks* are methods to efficiently represent quantum states in terms of smaller interconnected tensors. In particular, these are often used to describe states whose entanglement is constrained by local interactions. By looking only at a smaller portion of the vector space, the computational cost is then reduced and becomes a polynomial function of the system size. This enables the numerical treatment of systems through layers of abstraction, reminiscent of deep neural networks. Indeed, some of the most studied tensor networks such as the matrix product state, the tree tensor network (TTN), and the multi-scale entanglement renormalization ansatz (MERA) have been tested for classification and generative modeling [28–30].

Figure 5 (a) shows an example of a TTN for supervised learning. After the application of each unitary, half of the qubits are traced out, while the other half continues to the next layer. Huggins *et al.* [29] suggest a *qubit-efficient* version where the traced qubits are reinitialized and used as the inputs of another unitary, as shown in Fig. 5 (b). Qubit-efficient schemes could significantly reduce the required number of qubits, a favorable condition to some NISQ hardware.

Neural networks and deep learning have proven to be very successful and therefore offer a further source of inspiration for circuit design. Both variational circuits and neural networks can be thought of as layers of connected computational units controlled by adjustable parameters. This has led some authors to refer to variational circuits as 'quantum neural networks'. Here we shall briefly discuss the key differences that make this approach to circuit design rather difficult.

First, quantum circuit operations are unitary and therefore linear; this is in contrast with the non-linear activation functions used in neural networks, which are key to their success and universality [33]. There are several ways to construct non-linear operations in quantum circuits, both coherently (i.e., exploiting entanglement) or non-coherently (e.g., exploiting the natural coupling of the system to the environment). These can in turn be used to implement classical artificial neurons in quantum circuits [34–36].

The second key difference is that it is impossible to access the quantum state at intermediate points during computation. Although measurement of ancillary quantum variables can be used to extract limited information, any attempt to observe the full state of the system would disrupt its quantum character. This implies that executing the variational circuit cannot be seen as performing the forward pass of a neural network. Moreover, it is difficult to conceive a circuit learning algorithm that truly resembles *backpropagation*, as it would rely on storing the intermediate state of the network during computation [37]. Backpropagation is the gold standard algorithm for neural networks and can be described as a computationally efficient organization of the chain rule that allows gradient descent to work on large-scale models.

The questions of how to generalize a quantum artificial neuron and design a quantum backpropagation algorithm have been open for quite some time [38]. Some recent work goes towards this direction. Verdon *et al.* [39] quantize the parameters of the variational circuit which are then prepared in superposition in a dedicated register. This enables a backpropagation-like algorithm which exploits quantum effects such as phase kickback and tunneling. Beer *et al.* [40] use separate qubit registers for input and output, and define the quantum neuron as a completely positive map between the two. The resulting network is universal for quantum computation and can be trained by an efficient process resembling backpropagation.

C. Circuit learning

Just like classical models, PQC models are trained to perform data-driven tasks. The task of learning an arbitrary function from data is mathematically expressed as the minimization of a loss function $L(\boldsymbol{\theta})$, also known as the objective function, with respect to the parameter vector $\boldsymbol{\theta}$. We discuss two types of algorithms, namely gradient-based and gradient-free, that can be applied to

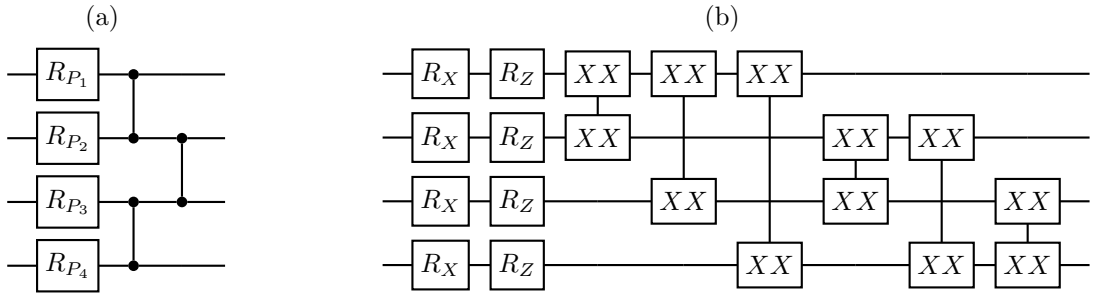

FIG. 4. Examples of hardware-efficient layers that can be used for encoder and variational circuits. Hardware-efficient constructions use entangling interactions that are naturally available on hardware and do not require compilation. Layers are repeated a number of times which is compatible with the hardware coherence time. (a) The construction in Ref. [31] uses single-qubit rotations $R_{P_j} = \exp\left(-\frac{i}{2}\theta_j P_j\right)$ about randomly sampled directions $P_j \in \{X, Y, Z\}$, and a ladder of control-Z entangling gates. Both the gate set and the connectivity are naturally implemented by many superconducting computers. (b) The construction in Ref. [32] uses single-qubit rotations about X and Y, and a fully-connected pattern of XX entangling gates. Both the gate set and the connectivity are naturally implemented by trapped ions computers.

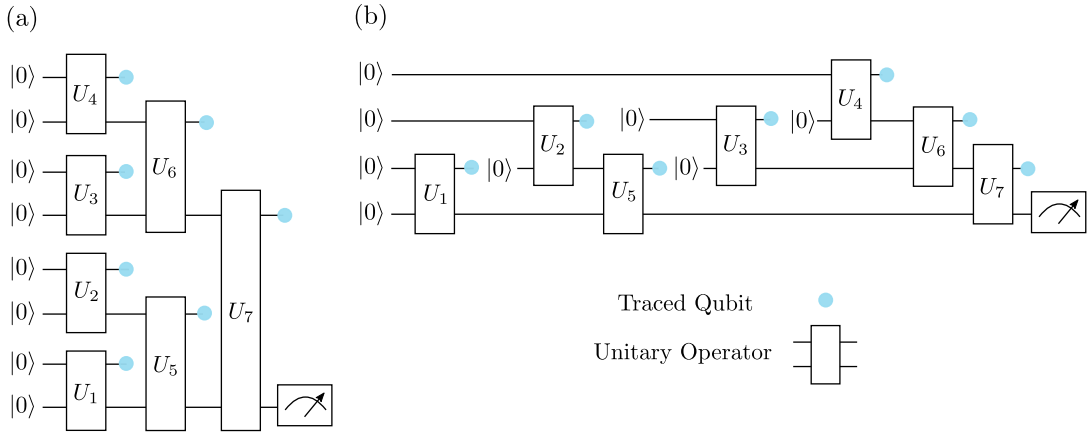

FIG. 5. Discriminative binary tree tensor network and its qubit-efficient version – adapted from Ref. [29]. (a) The binary TTN implements a coarse graining procedure by tracing over half of the qubits after the application of each unitary. (b) A qubit-efficient version re-initializes the discarded qubits to be used in parallel operations. This scheme implements the same operation in (a) but requires fewer qubits on the device. It may however result in a deeper circuit.

optimize the parameters of a variational circuit $U_{\boldsymbol{\theta}}$.

One instance of gradient-based algorithms is the iterative method called gradient descent. Here the parameters are updated towards the direction of steepest descent of the loss function

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} L, \tag{2}$$

where $\nabla_{\boldsymbol{\theta}} L$ is the gradient vector and η is the learning rate – a hyperparameter controlling the magnitude of the update. This procedure is iterated and, assuming suitable conditions, converges to a local minimum of the loss function.

The required partial derivatives can be calculated numerically using a finite difference scheme

$$\frac{\partial L}{\partial \theta_j} \approx \frac{L(\boldsymbol{\theta} + \Delta \boldsymbol{e}_j) - L(\boldsymbol{\theta} - \Delta \boldsymbol{e}_j)}{2\Delta}, \tag{3}$$

where Δ is a (small) hyperparameter and $\boldsymbol{e}_j$ is the Cartesian unit vector in the j direction. Note that in order to estimate the gradient vector $\nabla_{\boldsymbol{\theta}} L$, this approach evaluates the loss function twice for each parameter.

Alternatively, Spall's simultaneous perturbation stochastic approximation (SPSA) [41, 42] computes an approximate gradient vector with just two evaluations of the loss function as

$$\frac{\partial L}{\partial \theta_j} \approx \frac{L(\boldsymbol{\theta} + c\,\boldsymbol{\Delta}) - L(\boldsymbol{\theta} - c\,\boldsymbol{\Delta})}{2\,c\,\Delta_j}, \tag{4}$$

where $\boldsymbol{\Delta}$ is a random perturbation vector and c is a (small) hyperparameter.

There are cases when finite difference methods are ill-conditioned and unstable due to truncation and round-off errors. This is one of the reasons why machine learning relies on the analytical gradient when possible,

and it is often calculated with automatic differentiation schemes [43]. The analytical gradient can also be estimated for variational circuits, although the equations depend on the choice of parameterization for the gates. For our discussion, we consider circuits $U_{J:1} = U_J \cdots U_1$, where trainable gates are of the from $U_j = \exp\left(-\frac{i}{2}\theta_j P_j\right)$, and where $P_j \in \{I, Z, X, Y\}^{\otimes n}$ is a tensor product of n Pauli matrices. Arguably, this is the most common parameterization found in the literature.

