Single-qubit universal classifier implemented on an ion-trap quantum device

Tarun Dutta,[1,*] Adrián Pérez-Salinas,[2,3,†] Jasper Phua Sing Cheng,[1] José Ignacio Latorre,[1,4,5] and Manas Mukherjee[1,6,7,‡]

[1]*Centre for Quantum Technologies, National University of Singapore, Singapore 117543*
[2]*Barcelona Supercomputing Center, Barcelona 08034, Spain*
[3]*Departament de Física Quàntica i Astrofísica and Institut de Ciències del Cosmos, Universitat de Barcelona, Barcelona 08028, Spain*
[4]*Qilimanjaro Quantum Tech, Barcelona 08007, Spain*
[5]*Quantum Research Centre, Technology Innovation Institute, P. O. Box 9639, Abu Dhabi, United Arab Emirates*
[6]*MajuLab, International Joint Research Unit UMI 3654, CNRS, Université Côte d'Azur, Sorbonne Université, National University of Singapore, Nanyang Technological University, Singapore 117543*
[7]*Center for Quantum Engineering Research and Education, TCG CREST, Techna, Sector V, Kolkata 700091, India*

(Received 11 August 2021; accepted 15 June 2022; published 7 July 2022)

Quantum computers can provide solutions to certain classically intractable problems. However, current devices have limited computational resources, and an effort is made to develop useful quantum algorithms under this constraint. This work demonstrates experimentally that a single-qubit processor can host a universal classifier based on the re-uploading scheme. The quantum processor used in this work is built with an ion trap, providing highly accurate control on a one-qubit system as required by the re-uploading scheme. A set of nontrivial classification tasks are completed successfully. The training procedure is performed in two steps combining simulation and experiment. Final results are benchmarked against exact simulations of the same method and also classical algorithms, showing a competitive performance of the ion-trap quantum classifier. This work constitutes the experimental implementation of a classification algorithm based on the re-uploading scheme.

DOI: 10.1103/PhysRevA.106.012411

I. INTRODUCTION

Quantum computing is an active field of research both in academia and industry. It is expected that quantum computers will provide solutions to classically intractable problems. A handful of problems have been found for which quantum computers can provide a solution with certain, even exponential, speedups with respect to a classical computer [1–5]. On the experimental side, recent advances have achieved a quantum advantage, that is, solving a problem in a quantum computer can be done more efficiently than using classical methods. This has been demonstrated both in superconducting [6] and photonic [7,8] platforms.

Finding a quantum advantage for a practical problem is an open question. Among various possibilities, quantum machine learning is considered to be a potential game changer. Data classification is a ubiquitous problem appearing across many different fields. Classical machine learning provides a plethora of algorithms for this purpose, notably those related to support vector machines [9] or neural networks [10,11]. In recent years, classical machine learning has improved substantially and can now provide solutions to a large variety of classification problems [12,13]. These classical algorithms are generally implemented using specialized software and hardware that are expensive in both computational and energy resources.

Quantum machine learning (QML) aims to combine the spirit of classical machine learning and the capabilities of quantum computing. This field of research is still in an early stage and, in general, cannot compete against the state-of-the-art classical algorithms. However, recent theoretical works have proven quantum advantages in selected problems [14]. Nowadays, most QML algorithms rely on variational methods [15–18], where theoretical arguments assessing robust and general quantum advantages of quantum devices are still missing. Variational quantum algorithms are believed to fit the requirements of noisy intermediate-scale quantum (NISQ) devices [19], motivating the use of these methods for the early experiments of QML. These experiments have progressed in different platforms [20–22]. Photonic devices achieved one of the earliest experiments for addressing a classification problem [20] based on quantum support vector machines [21]. Superconducting qubits have addressed QML using variational algorithms [22].

In this paper, we implement a QML supervised learning algorithm on an ion-trap quantum device. This device is referred to throughout this paper as a quantum processing unit (QPU). Trapped ions are an available platform for designing quantum devices whose main strength is the accurate control of small quantum systems [23,24]. Recent works have widened the range of feasible experiments [25–30]. The data re-uploading algorithm is chosen to address classification [31,32]. The features of the ion-trap device and the algorithmic requirements

[*]cqttaru@nus.edu.sg
[†]adrian.perezsalinas95@gmail.com
[‡]manas.mukh@gmail.com

match each other since data re-uploading is a method specifically designed to take advantage of the fixed Hilbert space available in quantum systems with few qubits. Thus, in order to attain good performances, it is necessary to execute highly accurate quantum operations.

The data re-uploading algorithm is a general scheme for supervised learning in QML using classical data. It was originally conceived as a quantum classifier [32]. The differentiating characteristic of this method is that the data are re-uploaded several times along the computation process. The performance of this algorithm improves as the query complexity, that is, the number of data re-uploadings, increases. The theoretical capability of this model to approximate any function is guaranteed and emerges from its quantum nature [31,33].

The data re-uploading scheme for classification was already tested on classical simulators where not only quantum operations but also the coupling to a noisy environment is taken into account [34]. However, a classical simulator does not necessarily capture the merits and demerits of a real QPU. Indeed, a classical simulator is agnostic to the QPU platform and it does not comprise implementation details such as native gates and specific noise models. The experimental approach presented in this work is a successful experimental implementation of the re-uploading scheme performing quantum classification with a single-qubit quantum processor. On the other hand, the present work accomplishes and surpasses experiments on superconducting qubits for the single-qubit approximant for function regression [31]. The work is oriented to show an experimental realization of the single-qubit classifier. The extension of the re-uploading technique to larger and potentially entangled systems has been explored theoretically [32], but an algorithmic analysis of it is beyond the scope of this paper. Surely, a single-qubit quantum system can be efficiently simulated by classical means, but it allows better insight about the required experimental control to implement the algorithm on a large entangled system to pursue a quantum speed advantage. The present paper is a step taken in the attainment of this purpose.

The paper is structured as follows: Section II explains the experiment including both theoretical and implementation aspects. Section III details the performance of the classifier in several examples. A discussion on the results is carried out in Sec. IV. Some more information on the experiment and results can be found in the Appendixes.

II. FRAMEWORK

The problem solved in this work corresponds to supervised learning. In supervised learning, a model is fed with some data in the form $\mathbb{D}_{\text{train}} = \{\vec{x}, c\}$, where $\vec{x}$ is a feature point and c is the class to which it belongs. The algorithm then learns the latent properties of the dataset in such a way that it can predict the class of previously unseen points belonging to another equivalent dataset $\mathbb{D}_{\text{test}}$.