Using this, Li *et al.* [44] propose a way to efficiently compute analytical gradients in the context of quantum optimal control. Mitarai *et al.* [16] bring this method into the context of supervised learning. Recall that the model's output is a function of expectation values $\langle M_k \rangle_{\boldsymbol{\theta}}$. Using the chain rule we can write the derivative $\frac{\partial L}{\partial \theta_j}$ as a function of the derivatives of the expectation values $\frac{\partial \langle M_k \rangle_{\boldsymbol{\theta}}}{\partial \theta_j}$. Each of these quantities can be estimated on quantum hardware using the *parameter shift rule*

$$\frac{\partial \langle M_k \rangle_{\boldsymbol{\theta}}}{\partial \theta_j} = \frac{\langle M_k \rangle_{\boldsymbol{\theta}+\frac{\pi}{2}\boldsymbol{e}_j} - \langle M_k \rangle_{\boldsymbol{\theta}-\frac{\pi}{2}\boldsymbol{e}_j}}{2}, \tag{5}$$

where subscripts $\boldsymbol{\theta} \pm \frac{\pi}{2}\boldsymbol{e}_j$ indicate the shifted parameter vector to use for the evaluation (see Schuld *et al.* [45] for a detailed derivation). Note that this estimation can be performed by executing two circuits.

An alternative method can estimate the partial derivative with a single circuit, but at the cost of adding an ancilla qubit. A simple derivation using the gate parameterization introduced above (e.g., see Farhi and Neven [46]) shows that the partial derivative can be written as

$$\frac{\partial \langle M_k \rangle_{\boldsymbol{\theta}}}{\partial \theta_j} = \mathrm{Im}\left(\mathrm{tr}\left(M_k U_{J:j+1} P_j U_{j:1} |0\rangle\langle 0| U_{J:1}^\dagger\right)\right). \tag{6}$$

This can be thought of as an *indirect measurement* and can be evaluated using the Hadamard test shown in Fig. 6. This method can be generalized to compute higher order derivatives, as presented for example by Dallaire-Demers and Killoran [47], and with alternative gate parameterizations, as done for example by Schuld *et al.* [48].

We shall note that despite the apparent simplicity of the circuit in Fig. 6, the actual implementation of Hadamard tests may be challenging due to non-trivial controlled gates. Coherence must be guaranteed in order for quantum interference to produce the desired result. Mitarai and Fujii [49] propose a method for replacing a class of indirect measurements with direct ones. Instead of an interference circuit one can execute, in some cases, multiple simpler circuits that are suitable for implementations on NISQ computers. The 'parameters shift rule' in Eq. (5) is nothing but the direct version of the measurement in Eq. (6).

Compared to finite difference and SPSA, the analytical gradient has the advantage of providing an unbiased estimator. Additionally, Harrow and Napp [50] find evidence that circuit learning using the analytical gradient outperforms any finite difference method. This is done by showing that for n qubits and precision ϵ, the query cost of an oracle for convex optimization in the vicinity of the optimum scales as $\mathcal{O}(\frac{n^2}{\epsilon})$ for the analytical gradient, whereas finite difference needs at least $\Omega(\frac{n^3}{\epsilon^2})$ calls to the oracle. In practice though, it is found that SPSA performs well in small-scale noisy experimental settings (e.g., see Kandala *et al.* [7] and Havlíček *et al.* [22]).

Particular attention should be given to the problems of exploding and vanishing gradients which are well-known to the machine learning community. Classical models, in particular recurrent neural networks, are often constrained to perform unitary operations so that their gradients cannot explode (see Wisdom *et al.* [51] for an example). Quantum circuits implementing unitary operations naturally avoid the exploding gradient problem. On the other hand, McClean *et al.* [31] show that random circuits of reasonable depth lead to an optimization landscape with exponentially large plateaus of vanishing gradients with an exponentially decaying variance. This can be understood as a consequence of Levy's lemma [52] which states that a random variable that depends on many independent variables is essentially constant. The learning algorithm is thus unable to estimate the gradient and may perform a random walk in parameter space. While this limits the effectiveness of variational circuits initialized at random, the use of highly structured circuits could alleviate the problem (e.g., see Grant *et al.* [53] for a structured initialization strategy).

We shall stress here that in hybrid systems parameter updates are performed classically. This implies that some of the most successful deep learning methods can be readily used for circuit learning. Indeed, heuristics such as stochastic gradient descent [54], resilient backpropagation [55], and adaptive momentum estimation [56], have already been applied with success. These were designed to deal with issues of practical importance such as large datasets, large noise in gradient estimates, and the need to find adaptive learning rates in Eq. (2). In practice, these choices can reduce the time for successful training from days to hours.

There are cases where gradient-based optimization may be challenging. For example, in a noisy experimental setting the loss function may be highly non-smooth and not suitable for gradient descent. As another example, the objective function may be itself unknown and therefore should be treated as a black-box. In these cases, circuit learning can be carried out by gradient-free methods. A well-known method of this type is particle swarm optimization [57]. Here the system is initialized with a number of random solutions called particles, each one moving through solution space with a certain velocity. The trajectory of each particle is adjusted according to its own experience and that of other particles so that they converge to a local minima. Another popular method is Bayesian optimization [58]. It uses evaluations of the objective function to construct a model of the function itself. Subsequent evaluations can be chosen either to

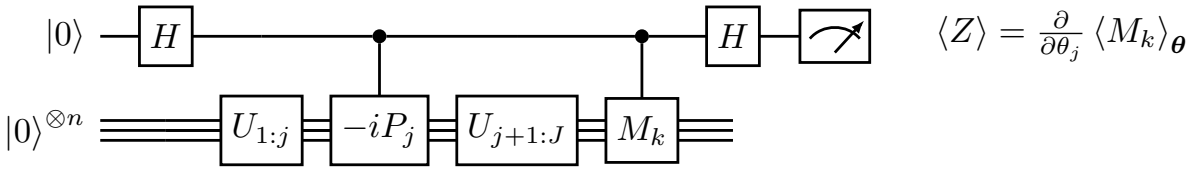

FIG. 6. The Hadamard test can be used to estimate the partial derivative of an expectation $\langle M_k\rangle_{\boldsymbol{\theta}}$ with respect to the parameter θ_j. Here we show a simple case where gates are of the form $U_j = \exp\left(-\frac{i}{2}\theta_j P_j\right)$ and where both P_j and M_k are tensor products of Pauli matrices. It can be shown that measurements of the Z Pauli observable on the ancilla qubit yield Eq. (6), the desired partial derivative. Hadamard tests can be designed to estimate higher order derivatives and to work with different measurements and gate parameterizations.

improve the model or to find a minima.

Zhu *et al.* [59] compare Bayesian and particle swarm optimization for training a generative model on a trapped ion quantum computer. While Bayesian optimization outperforms particle swarm in their setting, they found that the large number of parameters challenges both optimizers. They show that an ideal simulated system is not significantly faster than the experimental system, indicating that the actual bottleneck is the classical optimizer. Leyton-Ortega *et al.* [60] train a generative model on a superconducting quantum computer and compare the gradient-free methods of zeroth-order optimization package [61] and stochastic hill-climbing, with gradient descent. They find that on average zeroth-order optimization achieves the lowest loss on their hardware. They argue that the main optimization challenge is to overcome the variance of the loss function which is due to random parameter initialization, hardware noise, and finite number of measurements.

Genetic algorithms [62] are another large class of gradient-free optimization algorithms. At each step, candidate solutions are evolved using biology-inspired operations such as recombination, mutation, and natural selection. When used for circuit learning, genetic algorithms define a set of allowed gates and the maximum number to be employed. Lamata *et al.* [63] suggest the use of genetic algorithms to train a PQC model for compression using a universal set of single- and two-qubit gates. Ding *et al.* [64] validate the idea experimentally by deploying a pre-trained PQC model on a superconducting computer and find that using a subsequent genetic algorithm improves its fidelity.

To conclude, we note that optimization algorithms should be tailored for PQC models if we want to achieve better scalability. Very recent work has been approaching circuit learning from this perspective (e.g., see Ostaszewski *et al.* [65] and Nakanishi *et al.* [66]).

III. APPLICATIONS

In this Section we look at machine learning applications using PQC models where the goal is to obtain an advantage over classical models. For supervised learning with classical data we give a general overview of how PQC model can be applied to classification and regression. For unsupervised learning with classical data we focus on generative modeling since this comprises most of the literature.

PQC models can also handle inputs and outputs that are inherently quantum mechanical, i.e., already in superposition. These are often referred to as *quantum data* [67]. Quantum input data could originate remotely, for example, from other quantum computers transmitting over a quantum Internet [68]. Otherwise, if a preparation recipe is available, one could prepare the input data locally using a suitable encoder circuit. Assuming this data preparation is efficient, one can extend supervised and unsupervised learning to quantum states and quantum information.

Figure 7 shows examples for all these cases. Intuitively each application is a specification of the components outlined in Fig. 2, which the Reader is encouraged to refer to throughout the Section for clarity.

In many practical decision-making scenarios there is no available data concerning the best course of action. In this case, the model needs to interact with its environment to obtain information and learn how to perform a task from its own experience. This is known as reinforcement learning. An example would be a video game character that learns a successful strategy by repeatedly playing the game, analyzing results, and improving. Although quantum generalizations and algorithms for reinforcement learning have been proposed, to the best of our knowledge, none of them are based on hybrid systems and PQC models.

A. Supervised learning

Let us first consider supervised learning tasks, e.g., classification and regression, on classical data. Given a dataset $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^{N}$, of N samples, the goal is to learn a model function $f : \mathcal{X} \to \mathcal{Y}$ that maps each $\boldsymbol{x} \in \mathcal{X}$ to its corresponding target $\boldsymbol{y} \in \mathcal{Y}$. A standard approach

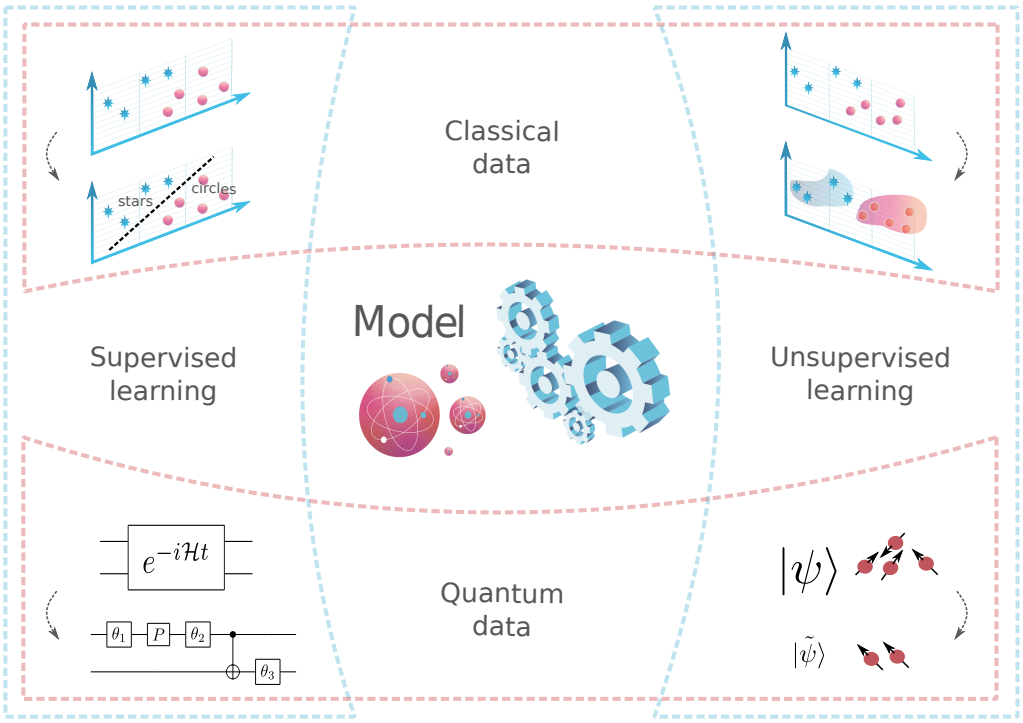

FIG. 7. Parameterized quantum circuit models can be trained for a variety of machine learning tasks, such as supervised and unsupervised learning, on both classical and quantum data. This figure shows examples from each category. In the top-left panel, the model learns to recognize patterns to classify the classical data. In the top-right panel, the model learns the probability distribution of the training data and can generate new synthetic data accordingly. For supervised learning of quantum data, bottom-left panel, the model assists the compilation of a high-level algorithm to low-level gates. Finally, for unsupervised learning of quantum data, bottom-right panel, the model performs lossy compression of a quantum state.

is to minimize a suitable regularized loss function, that is,

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \frac{1}{N}\sum_{i=1}^{N} L\left(f(\boldsymbol{x}^{(i)},\boldsymbol{\theta}),\boldsymbol{y}^{(i)}\right) + R(\boldsymbol{\theta}), \qquad (7)$$

where $\boldsymbol{\theta}$ is the set of parameters defining the model function, L quantifies the error of a forecast, and R is a regularization function penalizing undesired values for the parameters. The latter is used to prevent overfitting; indeed, if the training set is not sufficiently large, the model could simply memorize the training data and not generalize to unseen data.

In the PQC framework, once the encoder circuit $U_{\phi(\boldsymbol{x})}$ is set up, there are two main options for the remaining part of the circuit: the quantum kernel estimator (QKE), and the variational quantum model (VQM). We now briefly discuss both, and refer the Reader to Schuld and Killoran [69] for a more in-depth theoretical exposition.

The QKE does not use a variational circuit $U_{\boldsymbol{\theta}}$ to process the data; instead, it uses the SWAP test (e.g., see Fig. 3) to evaluate the possibly intractable kernel $k(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)})$. Then, resorting to the representer theorem [70], the model function is expressed as an expansion over kernel functions $f(\boldsymbol{x},\boldsymbol{w}) = \sum_{i=1}^{N} w_i k(\boldsymbol{x},\boldsymbol{x}^{(i)})$. The learning task is to find parameters $\boldsymbol{w}$ so that the model outputs correct forecasts. Note that these parameters define the classical post-processing function, as opposed to an operation of the PQC. A potential caveat is that QKE relies on a coherent SWAP test which may be non-trivial to implement on NISQ computers.

The VQM, on the other hand, uses a variational circuit $U_{\boldsymbol{\theta}}$ to process data directly in the feature space. A set of expectation values $\{\langle M_k\rangle_{\boldsymbol{x},\boldsymbol{\theta}}\}_{k=1}^{K}$ is estimated and post-processed to obtain the model output (see Fig. 2). In contrast to QKE, VQM parameters define the operations carried out by the quantum computer and require a circuit learning algorithm of the kind discussed in Section II C.

Havlíček *et al.* [22] experimentally demonstrate QKE and VQM classifiers on two superconducting qubits of the IBM Q5 Yorktown. Their QKE estimates a classically intractable feature map (see Section II A for details) which is then fed into a support vector machine to find the separating hyper-plane. Their VQM uses a hardware-efficient circuit instead. By employing a suitable error mitigation protocol, they find an increase in classification success with increasing circuit depth. In the future, it would be interesting to systematically compare these proposals

against established classical models by evaluating accuracy and training efficiency, for example.

We now focus our discussion on VQM proposals. Farhi and Neven [46] propose a VQM binary classifier for bitstrings. The encoder circuit simply maps bitstrings to the computational basis states by applying identity and NOT gates at almost no cost. The variational circuit acts on the input register and one ancilla qubit which is measured to yield a class forecast. With n-bit data strings as the input, there are 2^{2^n} possible binary functions that could generate the class labels. The authors show that for any of the possible label functions there exists a variational circuit that achieves zero classification error. For some of these functions, the circuit is exponentially deep and therefore impractical. This result parallels the well known universal approximation theorem [24] which states that neural networks with an exponentially large hidden layer of non-linear neurons are able to represent any Boolean function.

Mitarai *et al.* [16] propose VQMs for classification and regression of real-valued data using a highly non-linear qubit encoding. The variational circuit must then entangle the qubits such that a local observable can extract the relevant non-linear features. As discussed in Section II B one possible way to strategically construct highly entangling variational circuits is inspired by tensor networks. Grant *et al.* [30] use TTN and MERA variational circuits to perform binary classification on canonical datasets such as Iris and MNIST. In their simulations, MERA always outperforms TTN. One of their simplest models is efficiently trained classically and then deployed on the IBM Q5 Tenerife quantum computer with significant resilience to noise.

Stoudenmire *et al.* [28] train a TTN to perform pairwise classification of the MNIST image data. In their simulations, they use entanglement entropy to quantify the amount of information in a detail of the image that is gained by observing the context. This is an example of how quantum properties can be used to characterize the complexity of classical data, which is a developing area of research.

Schuld *et al.* [48] propose a VQM classifier assuming amplitude-encoded input data. Since this encoder circuit may be very expensive, the authors aim to keep the variational circuit low-depth and highly expressive at the same time. This is achieved through a systematic use of entangling gates, and by keeping the number of parameters polynomial in the number of qubits. Their simulations on benchmark datasets show performance comparable to that of off-the-shelf classical models while using significantly fewer parameters.

To date, all supervised learning experiments involved scaled-down, often trivial, datasets due to the limitation of available quantum hardware, and demonstrations at a more realistic scale are desirable. As a last comment, we note that a largely undeveloped area is that of regularization techniques specifically designed for PQC models which is, in our opinion, an interesting area for future research.

B. Generative modeling

We now discuss generative modeling, an unsupervised learning task where the goal is to model an unknown probability distribution and generate synthetic data accordingly. Generative models have been successfully applied in computer vision, speech synthesis, inference of missing text, de-noising of images, chemical design, and many other automated tasks. It is believed that they will play a key role in the development of general artificial intelligence; a model that can generate realistic synthetic samples is likely to 'understand' its environment.

Concretely, the task is to learn a model distribution $q_{\boldsymbol{\theta}}$ that is close to a target distribution p. The closeness is defined in terms of a divergence D on the statistical manifold, and learning consists of minimizing this divergence; that is,

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} D(p, q_{\boldsymbol{\theta}}). \tag{8}$$

Since the target probability distribution is unknown, it is approximated using a dataset $\mathcal{D} = \{\boldsymbol{v}^{(i)}\}_{i=1}^N$ which we have access to and which is distributed according to the target distribution. As an example, $\boldsymbol{v}^{(i)}$ could be natural images extracted from the Internet.