The algorithm is executed on an ion-trap quantum machine; see Sec. II A. The method of data re-uploading is briefly explained in Sec. II B. The optimization procedure is depicted in Sec. II C. Final experiment-dependents implementations to optimize performance are detailed in Sec. II D.

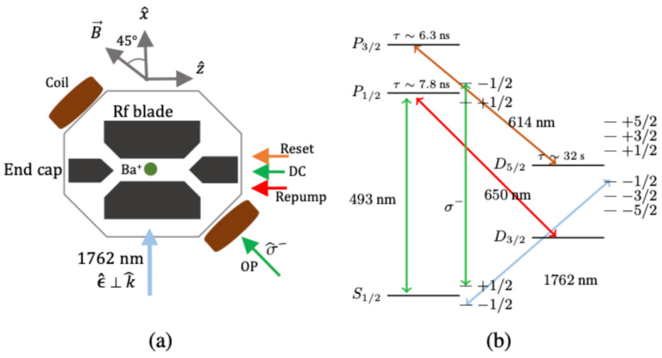

FIG. 1. (a) Schematic drawing of the ion-trap setup viewed from the top showing trap orientation including the directions and polarizations of the laser fields and the magnetic field direction. The $^{138}\text{Ba}^+$ ion is confined in a linear Paul trap. Radial confinement is achieved by a 16.5 MHz rf potential applied to a pair of electrodes (rf blade) via a helical resonator, and endcaps are held at a few hundred volts to provide axial confinement. Two coils in a Helmholtz configuration generate a magnetic field (B) defining the quantization axes. Laser lights at 493 nm (DC), 650 nm (Repump), and 614 nm (Reset) are mixed using a dichroic mirror and aligned to the ion along the direction ($\hat{z}$) of the endcap electrode, whereas the direction of propagation (k) of a 1762 nm laser (polarization, ϵ) is along the $\hat{x}$ direction. (b) Energy level diagram of the $^{138}\text{Ba}^+$ ion showing the Zeeman splitting of atomic states $S_{\frac{1}{2}}$, $P_{\frac{1}{2}}$, and $D_{\frac{5}{2}}$. The degeneracy of the Zeeman sublevels is lifted by a magnetic field of 1.5 G. Laser light at 493 nm is used for Doppler-cooling (DC), optical pumping (OP), and detection. The lasers at 650 and 614 nm pump out the D-states. A narrow linewidth laser light at 1762 nm with outstanding frequency stability is used for manipulating the qubit encoded in the quadrupole transition.

A. Experimental setup

The qubit is realized in a $^{138}\text{Ba}^+$ trapped ion. The computational basis corresponds to the states

$$|0\rangle \equiv S_{\frac{1}{2},-\frac{1}{2}} \quad |1\rangle \equiv D_{\frac{5}{2},-\frac{1}{2}}. \tag{1}$$

Both states are coupled by the electric quadrupole E2 transition at a 1762 nm wavelength shown in Fig. 1(b). The ion is confined in a linear Paul trap, as shown in Fig. 1(a), operating at a radial frequency of 1.5 MHz and an axial frequency of 0.3 MHz. The most relevant parameters of the trapped ion in this experiment are the qubit coherence time (5 ms) and the Rabi π-time of the qubit (12 μs) [35,36]. The qubit is well-characterized in terms of both its internal [36] and external degrees of freedom [37].

Prior to performing each algorithmic cycle, the qubit is initialized to the state $S_{\frac{1}{2},-\frac{1}{2}}$. To do so, the qubit is Doppler-cooled to the Lamb-Dicke regime [36] via a fast dipole transition between S-P levels at 493 nm and a repump laser between D-P levels at 650 nm as explained in Fig. 1(b). In this regime, the internal and external states of the qubit are decoupled and hence single-qubit gate errors are not influenced by the motion of the qubit. Then, the alternative ground state $S_{\frac{1}{2},+\frac{1}{2}}$ is selectively depopulated by optical pumping using a σ^--polarized 493 nm laser together with a 650 nm laser pulse [38]. A σ^--polarized 493 nm laser propagates along the quantization axis defined by the direction of an externally applied magnetic field B. The strength and direction of the B field is optimized to achieve high-fidelity state initialization $\geqslant 99\%$ within an optical pumping time $\leqslant 50$ μs. The magnetic field strength is set at 1.5 G, directed at an angle of 45° with respect to the trap axis (z-axis) as illustrated in Fig. 1(a). The resultant strength of the magnetic field corresponds to a ground-state Zeeman splitting of about 4.2 MHz. Once the qubit is initialized, any single-qubit rotational gate is implemented by resonantly driving the qubit with full control over the laser phase, power, and laser on-time. The qubit is operated by means of an ultralow linewidth laser operated at 1762 nm. The laser is phase-locked to an ultralow expansion cavity achieving a linewidth $\leqslant 100$ Hz. This linewidth is estimated from the measured atomic resonance. The laser-qubit interaction time sets the rotation angle. It is controlled by a radiofrequency (rf) switch. An acousto-optic modulator (AOM) controls the phase and frequency of the laser implementing the rotation gates. Therefore, direct and precise control over the axis and angle of rotations is achieved.

A direct digital synthesizer (DDS) based on an *AD*9959 chip is used to control the application of gates on the qubit. The DDS is disciplined by a rubidium frequency standard (*SRS FS725*), eliminating the long-term frequency drift of the DDS clock, thus maintaining the errors in phase relations of the sequential gates of the classifier below 0.01%. Since the parameters are uploaded to the circuit on-the-fly, the latency of uploading the DDS parameters plays a crucial role. The latency is minimized by preloading the full sequence of the phase, frequency, and power data to an on-chip memory of the DDS. The current version of the DDS controlling the phase of the laser is limited by the on-chip memory to 16 phase modulation steps, thus limiting the layers to six, which is sufficient for the current discussion. The DDS output is then controlled by an external trigger generated from a field programmable gate array (FPGA) based pulse pattern generator that controls the time sequence of the entire experiment with a time jitter below 10 ns. See Fig. 2 for a scheme of the experiment at the time-sequence level.