The probabilistic nature of quantum mechanics suggests that a model distribution can be encoded in the wave function of a quantum system [71, 72]. Let us see how a simple adaptation of the model shown in Fig. 2 gives a generative model for n-dimensional binary data $\boldsymbol{v}^{(i)} \in \{0,1\}^n$. First, we set the encoder circuit to the identity $U_{\phi(\boldsymbol{x})} = I$ since in this problem there is no input data. Second, we apply a variational circuit $U_{\boldsymbol{\theta}}$ to the initial state $|0\rangle^{\otimes n}$. Finally, we perform a measurement in the computational basis, i.e., we measure the set of operators $\{\langle M_{\boldsymbol{v}}\rangle_{\boldsymbol{\theta}}\}_{\boldsymbol{v}}$ where $M_{\boldsymbol{v}} = |\boldsymbol{v}\rangle\langle\boldsymbol{v}|$ are projectors for the bitstrings. The resulting generative model, known as the quantum circuit Born machine (QCBM) [26, 32], implements the probability distribution

$$q_{\boldsymbol{\theta}}(\boldsymbol{v}) = \mathrm{tr}\Big(M_{\boldsymbol{v}} U_{\boldsymbol{\theta}} |0\rangle\langle 0| U_{\boldsymbol{\theta}}^\dagger\Big). \tag{9}$$

Since the target data is binary, no post-processing is needed and each measurement outcome $\boldsymbol{v} \sim q_{\boldsymbol{\theta}}$ is an operational output. If the target data were instead real-valued, we could interpret bitstrings as discretized outputs and use a post-processing function to recover real-values.

As one does not have access to the wave function, characterizing the distribution $q_{\boldsymbol{\theta}}$ may be intractable for all but the smallest circuits. For this reason, QCBMs belong to the class of *implicit models*, models where it is easy to obtain a sample $\boldsymbol{v} \sim q_{\boldsymbol{\theta}}$, but hard to estimate the likelihood $q_{\boldsymbol{\theta}}(\boldsymbol{v})$. Machine learning researchers have become increasingly interested in implicit models because

of their generality, expressive power, and success in practice [73]. Interestingly, Du *et al.* [74] show that QCBMs have strictly more expressive power than classical models such as deep Boltzmann machines, when only a polynomial number of parameters are allowed. Coyle *et al.* [75] show that some QCBMs cannot be efficiently simulated by classical means in the worst case, and that this holds for all the circuit families encountered during training.

Benedetti *et al.* [32] build low-depth QCBMs using variational circuits suitable for trapped ion computers (see Fig. 4 (b) for an example). They use particle swarms to minimize an approximation to the Kullback-Leibler divergence [78] $D(p, q_{\boldsymbol{\theta}}) = \sum_{\boldsymbol{v}} p(\boldsymbol{v}) \ln \frac{p(\boldsymbol{v})}{q_{\boldsymbol{\theta}}(\boldsymbol{v})}$. In their simulations they successfully train models for the canonical Bars-and-Stripes dataset and for Boltzmann distributions, and use them to design a performance indicator for hybrid quantum-classical systems. Zhu *et al.* [59] implement this scheme on four qubits of an actual trapped ion computer and experimentally demonstrate convergence of the model to the target distribution.

Liu and Wang [26] propose the use of gradient descent to minimize the maximum mean discrepancy [79] $D(p, q_{\boldsymbol{\theta}}) = \|\sum_{\boldsymbol{v}} p(\boldsymbol{v})\phi(\boldsymbol{v}) - \sum_{\boldsymbol{v}} q_{\boldsymbol{\theta}}(\boldsymbol{v})\phi(\boldsymbol{v})\|^2$, where ϕ is a classical feature map, and the expectations are estimated from samples. Their approach allows for gradient estimates with discrete target data, which is often not possible in classical implicit models. In their simulations they successfully train QCBMs for the Bars-and-Stripes dataset and for discretized Gaussian distributions. Hamilton *et al.* [80] implement this schema on the IBM Q20 Tokyo computer, and examine how statistical and hardware noise affect convergence. They find that the generative performance of state-of-the-art hardware is usually significantly worse than that of the numerical simulations. Leyton-Ortega *et al.* [60] perform a complementary experimental study on the Rigetti 16Q-Aspen computer. They argue that due to the many components involved in hybrid quantum-classical systems (e.g., choice for the entangling layers, optimizers, post-processing, etc.), the performance ultimately depends on the ability to correctly set hyperparameters; that is, research on automated hyperparameter setting will be key to the success of QCBMs.

Another challenge in QCBMs is the choice of a suitable loss functions. Non-differentiable loss functions are often hard to optimize; one can use gradient-free methods, but these are likely to struggle as the number of parameters becomes large. Differentiable loss functions are often hard to design; recall that since QCBM are implicit models, one does not have access to the likelihood $q_{\boldsymbol{\theta}}(\boldsymbol{v})$. *Adversarial* methods developed in deep learning can potentially overcome these limitations. Figure 8 (a) shows the intuition; the adversarial method introduces a discriminative model whose task is to distinguish between true data coming from the dataset and synthetic data coming from the generative model. This creates a 'game' where the two players, i.e., the models, compete. The advantage is that both models are trained at the same time, with the discriminator providing a differentiable loss function for the generator.

Lloyd and Weedbrook [81] put forward the quantum generative adversarial network (QGAN) and theoretically examine variants where target data, generator and discriminator are either classical or quantum. We discuss the case of quantum data in the next Section while here we focus on classical data. Both Situ *et al.* [82] and Zeng *et al.* [83] couple a PQC generator to a neural network discriminator and successfully reproduce the statistics of some discrete target distributions. Romero and Aspuru-Guzik [84] extend this to continuous target distributions using a suitable post-processing function. Zoufal *et al.* [76] propose a QGAN to approximately perform amplitude encoding. While the best known generic method has exponential complexity, their circuit uses a polynomial number of gates. If both the cost of training and the required precision are kept low, this method has the potential to facilitate algorithms that require amplitude encoding.

One key aspect of generative models is their ability to perform *inference*. That is, when some of the observable variables are 'clamped' to known values, one can infer the expectation value of all other variables by sampling from the conditional probability. For example, inpainting, the process of reconstructing lost portions of images and videos, can be done by inferring missing values from a suitable generative model. Low *et al.* [85] use Grover's algorithm to perform inference on quantum circuits and obtain a quadratic speedup over naïve methods, although the overall complexity remains exponential. Zeng *et al.* [83] propose to equip QCBMs with this method, although this requires amplitude amplification and estimation methods that may be beyond NISQ hardware capabilities. It is an open question how to perform inference on QCBMs in the near term.

C. Quantum learning tasks

We finally consider learning tasks that are inherently quantum mechanical. As discussed in the Introduction, early hybrid approaches [11, 12] were proposed to assist the implementation of quantum algorithms (e.g., Deutsch's, Grover's, and Shor's) from datasets of input-output pairs. *Quantum algorithm learning* has been recently rediscovered by the community.

Morales *et al.* [86] train PQC models for the diffusion and oracle operators in Grover's algorithm. Noting that Grover's algorithm is optimal up to a constant, the authors show that the approach can find new improved operators for the specific case of three and four qubits. Wan *et al.* [87] train a PQC model to solve the hidden subgroup problem studied by Simon [88]. In their simulations, they recover the original Simon's algorithm with equal performance. Anschuetz *et al.* [89] use known techniques to map integer factoring to an Ising Hamiltonian, then train a PQC model to find the ground state hence

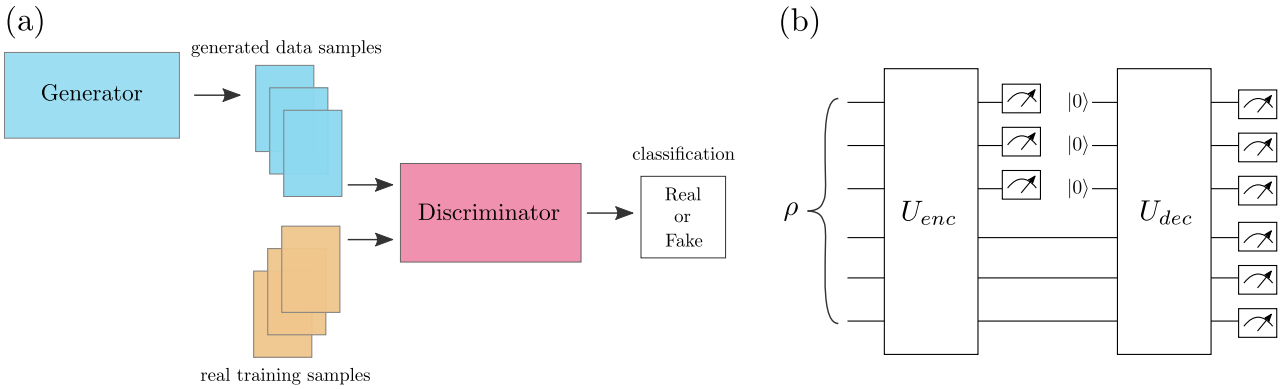

FIG. 8. Illustration of quantum generative models. (a) In the quantum generative adversarial network the generator creates synthetic samples and the discriminator tries to distinguish between the generated and the real samples. The network is trained until the generated samples are indistinguishable from the training samples. In this method the target data, the generator, and the discriminator can all be made quantum or classical. (b) The quantum autoencoder reduces the dimensionality of quantum data by applying an encoder circuit U_{enc}, tracing over a number of qubits and finally reconstructing the state with a decoder circuit U_{dec}. Panels (a) and (b) are adapted from Refs. [76] and [77], respectively.

finding the factors. Cincio *et al.* [90] train circuits to implement the SWAP test (see Fig. 3) and find solutions with a smaller number of gates than the known circuits.