The phase of the AOM is directly controlled by a DDS which supplies the rf signal to the AOM via an amplifier. To avoid accumulation of phase noise, during the on-off time of the laser, rf pulse shaping has been implemented before being fed to the AOM for switching. All the rf sources are synchronized to the DDS clock. The main upgrade with respect to our previous setup lies in the hardware to better control the qubit phase. In addition, we modified the control software to implement quantum algorithms requiring long circuit depth with low latency.

For the measurement step, the final state is projected onto the computational state $|0\rangle$, and the ground-state occupation probability is obtained by observing spontaneously emitted photons while the qubit is excited by the Doppler-cooling 493 nm laser. With a photon collection time of 2 ms, the qubit state is determined by choosing a photon-count threshold of 15 counts/2 ms such that the bright state $|0\rangle \equiv S_{1/2}$ is clearly discriminated from the dark $|1\rangle \equiv D_{5/2}$ state. This projective measurement is 99.8% [35] efficient in discriminating both states. Taking repeated measurements on the same state provides the probability and hence the projection of the state along the σ_z-axis.

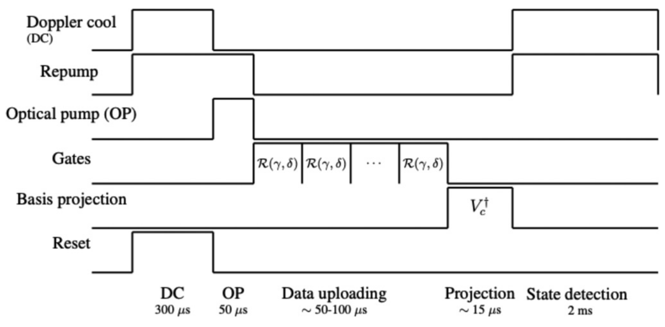

FIG. 2. Time sequence used in the experiment to perform each classifier measurement. At the beginning of each run, the qubit is reset to the $|0\rangle \equiv |S_{1/2,-1/2}\rangle$ state via Doppler cooling and optical pump. Then, the different re-uploading $\mathcal{R}(\gamma, \delta)$ gates are applied sequentially via laser pulses. The duration of each pulse depends on the value of its β parameter. As a final step, the state is projected to be compared against a label state defined by $V_c|0\rangle = |\phi_c\rangle$, and the relative fidelity is measured.

B. Re-uploading scheme

The re-uploading algorithm presented here delivers an output quantum state in the form

$$|\psi(\vec{x}, \Theta)\rangle = \prod_{i=1}^{L} U(\vec{x}, \vec{\theta}_i)|0\rangle, \tag{2}$$

where each $U(\vec{x}, \vec{\theta}_i)$ is referred to as a layer and depends on both sampling data $\vec{x}$ and tunable parameters $\vec{\theta}_i$, with $\Theta = \{\vec{\theta}_1, \vec{\theta}_2, \ldots, \vec{\theta}_L\}$. Each layer is composed by R_y and R_z rotations. In particular, two *Ansätzes* are proposed in this work and inspired from Ref. [32]. *Ansatz* A holds for multidimensional data and introduces $\vec{x}$ only in R_y rotations as

$$U_A(\vec{x}, \vec{\theta}) = R_z(\varphi)R_y(\vec{\omega} \cdot \vec{x} + \alpha), \tag{3}$$

where $\vec{\omega}$ is a vector of tunable weights, and α, φ are tunable angles acting as biases. On the other hand, *Ansatz* B holds only for two-dimensional datasets and introduces data in all rotation gates. It is defined as

$$U_B(\vec{x}, \vec{\theta}) = R_z(\omega x_2 + \beta)R_y(\nu x_1 + \alpha), \tag{4}$$

where ω and ν are tunable weights and α, β are tunable biases. In both cases, $\vec{\theta}$ are flexible sets of parameters that accommodate the requirements of each gate.

The next step consists in training the circuit to obtain a set of parameters Θ such that it solves a given problem. Since we aim to build a classifier for supervised learning, each sample $\vec{x}$ in the dataset is associated with a class within the set $\{\vec{c}\}$. We must also define as many label states $|\phi_c\rangle$ as existing classes. Then, the probability of a state corresponding to a class $c \in \vec{c}$ corresponds to

$$\text{Prob}(\vec{x}, \Theta|c) \propto |\langle\phi_c|\psi(\vec{x}, \Theta)\rangle|^2, \tag{5}$$

properly normalized. This probability assignment needs to choose a target ket $|\phi_c\rangle$ for each class in every different problem considered herein. The general rule is to choose the target kets as distant as possible from each other to maximize the distinguishability among the classes [32]. Given the output from the quantum computer, a final class c is assigned to a given pattern according to the label ket that maximizes the probability $\text{Prob}(\vec{x}, \Theta|c)$.

For the algorithm to work properly, it is necessary to find the configuration of parameters Θ such that the fidelity between the output states and their corresponding label states is on average maximum for all patterns in the training dataset, $\{\vec{x}, c\} \in \mathbb{D}_{\text{train}}$. For a training dataset with N_{train} patterns, the loss function describing this behavior is just

$$\chi^2(\Theta; \mathbb{D}_{\text{train}}) = \frac{1}{N_{\text{train}}} \sum_{(\vec{x},c)\in\mathbb{D}_{\text{train}}} (|\langle\phi_c|\psi(\vec{x}, \Theta)\rangle|^2 - 1)^2, \quad (6)$$

and an optimal configuration of Θ can be obtained by minimizing $\chi^2(\Theta, \mathbb{D}_{\text{train}})$ using classical standard optimizers like genetic algorithms [39], or L-BFGS-B [40]. The results displayed correspond to the best instance.

The performance of supervised-learning models is not measured by their capability to learn the training dataset, but rather by their generalization power. Then, the quantity of interest is the accuracy $\mathcal{A}$. This quantity counts the number of correct guesses over samples on an unlearned test dataset $\mathbb{D}_{\text{test}}$. The guessed class is

$$c_g(\vec{x}, \Theta) = \arg\max_{\vec{c}} |\langle\phi_c|\psi(\vec{x}, \Theta)\rangle|^2 \quad (7)$$

and the accuracy $\mathcal{A}(\Theta; \mathbb{D}_{\text{test}})$ is

$$\mathcal{A}(\Theta; \mathbb{D}_{\text{test}}) = \frac{1}{N_{\text{test}}} \sum_{(\vec{x},c)\in\mathbb{D}_{\text{test}}} \text{bool}(c_g(\vec{x}, \Theta) = c), \quad (8)$$

where $\text{bool}(a = b)$ is 1 if $a = b$, and 0 otherwise. The value of $\mathcal{A}(\Theta; \mathbb{D}_{\text{test}})$ is bounded between 0 and 1, 1 being the optimal result.

C. Optimization

The classifier proposed in this work is trained in two steps. First, given a $\mathbb{D}_{\text{train}}$ dataset, a simulated version of the model is created and trained using the χ^2. The optimization returns a set of parameters Θ^{sim}. Using exact simulations enables us to circumvent difficulties in the training process due to noise and uncertainties.

In a second step, the obtained parameters are ported to the experimental classifier. A relevant improvement can be obtained by fine-tuning the given parameters when executed on the quantum hardware. In most scenarios, parameters for simulation and experiment do not exactly match each other. The difference may stem from the experimental inaccuracy when implementing optimal angles and occasional loss of information due to collisions with background molecules. This second optimization step polishes the experiment to reduce any systematical error occurring during the execution of the quantum circuit. The parameter set Θ^{sim} is taken as a starting point to explore the parameter space in its vicinity. A new set of parameters reducing experimental inaccuracies is obtained as $\Theta^q = \Theta^{\text{sim}} + \delta\Theta$. The figure of merit to be optimized in this case is $\mathcal{A}^q$. This second step is currently available for quantum devices only if the loss function in the parameter space near the vicinity of the minimum is shallow, and large deviations from the optimal parameters translate into small changes in the loss functions. See Fig. 3 for a scheme describing the two-step optimization performed.

The full multi-dimensional scan of the space of parameters is very time consuming in the present setup. Despite the limitation, in this work the optimization of the parameters is performed sequentially taking two parameters at a time while

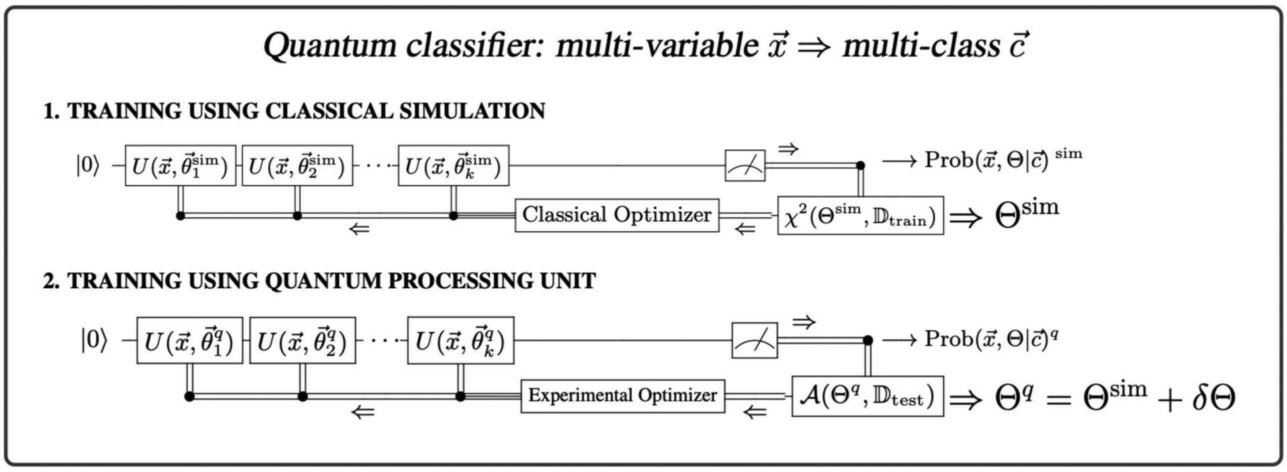
Quantum classifier: multi-variable $\vec{x} \Rightarrow$ multi-class $\vec{c}$
1. TRAINING USING CLASSICAL SIMULATION
$|0\rangle$
$U(\vec{x}, \vec{\theta}_1^{\text{sim}})$
$U(\vec{x}, \vec{\theta}_2^{\text{sim}})$
$U(\vec{x}, \vec{\theta}_k^{\text{sim}})$
$\rightarrow \text{Prob}(\vec{x}, \Theta|\vec{c})^{\text{sim}}$
Classical Optimizer
$\chi^2(\Theta^{\text{sim}}, \mathbb{D}_{\text{train}}) \Rightarrow \Theta^{\text{sim}}$
2. TRAINING USING QUANTUM PROCESSING UNIT
$|0\rangle$
$U(\vec{x}, \vec{\theta}_1^q)$
$U(\vec{x}, \vec{\theta}_2^q)$
$U(\vec{x}, \vec{\theta}_k^q)$
$\rightarrow \text{Prob}(\vec{x}, \Theta|\vec{c})^q$
Experimental Optimizer
$\mathcal{A}(\Theta^q, \mathbb{D}_{\text{test}}) \Rightarrow \Theta^q = \Theta^{\text{sim}} + \delta\Theta$

FIG. 3. Schematic description of the optimization algorithm used in this work. 1. Data re-uploading is trained using a classical simulation. The simulated quantum circuit is composed of single-qubit gates U that depend on variational parameters $\Theta = \{\vec{\theta}_1, \vec{\theta}_2, \dots, \vec{\theta}_L\}$ and the variables $\vec{x}$ associated with a given pattern. The output state is measured to obtain a vector $\text{Prob}(\vec{x}, \Theta|\vec{c})$ encoding the probabilities that will serve to classify the given pattern into a category. A classical optimization is performed to obtain optimal values Θ^{sim}. This optimization is driven by the cost function χ^2 evaluated on training data, $\mathbb{D}_{\text{train}}$. 2. A further optimization is accomplished only using the quantum device, taking as a starting point the values Θ^{sim} and delivering a better set Θ^q. The quantity that is now maximized is the accuracy $\mathcal{A}$ evaluated on the test dataset $\mathbb{D}_{\text{test}}$. The aim of the experimental optimization is to mitigate and even compensate for possible systematic experimental errors.

keeping others fixed. A two-dimensional scan is performed for those parameters in the vicinity around Θ^{sim}. The pair of parameters is then modified for improving the obtained accuracy. With each iteration, one new parameter is taken into account. This method returns substantial improvements only on the first layers, and it can be stopped after a few iterations without a significant loss of performance. As seen in Ref. [31], the first layers give a first approximation to the solution, and every change in their parameters is propagated through the entire circuit. Thus, the earlier a layer is applied, the larger the influence it has in the final result. Therefore, experimentally optimizing the first layers will turn into a more rewarding improvement.