These methods promise to assist the implementation of algorithms on near-term computers. Experimental studies will be needed to assess their scaling under realistic NISQ constraints and noise. Theoretical studies will be needed to understand their *sample complexity*, that is, the number of training samples required in order to successfully learn the target algorithm. Even in small-scale computers, we shall avoid exponential sampling complexity if we want these methods to be practical.

In the context of *quantum state classification*, Grant *et al.* [30] simulate the training of a TTN variational circuit for the classification of pure states that have different levels of entanglement. They found that, if the unitary operations in the TTN are too simple, classification accuracy on their synthetic dataset is no better than random class assignments. When using more complex operations involving ancilla qubits the TTN is able to classify quantum states with some accuracy. Chen *et al.* [91] simulate the training of PQC models to classify quantum states as pure or mixed, including a third possible output associated with an inconclusive result. Their circuits rely on layers of gates that are conditioned on measurement outcomes, with the purpose of introducing non-linear behaviour similar to that of neural networks.

State tomography is another ubiquitous task aiming at predicting the outcome probabilities of any measurement performed on an unknown state. To completely model the unknown state, one would require a number of measurements growing exponentially with the number of qubits. However, this can be formulated as a *quantum state learning* problem with the hope of minimizing the number of required measurements. Aaronson [92] studies the sampling complexity of this problem under Valiant's probably approximately correct (PAC) learning model [93]. They find that for practical purposes one needs a number of measurements scaling linearly with the number of qubits. Rocchetto *et al.* [94] experimentally verify the linear scaling on a custom photonic computer and extrapolate the value of the scaling constant. In terms of methodology, Lee *et al.* [95] propose to train a variational circuit $U_{\boldsymbol{\theta}}$ that transforms the unknown state $|\psi\rangle$ to a known fiducial state $|f\rangle$. The unknown state can be reproduced by evaluating the adjoint circuit on the fiducial state, that is, $|\psi\rangle \approx U_{\boldsymbol{\theta}}^{\dagger}|f\rangle$. A related learning tasks is that of *quantum state diagonalization* for mixed states. LaRose *et al.* [96] propose to train a variational circuit $U_{\boldsymbol{\theta}}$ such that the density matrix $\tilde{\rho} = U_{\boldsymbol{\theta}}\rho U_{\boldsymbol{\theta}}^{\dagger}$ is diagonalized, hence representing a classical probability distribution.

In the previous Section and in Fig. 8 (a) we introduced QGANs for classical data. We now discuss the case where all components are quantum mechanical, hence enabling the generative modeling of quantum data. The discriminator, now taking target and synthetic quantum states in input, aims at modeling the measurement for optimal distinguishability, also known as the Helstrom measurement [97]. In turn, the generator tries to make the task of distinguishing more difficult by minimizing its distance from the target state [81, 98]. In practice, this game can be implemented by coupling two PQC models and optimizing them in tandem. For example, Dallaire-Demers and Killoran [47] propose a QGAN that generates states conditioned on labels. This may find application in chemistry where the label is 'clamped' to a desired physical property and the model generates new molecular states accordingly. Benedetti *et al.* [98] propose a QGAN that generates approximations of pure states. They numerically show how the depths of generator and discriminator impact the quality of approximation. They also design a heuristic for stopping training, which is a non-trivial problem even in classical adversarial methods. Hu *et*

al. [99] experimentally demonstrate adversarial learning on a custom superconducting qubit.

Finally, PQC models can be used to attack well-known problems in quantum information from a novel machine learning perspective. Let us see some examples within the context of compression, error correction and compilation.

Romero *et al.* [77] propose a quantum autoencoder (QAE) to reduce the amount of resources needed to store quantum data. As shown in Fig. 8 (b) an encoder circuit U_{enc} is applied to the quantum data stored in n qubits. After tracing out $n-k$ qubits, a decoder circuit U_{dec} is used to reconstruct the initial state. The circuits are trained to maximize the expected fidelity between inputs and outputs, effectively performing a lossy compression of an n-qubit state into a k-qubit state.

Fault-tolerant quantum computers require error correction schemes that can deal with noisy and faulty operations. Leading proposals such as the color code and the surface code devote a large number of physical qubits to implement error-corrected logical qubits (see Gottesman [100] for an introduction to quantum error correction). Johnson *et al.* [101] suggest that a reduced overhead could be achieved in NISQ devices by training encoding and recovery circuits to optimize the average code fidelity.

The implementation of a quantum algorithm is also limited by the available gate set and qubit-to-qubit connectivity of the underlying hardware. This is where quantum compilers come into play, by abstracting the user from the low-level details. Khatari *et al.* [102] propose to train a hardware-efficient variational circuit $U_{\boldsymbol{\theta}}$ to approximately execute the same action as a target unitary $U = e^{-i\mathcal{H}t}$.

Reference	Task	Model	Learning	Qubits	Computer
Schuld *et al.* [103]	Classification	QKE	N/A	4	IBM Q5 Yorktown (S)
Grant *et al.* [30]	Classification	VQM	N/A	4	IBM Q5 Tenerife (S)
Havlíček *et al.* [22]	Classification	QKE, VQM	Gradient-based	2	IBM Q5 Yorktown (S)
Tacchino *et al.* [36]	Classification	Perceptron	Gradient-based	3	IBM Q5 Tenerife (S)
Benedetti *et al.* [32]	Generative	QCBM	N/A	4	Custom (T)
Hamilton *et al.* [80]	Generative	QCBM	Gradient-based	4	IBM Q20 Tokyo (S)
Zhu *et al.* [59]	Generative	QCBM	Gradient-free	4	Custom (T)
Leyton-Ortega *et al.* [60]	Generative	QCBM	Gradient-based, gradient-free	4	Rigetti 16Q-Aspen (S)
Coyle *et al.* [75]	Generative	QCBM	Gradient-based	4	Rigetti 16Q-Aspen (S)
Hu *et al.* [99]	State learning	QGAN	Gradient-based	1	Custom (S)
Zoufal *et al.* [76]	State learning	QGAN	Gradient-based	3	IBM Q20 Poughkeepsie (S)
Rocchetto *et al.* [94]	State learning	PAC	N/A	6	Custom (P)
Otterbach *et al.* [10]	Clustering	QAOA	Gradient-free	19	Rigetti 19Q-Acorn (S)
Ding *et al.* [64]	Compression	QAE	Gradient-free	3	Rigetti 8Q-Agave (S)
Ristè *et al.* [104]	Learning parity with noise	Oracle	N/A	5	IBM Q5 Yorktown (S)

TABLE II. Overview of parameterized quantum circuit models that have been demonstrated experimentally on superconducting (S), trapped ion (T), and photonic (P) hardware. N/A labels the cases where a learning algorithm was either not required or not used, e.g., when learning is simulated classically and the model is deployed on quantum hardware.

Reference	Name	Developer	PQC models	Language	Backend
Aleksandrowicz *et al.* [105]	Qiskit Aqua	IBM Research	VQE, QAOA, VQM, QKE	Python	Superconducting, Simulator
Bergholm *et al.* [106]	Pennylane	Xanadu	VQE, VQM, QGAN	Python	Superconducting, Simulator
Luo *et al.* [107]	Yao	QuantumBFS	VQE, QAOA, QCBM, QGAN	Julia	Simulator

TABLE III. Open-source software for developing machine learning models based on parameterized quantum circuits and, in some cases, for experimenting on existing quantum computers.

IV. OUTLOOK

In this Review we discussed parameterized quantum circuits (PQCs), a novel framework at the intersection of quantum computing and machine learning. This approach has not been restricted to theory and simulation but involved a series of experimental demonstrations on scaled-down problems being performed in the past two years. In Table II we summarize the relevant demonstra-

tions, and the Reader interested in experimental setups is invited to delve into the references therein.

The software development has also been moving at a fast pace (see Fingerhuth *et al.* [108] for a Review of general quantum computing software). There now exist several platforms for hybrid quantum-classical computation which are specifically dedicated to machine learning and provide PQC models, automatic differentiation techniques, and interfaces to both simulators and existing quantum computers. We shall stress here the importance of open-source software and the key role of numerical analysis. While traditional quantum algorithms have been subject to much analytical study of their performance, algorithms for PQC models often relies on heavy numerical study. This is due to the large number of components of the hybrid system, each one affecting the overall performance in a complex way. Open-source software enables experimentation at a much higher rate than previously possible, a scenario reminiscent of the deep learning developments a decade ago. It is therefore recommended to use available libraries when possible, enabling comparison of algorithms on an equal footing and to facilitate the replicability of the results. We summarize the relevant open-source software in Table III, without claiming to be comprehensive.

Researchers have also begun to explore connections between quantum supremacy proposals and quantum algorithms for optimization [109], getting us closer to practical utility if some key requirements can be met [110–112]. It is natural to explore similar connections between quantum supremacy and machine learning [75, 113].

We have seen that PQCs can implement classically intractable feature maps and kernel functions. Further studies will be needed to assess whether these can improve the performance of established kernel-based models such as the support vector machine, the Gaussian process and the principal component analysis. We also know that sampling from the probability distribution generated by instantaneous quantum polynomial-time circuits is classically intractable in the average case. A natural application for them is in generative modeling where the task itself requires sampling from complex probability distributions. But does classical intractability of these circuits imply an advantage in practice? One possible pitfall is that as the circuits become more expressive, the optimization landscape might also become harder to explore. As previously mentioned, demonstrations on real-world datasets of meaningful scale could answer these questions and should therefore be prioritised.