Notice that the two-step optimization procedure returns three different values of the accuracy $\mathcal{A}$ to be measured:

(i) $\mathcal{A}^*$ obtained by running an exact simulation of the quantum circuit with the optimal parameters Θ^{sim}.

(ii) $\mathcal{A}^{\text{sim}}$ run on the QPU with the same parameters Θ^{sim}.

(iii) $\mathcal{A}^q$ obtained with the parameters Θ^q after the second optimization step.

In addition, for every classification problem, the obtained accuracies can be compared with genuinely classical models. It is expected that experimental errors will deteriorate the ideal performance of the computer. However, results remain very competitive in the experimental setup.

Both the two-step optimization and the accuracies related to classification problems on experiments are features surpassing previous results in Ref. [31]. In previous works, parameters are directly ported from simulation setups to the experiment. A reasonable degradation in the final results is observed, but there is no further study on how to improve the hardware implementation of the algorithm. With regard to the accuracies, the previous work does not need to generalize from training to test dataset, but only shows the flexibility to mimic a given behavior. In the classifier exposed in this work, generalization is a requirement that is successfully achieved.

D. Optimal hardware control

To improve the performance of the classifier to the limit of the capabilities of the present QPU, some features were implemented specifically for this experiment.

First, the error in the applied gates is reduced by cleverly choosing the application of the axis. In the ion qubit, active rotations R_x and R_y are realized by the interaction of the resonant laser with the qubit at the specified frequency during a time proportional to the rotation angle. On the other hand, R_z can be implemented by a combination of the other two active gates or by varying the qubit energy [41,42]. Both of these methods are prone to error as the qubit-light interaction is switched on for a certain time. We chose instead to perform the R_z by changing the laser phase without interacting with the ion [43], thus the resultant error is limited by only the R_y gate in each layer.

Second, the depth of the circuit can be effectively halved by combining $R_z(\gamma)$ and $R_y(\delta)$ gates into a single gate $\mathcal{R}(\gamma, \delta)$, which applies a rotation of angle δ around the axis $(\gamma, 0)$, defined in spherical coordinates.

This rotation around an arbitrary axis is defined as

$$\mathcal{R}(\gamma, \delta) = \begin{pmatrix} \cos\frac{\delta}{2} & -ie^{-i\gamma}\sin\frac{\delta}{2} \\ -ie^{i\gamma}\sin\frac{\delta}{2} & \cos\frac{\delta}{2} \end{pmatrix}, \tag{9}$$

which can be understood as

$$\mathcal{R}(\gamma, \delta) = R_z(\gamma)R_x(\delta)R_z(-\gamma). \tag{10}$$

Thus, it is possible to relate $R_x(\delta) = \mathcal{R}(\gamma, \delta)$ and $R_y(\delta) = \mathcal{R}(\pi/2, \delta)$. Taking the *Ansatz* from Sec. II B, it is possible to decompose those gates into $\mathcal{R}(\gamma, \delta)$ operations to effectively reduce the number of operations. The first $R_y(\cdot)$ gate must be applied on its own. For the nth step, the R_z rotation from the $(n-1)$th layer and the R_y from the nth layers can be composed into only one gate. As a consequence, the nth $\mathcal{R}$ gate must adjust its parameters to

$$\bar{\theta}_1^z = \pi/2 + \theta_1^z, \tag{11}$$

$$\bar{\theta}_n^z = \bar{\theta}_{n-1}^z + \theta_n^z. \tag{12}$$

See Fig. 4(a) for a schematic description of this process.

This reduction can only be applied to pairs of gates and not to the entire circuit. The reason is that the data (x) and tunable parameters $(\vec{\theta})$ from Eq. (2) come into the circuit through linear operations. Thus, two gates can be easily combined by classically performing this operation. On the other hand, the chain of repeated gates gives rise to complex nonlinear behaviors whose explicit form is not easily computed by analytical means.

The measurement process to construct the cost function involves the quantities $|\langle\phi_c|\psi(\vec{x}, \Theta)\rangle|$ as described in Eq. (5). A direct method to obtain this quantity is by performing tomography and obtaining an indirect computation of the fidelity. In this case, the fidelity is measured by direct comparison. This method requires fewer operations than tomography, especially when the system size increases, although the difference is not very significant in the single-qubit example. For this purpose, the label states are defined by means of a simple gate V_c as

$$|\phi_c\rangle = V_c|0\rangle = R_z(\eta)R_y(\lambda)|0\rangle, \tag{13}$$

where the parameters η, λ are problem-dependent. They must be tuned to accommodate several states as orthogonally as possible across the space in the Bloch sphere. This is a problem on its own, but in some cases, including those tackled in the present work, the solution is trivial [32]. Thus, fidelity can be measured by comparing the output and label states as

$$|\langle\phi_c|\psi(\vec{x}, \Theta)\rangle|^2 = |\langle 0|V_c^\dagger U(\vec{x}, \Theta)|0\rangle|^2 = P_0(V_c^\dagger U(\vec{x}, \Theta)), \tag{14}$$

where the last term corresponds to the probability of measuring the state $|0\rangle$ after executing the circuit composed by both operators. The corresponding scheme can be viewed in Fig. 4(b).

III. RESULTS

The problems solved in this work consists in separating points in a feature space depending on its location. The classes are defined by different curves acting as boundaries. The problems proposed can only be solved if the classifier is flexible enough so as to map close points to distant areas of the Hilbert

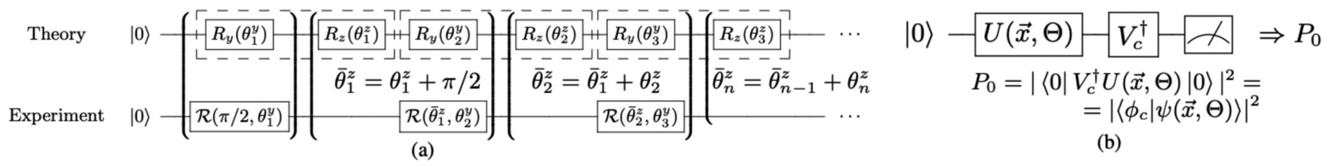
Theory
Experiment
$|0\rangle$
$R_y(\theta_1^y)$
$R_z(\theta_1^z)$
$R_y(\theta_2^y)$
$R_z(\theta_2^z)$
$R_y(\theta_3^y)$
$R_z(\theta_3^z)$
$\bar\theta_1^z = \theta_1^z + \pi/2$
$\bar\theta_2^z = \bar\theta_1^z + \theta_2^z$
$\bar\theta_n^z = \bar\theta_{n-1}^z + \theta_n^z$
$\mathcal{R}(\pi/2, \theta_1^y)$
$\mathcal{R}(\bar\theta_1^z, \theta_2^y)$
$\mathcal{R}(\bar\theta_2^z, \theta_3^y)$
(a)
$U(\vec{x}, \Theta)$
$V_c^\dagger$
$\Rightarrow P_0$
$P_0 = |\langle 0| V_c^\dagger U(\vec{x},\Theta) |0\rangle|^2 = |\langle\phi_c|\psi(\vec{x},\Theta)\rangle|^2$
(b)

FIG. 4. Methods applied in the classifier to optimize the implementation. (a) Depth reduction of layers. Layers are grouped in dashed lines. The rotations R_y from a layer and R_z for the previous one are combined to be executed by means of only one $\mathcal{R}(\cdot,\cdot)$ operation, thus effectively reducing the depth of the circuit. This is indicated by brackets. For consistency, a redefinition of parameters in the R_z rotations must be carried, as detailed in the figure. (b) Measurement of the fidelity between two states. The state $|\psi(\vec{x},\Theta)\rangle = U(\vec{x},\Theta)|0\rangle$ is compared against $|\phi_c\rangle = V_c|0\rangle$. The probability of measuring the $|0\rangle$ state at the end of this circuit equals the relative fidelity.

space. Experimental results are presented with an increasing level of difficulty. For every instance, we compare the accuracy of the simulated and the experimental classifiers $\mathcal{A}^*$ and $\mathcal{A}^{\rm sim}$, respectively, and we benchmark them against classical algorithms. In all cases, a training dataset of 200 randomly distributed points is used for optimization. The accuracy is tested against 1000 different points. This way, generalization is also tested on unseen data. In two cases, namely a circle and a hypersphere, see Table I, due to its high computational cost, the experimental optimization step to obtain $\mathcal{A}^q$ is carried. A full analysis is deferred to a separate publication. Images illustrating results are included for some examples. Additional images can be found in the Appendixes.

A. Two-dimensional binary classification

A binary classifier is designed to learn the difference between points in a plane inside (0) and outside (1) a circle. The circle is centered at the middle of the plane and has half the total area, so that the instances belonging to the two classes are balanced. A random classifier would only guess 50% of test data properly.

Figure 5 shows experimental results for this classification problem for an increasing number of layers, and a comparison between the test data as classified by the QPU and an ideal classical qubit simulator. Here, each data point is depicted by its coordinates (x_1, x_2) and colored according to its binary classification, namely blue diamonds for inside and orange circles for outside the circle, shown as a black solid line. The figure shows the gradual improvement of classification as more layers are added, confirming results from Ref. [32]. For four layers, the classification accuracy for the QPU is $\mathcal{A}^{\rm sim} = 93 \pm 2\%$, slightly lower than its classically simulated counterpart $\mathcal{A}^* = 97\%$. The error in $\mathcal{A}^{\rm sim}$ refers to the standard deviation of 10 repeated trials performed on the same dataset, since the inherent quantum uncertainty leads to different values of accuracy in each instance.

It is interesting to note a difference in the behavior of both simulated and quantum classifiers. In the simulation, Fig. 5(e), the guessed boundary between classes is sharply defined, even though the guesses do not match exactly the data and the circle is slightly deformed. The results on the experimental data show uncertainty in the determination of data. See, for instance, the higher part of the circle. There are several points guessed to belong to different classes interspersed in a small area. The origin of this phenomenon is the inherent sampling uncertainty of the quantum device.

In this example, the experimental second optimization step is also performed. Figure 6(a) shows the landscape of the accuracy for a specific subset of parameters in the vicinity of the optimum point as provided by the classical simulation $\Theta^{\rm sim}$. In this case, only three parameters effectively contribute to the final result, and thus the cost of searching optimal experimental configurations using a scanning technique is manageable.

The error in classification of the experimental implementation with respect to the exact simulation is also depicted in Fig. 6(b). After the experimental optimization, the deviation between $\mathcal{A}^q$ and $\mathcal{A}^*$ nearly vanishes within the experimental uncertainty. After optimization, we obtain an improvement by more than 5%, which results in the accuracy of QPU being as good as that of the simulator. This is better understood by looking at the optimized error in Fig. 6(b) with respect to the simulation plotted as a function of the number of layers in the supervised learning process. In a realistic situation, the loss of information due to collision (small percentage of the systematic error) can also be corrected by the observation of the emitted light level at the cost of a longer operation time.

TABLE I. Comparison between the accuracies, in %, of the single-qubit re-uploading quantum classifier and two well-known classical classification techniques, namely single-hidden-layer neural networks (NNs) and support vector machines (SVMs). The experimental data and their simulated analog are provided here with four layers and 100 repetitions on the quantum part, and the equivalent number of parameters for the neural network. The uncertainty of the experimental data is ±2%. The error refers to the standard deviation of 10 repeated trials performed on the same dataset, and it implies that underlying systematic uncertainty leads to an uncertainty of the accuracy; see Appendix A for a detailed analysis. Only two cases have been further optimized using an exploration done only with the quantum device.