PQC models can also help in the study of quantum mechanical systems. For systems that exhibit quantum supremacy, a classical model cannot learn to reproduce the statistics unless it uses exponentially scaling resources. Provided that we can efficiently load or prepare quantum data in a qubit register, PQC models will deliver a clear advantage over classical methods for quantum learning tasks.

From the machine learning practitioner's point of view, there are several desirable properties that are naturally captured by PQC models. For example, recurrent neural networks may suffer from the exploding gradient problem. This can be prevented by constraining the operations to be unitary and much work has been done to efficiently parameterize the unitary group [114, 115]. PQC models have the advantage of naturally implementing unitary operations on an exponentially large vector space. As another example, state-of-the-art classical generative models may not allow gradient-based training when the data is discrete [73]. In PQC models discrete data arises from measurements on the qubits and, as we have seen, this does not preclude the computation of gradients. We believe that this is only the "tip of the iceberg" and that there are a number of research opportunities in this field. Largely unexplored aspects of PQC models include Vapnik-Chervonenkis dimensions, regularization techniques, Bayesian inference, and applications to reinforcement learning.

Finally, hybrid systems based on PQCs provide a framework for the incremental development of algorithms. In the near term, hybrid algorithms will rely heavily on classical resources. As quantum hardware improves, classical resources shall gradually be replaced by quantum resources and generic methods. For example, Wang *et al.* [116] propose a method that interpolates between the near-term variational quantum eigensolver and the long-term quantum phase estimation. Similarly, destructive SWAP and Hadamard tests [49, 117] could be gradually replaced by non-destructive variants. Hardware-efficient circuits shall be replaced by new parameterizations driven by the theory of tensor networks. Quantum compilers [118, 119] will enable the implementation of these higher level constructions on existing devices.

In passing, we envisage that a closer integration between the quantum and the classical components is desirable. This will entail a new generation of hardware facilities, such as hybrid data centers, the improvement of the software interfaces for cloud access to these computational resources, and the development of software frameworks that are *native* of hybrid systems. We believe that the accomplishment of these goals will firstly, facilitate the general research efforts, secondly, it will enable more extensive demonstrations of hybrid algorithms' potential on real-world application, and ultimately pave the way for the implementation in production environments.

The ideas and examples presented in this Review show the remarkable flexibility of the hybrid framework and its potential to use existing quantum hardware to its full extent. If PQC models can be shown to scale well to realistic machine learning tasks, they may become an integral part of automated forecasting and decision-making systems.

V. ACKNOWLEDGEMENTS

The authors would like to thank Tiya-Renee Jones for her help with Figures 1 and 7, Ilyas Khan for his support, and Miles Stoudenmire and Leonard Wossnig for useful feedback on an early version of this manuscript. M.B. is supported by the UK Engineering and Physical Sciences Research Council (EPSRC).

[1] John Preskill, "Quantum computing in the NISQ era and beyond," Quantum **2**, 79 (2018).
[2] Masoud Mohseni, Peter Read, Hartmut Neven, Sergio Boixo, Vasil Denchev, Ryan Babbush, Austin Fowler, Vadim Smelyanskiy, and John Martinis, "Commercialize quantum technologies in five years," Nature **543**, 171–174 (2017).
[3] AP Lund, Michael J Bremner, and TC Ralph, "Quantum sampling problems, bosonsampling and quantum supremacy," npj Quantum Information **3**, 15 (2017).
[4] Aram W Harrow and Ashley Montanaro, "Quantum computational supremacy," Nature **549**, 203 (2017).
[5] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L Obrien, "A variational eigenvalue solver on a photonic quantum processor," Nature communications **5**, 4213 (2014).
[6] PJJ O'Malley, Ryan Babbush, ID Kivlichan, Jonathan Romero, JR McClean, Rami Barends, Julian Kelly, Pedram Roushan, Andrew Tranter, Nan Ding, *et al.*, "Scalable quantum simulation of molecular energies," Physical Review X **6**, 031007 (2016).
[7] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M Chow, and Jay M Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," Nature **549**, 242 (2017).
[8] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann, "A quantum approximate optimization algorithm," arXiv preprint arXiv:1411.4028 (2014).
[9] Nikolaj Moll, Panagiotis Barkoutsos, Lev S Bishop, Jerry M Chow, Andrew Cross, Daniel J Egger, Stefan Filipp, Andreas Fuhrer, Jay M Gambetta, Marc Ganzhorn, Abhinav Kandala, Antonio Mezzacapo, Peter Mller, Walter Riess, Gian Salis, John Smolin, Ivano Tavernelli, and Kristan Temme, "Quantum optimization using variational algorithms on near-term quantum devices," Quantum Science and Technology **3**, 030503 (2018).
[10] JS Otterbach, R Manenti, N Alidoust, A Bestwick, M Block, B Bloom, S Caldwell, N Didier, E Schuyler Fried, S Hong, *et al.*, "Unsupervised machine learning on a hybrid quantum computer," arXiv preprint arXiv:1712.05771 (2017).
[11] Jeongho Bang, James Lim, MS Kim, and Jinhyoung Lee, "Quantum learning machine," arXiv preprint arXiv:0803.2976 (2008).
[12] Søren Gammelmark and Klaus Mølmer, "Quantum learning by measurement and feedback," New Journal of Physics **11**, 033017 (2009).
[13] Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre GR Day, Clint Richardson, Charles K Fisher, and David J Schwab, "A high-bias, low-variance introduction to machine learning for physicists," Physics Reports (2019), https://doi.org/10.1016/j.physrep.2019.03.001.
[14] Michael A. Nielsen and Isaac L. Chuang, "Quantum computation and quantum information: 10th anniversary edition," (2011).
[15] Edwin Stoudenmire and David J Schwab, "Supervised learning with tensor networks," in *Advances in Neural Information Processing Systems 29*, edited by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Curran Associates, Inc., 2016) pp. 4799–4807.
[16] Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa, and Keisuke Fujii, "Quantum circuit learning," Physical Review A **98**, 032309 (2018).
[17] Javier Gil Vidal and Dirk Oliver Theis, "Input redundancy for parameterized quantum circuits," arXiv preprint arXiv:1901.11434 (2019).
[18] CM Wilson, JS Otterbach, Nikolas Tezak, RS Smith, GE Crooks, and MP da Silva, "Quantum kitchen sinks: An algorithm for machine learning on near-term quantum computers," arXiv preprint arXiv:1806.08321 (2018).
[19] Ali Rahimi and Benjamin Recht, "Random features for large-scale kernel machines," in *Advances in neural information processing systems* (2008) pp. 1177–1184.
[20] Maxwell Henderson, Samriddhi Shakya, Shashindra Pradhan, and Tristan Cook, "Quanvolutional neural networks: Powering image recognition with quantum circuits," arXiv preprint arXiv:1904.04767 (2019).
[21] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd, "Quantum support vector machine for big data classification," Phys. Rev. Lett. **113**, 130503 (2014).
[22] Vojtěch Havlíček, Antonio D Córcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow, and Jay M Gambetta, "Supervised learning with quantum-enhanced feature spaces," Nature **567**, 209 (2019).
[23] Martin Rötteler, "Quantum algorithms for highly nonlinear boolean functions," in *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete algorithms* (Society for Industrial and Applied Mathematics, 2010) pp. 448–457.
[24] Kurt Hornik, Maxwell Stinchcombe, and Halbert White, "Multilayer feedforward networks are universal approximators," Neural networks **2**, 359–366 (1989).
[25] Henry W Lin, Max Tegmark, and David Rolnick, "Why does deep and cheap learning work so well?" Journal of Statistical Physics **168**, 1223–1247 (2017).
[26] Jin-Guo Liu and Lei Wang, "Differentiable learning of quantum circuit born machines," Phys. Rev. A **98**, 062324 (2018).
[27] C Chow and Cong Liu, "Approximating discrete probability distributions with dependence trees," IEEE transactions on Information Theory **14**, 462–467 (1968).
[28] Ding Liu, Shi-Ju Ran, Peter Wittek, Cheng Peng, Raul Blázquez García, Gang Su, and Maciej Lewenstein, "Machine learning by unitary tensor net-