		Classical		Re-uploading			
Problem	(no. classes)	$\mathcal{A}^{\rm NN}$	$\mathcal{A}^{\rm SVM}$	$\mathcal{A}^*$	$\mathcal{A}^{\rm sim}$	$\mathcal{A}^q$	*Ansatz*
Circle	(2)	98	96	97	93	96	A
Crown	(2)	71	82	92	87		B
Nonconvex	(2)	98	79	95	92		B
Sphere	(2)	95	91	74	66		A
Hypersphere	(2)	76	92	75	64	73	A
Tricrown	(3)	97	83	95	91		A
Three circles	(4)	93	92	90	85		B
Squares	(4)	99	95	97	93		A
Wavy lines	(4)	99	89	94	90		A

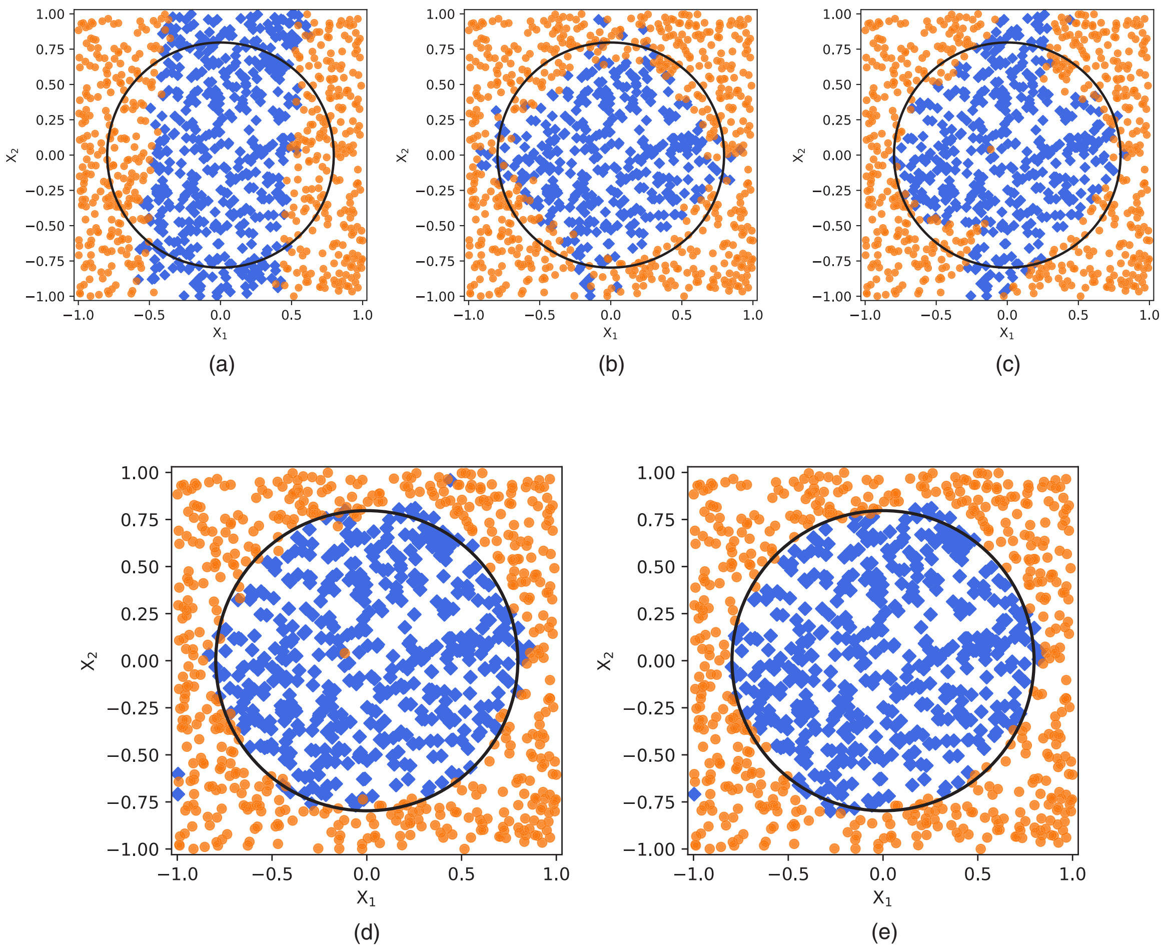

FIG. 5. Results for the dataset of the ion-trap re-uploading classifier as compared to classical simulation. Top, from left to right: one, two, and three layers in QPU; bottom: four layers in QPU ($\chi^2 = 93 \pm 2\%$) and simulation ($\chi^2 = 97\%$). 1000 random data points are tested, and the guessed classes are depicted as blue diamonds and orange circles, respectively, for points inside and outside the circular boundary, shown by a solid line. In (a)–(d), the depth of the circuit is increased from one to four layers, showing gradual improvements of the classification. The result of the four-layer QPU classifier (d) is compared with the equivalent four-layer simulation (e) for a benchmark. Notice that the border between classes in the experimental results is not as sharply defined as in the simulated classification due to the uncertainty of the quantum measurements and systematic errors.

However, those collisions that are not detected during the cooling/initialization stage of the processor would still contribute to the error budget. Experimental errors are estimated separately. A detailed discussion on experimental errors is carried out in Appendix A.

Additionally, two more two-dimensional (2D) binary classification problems, namely *nonconvex* and *crown*, can be seen in Fig. 9 of Appendix B for four layers. In this case, the final accuracies ($\mathcal{A}^{\text{sim}}$)/($\mathcal{A}^{*}$) are *nonconvex*: (92%)/(95%); *crown*: (87%)/(92%).

B. Higher-dimensional classification

A single qubit is able to address classification problems in an arbitrary number of dimensions [31–33]. To test a classification performance for high-dimensional data, we extend the 2D-*circle* problem to 3D-*sphere* and 4D-*hypersphere* problems. The statement of the problem is equivalent, and only the radius of the boundary is modified to accommodate the requirement that every class corresponds to half the feature space. Results for the *hypersphere* classification are depicted in Fig. 11 of Appendix B. The error in the experimental setup using Θ^{sim} can be further reduced from ∼13% to ∼2% after the experimental optimization step is performed.

C. Multiclass classification

The single-qubit QPU can also maximally separate into three, four, or more different classes [32]. To prove this experimentally, we provide an example of a three-class classification, *tricrown*, and three different examples of a four-classes problem, namely *three circles*, *squares,* and *waves*, in a 2D feature space. The results for all examples are depicted in Fig. 7 and in Fig. 10 of Appendix B for four layers for QPU and a classical simulator. Results are, respectively, *tricrown*: (95%)/(91%); *three circles*: (85%)/(90%); *squares*: (93%)/(97%); *waves*: (90%)/(94%).