work of hierarchical tree structure," arXiv preprint arXiv:1710.04833 (2017).
[29] William Huggins, Piyush Patil, Bradley Mitchell, K Birgitta Whaley, and E Miles Stoudenmire, "Towards quantum machine learning with tensor networks," Quantum Science and Technology **4**, 024001 (2019).
[30] Edward Grant, Marcello Benedetti, Shuxiang Cao, Andrew Hallam, Joshua Lockhart, Vid Stojevic, Andrew G. Green, and Simone Severini, "Hierarchical quantum classifiers," npj Quantum Information **4**, 65 (2018).
[31] Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush, and Hartmut Neven, "Barren plateaus in quantum neural network training landscapes," Nature communications **9**, 4812 (2018).
[32] Marcello Benedetti, Delfina Garcia-Pintos, Oscar Perdomo, Vicente Leyton-Ortega, Yunseong Nam, and Alejandro Perdomo-Ortiz, "A generative modeling approach for benchmarking and training shallow quantum circuits," npj Quantum Information **5**, 45 (2019).
[33] Kurt Hornik, "Approximation capabilities of multilayer feedforward networks," Neural networks **4**, 251–257 (1991).
[34] Yudong Cao, Gian Giacomo Guerreschi, and Alán Aspuru-Guzik, "Quantum neuron: an elementary building block for machine learning on quantum computers," arXiv preprint arXiv:1711.11240 (2017).
[35] E. Torrontegui and J. J. García-Ripoll, "Unitary quantum perceptron as efficient universal approximator," EPL (Europhysics Letters) **125**, 30004 (2019).
[36] Dario Gerace Francesco Tacchino, Chiara Macchiavello and Daniele Bajoni, "An artificial neuron implemented on an actual quantum processor," npj Quantum Information **5**, 26 (2019).
[37] Geoffrey E Hinton, DE Rumelhart, and Ronald J Williams, "Learning representations by back-propagating errors," Nature **323**, 533–536 (1986).
[38] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione, "The quest for a quantum neural network," Quantum Information Processing **13**, 2567–2586 (2014).
[39] Guillaume Verdon, Jason Pye, and Michael Broughton, "A universal training algorithm for quantum deep learning," arXiv preprint arXiv:1806.09729 (2018).
[40] Kerstin Beer, Dmytro Bondarenko, Terry Farrelly, Tobias J Osborne, Robert Salzmann, and Ramona Wolf, "Efficient learning for deep quantum neural networks," arXiv preprint arXiv:1902.10445 (2019).
[41] James C Spall, "A one-measurement form of simultaneous perturbation stochastic approximation," Automatica **33**, 109–112 (1997).
[42] James C Spall, "Adaptive stochastic approximation by the simultaneous perturbation method," IEEE transactions on automatic control **45**, 1839–1853 (2000).
[43] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind, "Automatic differentiation in machine learning: a survey," Journal of Marchine Learning Research **18**, 1–43 (2018).
[44] Jun Li, Xiaodong Yang, Xinhua Peng, and Chang-Pu Sun, "Hybrid quantum-classical approach to quantum optimal control," Physical review letters **118**, 150503 (2017).
[45] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran, "Evaluating analytic gradients on quantum hardware," arXiv preprint arXiv:1811.11184 (2018).
[46] Edward Farhi and Hartmut Neven, "Classification with quantum neural networks on near term processors," arXiv:1802.06002 (2018).
[47] Pierre-Luc Dallaire-Demers and Nathan Killoran, "Quantum generative adversarial networks," Physical Review A **98**, 012324 (2018).
[48] Maria Schuld, Alex Bocharov, Krysta Svore, and Nathan Wiebe, "Circuit-centric quantum classifiers," arXiv preprint arXiv:1804.00633 (2018).
[49] Kosuke Mitarai and Keisuke Fujii, "Methodology for replacing indirect measurements with direct measurements," arXiv preprint arXiv:1901.00015 (2018).
[50] Aram Harrow and John Napp, "Low-depth gradient measurements can improve convergence in variational hybrid quantum-classical algorithms," arXiv preprint arXiv:1901.05374 (2019).
[51] Scott Wisdom, Thomas Powers, John R. Hershey, Jonathan Le Roux, and Les Atlas, "Full-capacity unitary recurrent neural networks," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16 (Curran Associates Inc., USA, 2016) pp. 4887–4895.
[52] M. Ledoux, *The Concentration of Measure Phenomenon*, Mathematical surveys and monographs (American Mathematical Society, 2001).
[53] Edward Grant, Leonard Wossnig, Mateusz Ostaszewski, and Marcello Benedetti, "An initialization strategy for addressing barren plateaus in parametrized quantum circuits," arXiv preprint arXiv:1903.05076 (2019).
[54] Herbert Robbins and Sutton Monro, "A stochastic approximation method," The annals of mathematical statistics , 400–407 (1951).
[55] Martin Riedmiller and Heinrich Braun, "A direct adaptive method for faster backpropagation learning: The rprop algorithm," in *IEEE International Conference on Neural Networks* (1993) pp. 586–591 vol.1.
[56] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980 (2014).
[57] Russell C Eberhart and Xiaohui Hu, "Human tremor analysis using particle swarm optimization," in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, Vol. 3 (IEEE, 1999) pp. 1927–1930.
[58] Peter I Frazier, "A tutorial on bayesian optimization," arXiv preprint arXiv:1807.02811 (2018).
[59] D Zhu, NM Linke, M Benedetti, KA Landsman, NH Nguyen, CH Alderete, A Perdomo-Ortiz, N Korda, A Garfoot, C Brecque, *et al.*, "Training of quantum circuits on a hybrid quantum computer," arXiv preprint arXiv:1812.08862 (2018).
[60] Vicente Leyton-Ortega, Alejandro Perdomo-Ortiz, and Oscar Perdomo, "Robust implementation of generative modeling with parametrized quantum circuits," arXiv preprint arXiv:1901.08047 (2019).
[61] Yu-Ren Liu, Yi-Qi Hu, Hong Qian, Yang Yu, and Chao Qian, "Zoopt: Toolbox for derivative-free optimization," arXiv preprint arXiv:1801.00329 (2017).
[62] Kumara Sastry, David Goldberg, and Graham Kendall, "Genetic algorithms," in *Search methodologies* (Springer, 2005) pp. 97–125.
[63] Lucas Lamata, Unai Alvarez-Rodriguez, José David Martín-Guerrero, Mikel Sanz, and Enrique Solano,

"Quantum autoencoders via quantum adders with genetic algorithms," Quantum Science and Technology **4**, 014007 (2018).
[64] Yongcheng Ding, Lucas Lamata, Mikel Sanz, Xi Chen, and Enrique Solano, "Experimental implementation of a quantum autoencoder via quantum adders," Advanced Quantum Technologies , 1800065 (2019).
[65] Mateusz Ostaszewski, Edward Grant, and Marcello Benedetti, "Quantum circuit structure learning," arXiv preprint arXiv:1905.09692 (2019).
[66] Ken M Nakanishi, Keisuke Fujii, and Synge Todo, "Sequential minimal optimization for quantum-classical hybrid algorithms," arXiv preprint arXiv:1903.12166 (2019).
[67] Esma Aïmeur, Gilles Brassard, and Sébastien Gambs, "Machine learning in a quantum world," in *Conference of the Canadian Society for Computational Studies of Intelligence* (Springer, 2006) pp. 431–442.
[68] H Jeff Kimble, "The quantum internet," Nature **453**, 1023 (2008).
[69] Maria Schuld and Nathan Killoran, "Quantum machine learning in feature hilbert spaces," Phys. Rev. Lett. **122**, 040504 (2019).
[70] Bernhard Schölkopf, Ralf Herbrich, and Alex J. Smola, "A generalized representer theorem," in *Computational Learning Theory*, edited by David Helmbold and Bob Williamson (Springer Berlin Heidelberg, Berlin, Heidelberg, 2001) pp. 416–426.
[71] Song Cheng, Jing Chen, and Lei Wang, "Information perspective to probabilistic modeling: Boltzmann machines versus born machines," Entropy **20**, 583 (2018).
[72] Zhao-Yu Han, Jun Wang, Heng Fan, Lei Wang, and Pan Zhang, "Unsupervised generative modeling using matrix product states," Physical Review X **8** (2018), 10.1103/physrevx.8.031012.
[73] Ian Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," arXiv preprint arXiv:1701.00160 (2016).
[74] Yuxuan Du, Min-Hsiu Hsieh, Tongliang Liu, and Dacheng Tao, "The expressive power of parameterized quantum circuits," arXiv preprint arXiv:1810.11922 (2018).
[75] Brian Coyle, Daniel Mills, Vincent Danos, and Elham Kashefi, "The born supremacy: Quantum advantage and training of an ising born machine," arXiv preprint arXiv:1904.02214 (2019).
[76] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner, "Quantum generative adversarial networks for learning and loading random distributions," arXiv preprint arXiv:1904.00043 (2019).
[77] Jonathan Romero, Jonathan P Olson, and Alan Aspuru-Guzik, "Quantum autoencoders for efficient compression of quantum data," Quantum Science and Technology **2**, 045001 (2017).
[78] Solomon Kullback and Richard A Leibler, "On information and sufficiency," The annals of mathematical statistics **22**, 79–86 (1951).
[79] Arthur Gretton, Karsten M. Borgwardt, Malte Rasch, Bernhard Scholköpf, and Alexander J. Smola, "A kernel approach to comparing distributions," in *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 2*, AAAI'07 (AAAI Press, 2007) pp. 1637–1641.
[80] Kathleen E Hamilton, Eugene F Dumitrescu, and Raphael C Pooser, "Generative model benchmarks for superconducting qubits," arXiv preprint arXiv:1811.09905 (2018).
[81] Seth Lloyd and Christian Weedbrook, "Quantum generative adversarial learning," Physical review letters **121**, 040502 (2018).
[82] Haozhen Situ, Zhimin He, Lvzhou Li, and Shenggen Zheng, "Quantum generative adversarial network for generating discrete data," arXiv preprint arXiv:1807.01235 (2018).
[83] Jinfeng Zeng, Yufeng Wu, Jin-Guo Liu, Lei Wang, and Jiangping Hu, "Learning and inference on generative adversarial quantum circuits," Phys. Rev. A **99**, 052306 (2019).
[84] Jonathan Romero and Alan Aspuru-Guzik, "Variational quantum generators: Generative adversarial quantum machine learning for continuous distributions," arXiv preprint arXiv:1901.00848 (2019).
[85] Guang Hao Low, Theodore J Yoder, and Isaac L Chuang, "Quantum inference on bayesian networks," Physical Review A **89**, 062315 (2014).
[86] Mauro ES Morales, Timur Tlyachev, and Jacob Biamonte, "Variational learning of grover's quantum search algorithm," Physical Review A **98**, 062333 (2018).
[87] Kwok Ho Wan, Feiyang Liu, Oscar Dahlsten, and MS Kim, "Learning simon's quantum algorithm," arXiv preprint arXiv:1806.10448 (2018).
[88] Daniel R Simon, "On the power of quantum computation," SIAM journal on computing **26**, 1474–1483 (1997).
[89] Eric Anschuetz, Jonathan Olson, Alán Aspuru-Guzik, and Yudong Cao, "Variational quantum factoring," in *International Workshop on Quantum Technology and Optimization Problems* (Springer, 2019) pp. 74–85.
[90] Lukasz Cincio, Yiğit Subaşı, Andrew T Sornborger, and Patrick J Coles, "Learning the quantum algorithm for state overlap," New Journal of Physics **20**, 113022 (2018).
[91] Hongxiang Chen, Leonard Wossnig, Simone Severini, Hartmut Neven, and Masoud Mohseni, "Universal discriminative quantum neural networks," arXiv preprint arXiv:1805.08654 (2018).
[92] Scott Aaronson, "The learnability of quantum states," Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **463**, 3089–3114 (2007).
[93] Leslie G Valiant, "A theory of the learnable," in *Proceedings of the sixteenth annual ACM symposium on Theory of computing* (ACM, 1984) pp. 436–445.
[94] Andrea Rocchetto, Scott Aaronson, Simone Severini, Gonzalo Carvacho, Davide Poderini, Iris Agresti, Marco Bentivegna, and Fabio Sciarrino, "Experimental learning of quantum states," Science Advances **5**, eaau1946 (2019).
[95] Sang Min Lee, Jinhyoung Lee, and Jeongho Bang, "Learning unknown pure quantum states," Physical Review A **98**, 052302 (2018).
[96] Ryan LaRose, Arkin Tikku, Étude O'Neel-Judy, Lukasz Cincio, and Patrick J Coles, "Variational quantum state diagonalization," arXiv preprint arXiv:1810.10506 (2018).

[97] Carl W Helstrom, "Quantum detection and estimation theory," Journal of Statistical Physics **1**, 231–252 (1969).

[98] Marcello Benedetti, Edward Grant, Leonard Wossnig, and Simone Severini, "Adversarial quantum circuit learning for pure state approximation," New Journal of Physics (2019), 10.1088/1367-2630/ab14b5.

[99] Ling Hu, Shu-Hao Wu, Weizhou Cai, Yuwei Ma, Xianghao Mu, Yuan Xu, Haiyan Wang, Yipu Song, Dong-Ling Deng, Chang-Ling Zou, *et al.*, "Quantum generative adversarial learning in a superconducting quantum circuit," Science advances **5**, eaav2761 (2019).

[100] Daniel Gottesman, "An introduction to quantum error correction and fault-tolerant quantum computation," in *Quantum information science and its contributions to mathematics, Proceedings of Symposia in Applied Mathematics*, Vol. 68 (2010) pp. 13–58.

[101] Peter D Johnson, Jonathan Romero, Jonathan Olson, Yudong Cao, and Alán Aspuru-Guzik, "Qvector: an algorithm for device-tailored quantum error correction," arXiv preprint arXiv:1711.02249 (2017).

[102] Sumeet Khatri, Ryan LaRose, Alexander Poremba, Lukasz Cincio, Andrew T. Sornborger, and Patrick J. Coles, "Quantum-assisted quantum compiling," Quantum **3**, 140 (2019).

[103] Maria Schuld, M Fingerhuth, and F Petruccione, "Implementing a distance-based classifier with a quantum interference circuit," Europhys. Lett. **119** (2017), 10.1209/0295-5075/119/60002.

[104] Diego Ristè, Marcus P da Silva, Colm A Ryan, Andrew W Cross, Antonio D Córcoles, John A Smolin, Jay M Gambetta, Jerry M Chow, and Blake R Johnson, "Demonstration of quantum advantage in machine learning," npj Quantum Information **3**, 16 (2017).

[105] Gadi Aleksandrowicz, Thomas Alexander, Panagiotis Barkoutsos, Luciano Bello, Yael Ben-Haim, David Bucher, Francisco Jose Cabrera-Hernádez, Jorge Carballo-Franquis, Adrian Chen, Chun-Fu Chen, Jerry M. Chow, Antonio D. Córcoles-Gonzales, Abigail J. Cross, Andrew Cross, Juan Cruz-Benito, Chris Culver, Salvador De La Puente González, Enrique De La Torre, Delton Ding, Eugene Dumitrescu, Ivan Duran, Pieter Eendebak, Mark Everitt, Ismael Faro Sertage, Albert Frisch, Andreas Fuhrer, Jay Gambetta, Borja Godoy Gago, Juan Gomez-Mosquera, Donny Greenberg, Ikko Hamamura, Vojtěch Havlíček, Joe Hellmers, Łukasz Herok, Hiroshi Horii, Shaohan Hu, Takashi Imamichi, Toshinari Itoko, Ali Javadi-Abhari, Naoki Kanazawa, Anton Karazeev, Kevin Krsulich, Peng Liu, Yang Luh, Yunho Maeng, Manoel Marques, Francisco Jose Martín-Fernández, Douglas T. McClure, David McKay, Srujan Meesala, Antonio Mezzacapo, Nikolaj Moll, Diego Moreda Rodríguez, Giacomo Nannicini, Paul Nation, Pauline Ollitrault, Lee James O'Riordan, Hanhee Paik, Jesús Pérez, Anna Phan, Marco Pistoia, Viktor Prutyanov, Max Reuter, Julia Rice, Abdón Rodríguez Davila, Raymond Harry Putra Rudy, Mingi Ryu, Ninad Sathaye, Chris Schnabel, Eddie Schoute, Kanav Setia, Yunong Shi, Adenilton Silva, Yukio Siraichi, Seyon Sivarajah, John A. Smolin, Mathias Soeken, Hitomi Takahashi, Ivano Tavernelli, Charles Taylor, Pete Taylour, Kenso Trabing, Matthew Treinish, Wes Turner, Desiree Vogt-Lee, Christophe Vuillot, Jonathan A. Wildstrom, Jessica Wilson, Erick Winston, Christopher Wood, Stephen Wood, Stefan Wörner, Ismail Yunus Akhalwaya, and Christa Zoufal, "Qiskit: An open-source framework for quantum computing," (2019).

[106] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, and Nathan Killoran, "Pennylane: Automatic differentiation of hybrid quantum-classical computations," arXiv preprint arXiv:1811.04968 (2018).

[107] Xiuzhe Luo, Jin-guo Liu, Pan Zhang, and Lei Wang, "Yao," `https://github.com/QuantumBFS/Yao.jl` (2018).

[108] Mark Fingerhuth, Tom Babej, and Peter Wittek, "Open source software in quantum computing," PLOS ONE **13**, 1–28 (2018).

[109] Edward Farhi and Aram W Harrow, "Quantum supremacy through the quantum approximate optimization algorithm," arXiv preprint arXiv:1602.07674 (2016).

[110] G. G. Guerreschi and A. Y. Matsuura, "Qaoa for max-cut requires hundreds of qubits for quantum speed-up," Scientific Reports **9**, 6903 (2019).

[111] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D Lukin, "Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices," arXiv preprint arXiv:1812.01041 (2018).

[112] Gavin E Crooks, "Performance of the quantum approximate optimization algorithm on the maximum cut problem," arXiv preprint arXiv:1811.08419 (2018).

[113] Jirawat Tangpanitanon, Supanut Thanasilp, Marc-Antoine Lemonde, and Dimitris G Angelakis, "Quantum supremacy with analog quantum processors for material science and machine learning," arXiv preprint arXiv:1906.03860 (2019).

[114] Li Jing, Yichen Shen, Tena Dubcek, John Peurifoy, Scott Skirlo, Yann LeCun, Max Tegmark, and Marin Soljačić, "Tunable efficient unitary neural networks (eunn) and their application to rnns," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (JMLR. org, 2017) pp. 1733–1741.

[115] Stephanie L Hyland and Gunnar Rätsch, "Learning unitary operators with help from u (n)," in *Thirty-First AAAI Conference on Artificial Intelligence* (2017).

[116] Daochen Wang, Oscar Higgott, and Stephen Brierley, "Accelerated variational quantum eigensolver," Phys. Rev. Lett. **122**, 140504 (2019).

[117] Juan Carlos Garcia-Escartin and Pedro Chamorro-Posada, "Swap test and hong-ou-mandel effect are equivalent," Physical Review A **87**, 052330 (2013).

[118] Alexander Cowtan, Silas Dilkes, Ross Duncan, Alexandre Krajenbrink, Will Simmons, and Seyon Sivarajah, "On the qubit routing problem," arXiv preprint arXiv:1902.08091 (2019).

[119] Raban Iten, Oliver Reardon-Smith, Luca Mondada, Ethan Redmond, Ravjot Singh Kohli, and Roger Colbeck, "Introduction to universalqcompiler," arXiv preprint arXiv:1904.01072 (2019).