These datasets present different hardness for classification [44]. The *three circles* feature is composed of three different and separated classes, that is, the circles, and a fourth

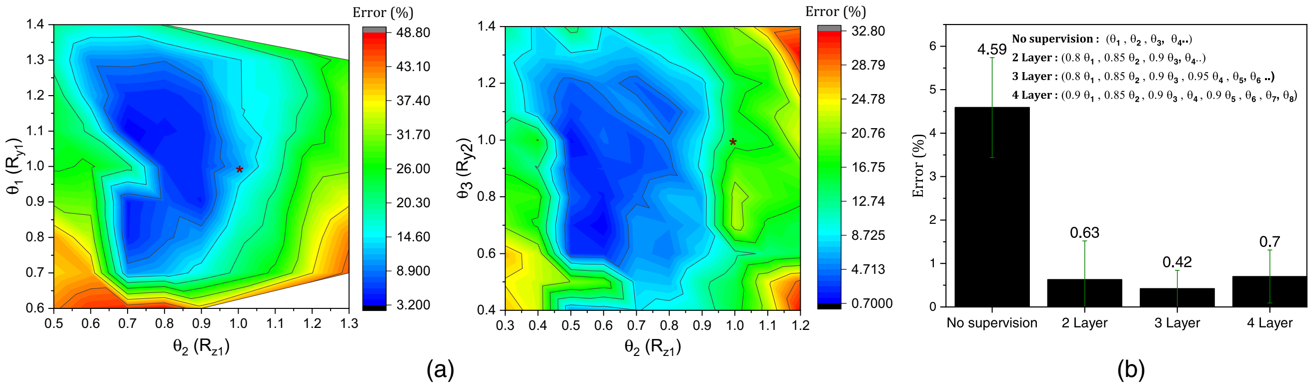

FIG. 6. Results for the experimental second step for optimization in the circle problem. The classifier is first trained on a classical simulator, obtaining a set of parameters Θ^{sim}. In a second step, the vicinity of this point in the parameter space is explored using the QPU. A new set of parameters $\Theta^q = \Theta^{\mathrm{sim}} + \delta\Theta$ is selected to optimize the accuracy of the classifier by mitigating experimental errors. (a) Error surface for the deviation of the optimal parameters from the Θ^{sim} denoted by $\star$. These plots correspond to the two-layer classifier from Fig. 5. (b) Betterment of the accuracy in the three- and four-layer QPU classifier, as compared to simulated results. The error in classification is about 5% when Θ^{sim} is used. On the contrary, if Θ^q are used, the errors are reduced below 1%. The improvements in the first two layers are the most prominent.

class filling the space in between. Thus, the classifier is forced to separate the space corresponding to different classes. In the *squares* case, all classes are equivalent and connected to each other. For *waves*, the difficulty of nonconvex datasets is added. Despite various level of difficulties in classification, the experimental classifier succeeds in solving all with nearly the same percentage.

D. Summary of results

Table I presents a summary of the accuracies obtained by the experimental quantum classifier as compared to the simulated results serving as the starting point for the experimental optimization. Each problem was solved using an independent *Ansatz* (A or B) specified in the table. We also benchmark the quantum results against classical models whose complexity is comparable. To be precise, we have considered a single-

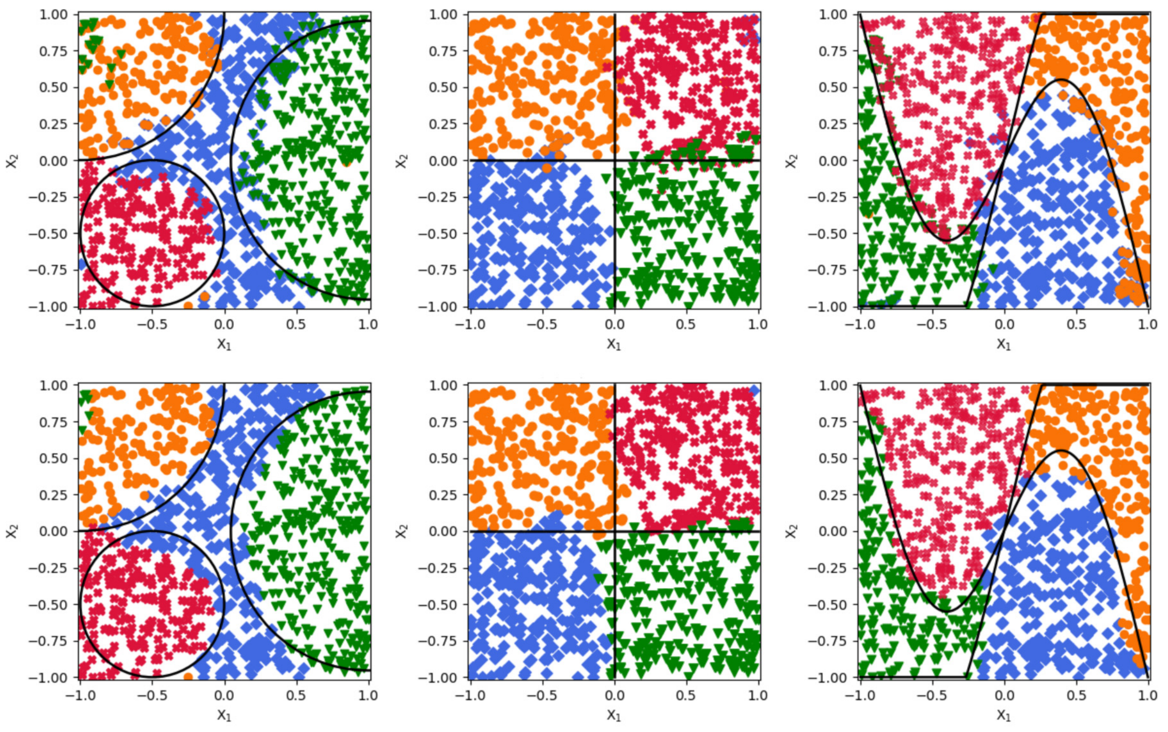

FIG. 7. Multiclass classification. The columns show, from left to right, *three circles*, *squares*, and *wavy lines* problems. The top row corresponds to experimental QPU results, while the bottom row corresponds to classical simulations. Results include 1000 random data points. The classes are depicted with different colors and symbols, and the boundaries are defined by solid black lines. Notice that the border between classes in the experimental results is not sharply defined, unlike in the simulated classification. This difference is due to the uncertainty of the quantum measurements.

hidden-layer neural network and a support vector machine classifier. In the case of neural networks, several activation functions were tested for every problem, and the result depicted in Table I is the best performing one. The number of neurons in the hidden layer is chosen to match the number of parameters in the quantum classifier. In the case of a support vector machine classifier, different kernels are used, and again the best result is retrieved. Notice that, in this case, it is not possible to tune the number of parameters available to carry the classification since they depend on the size of the training set. This model is, however, shown for completeness. In all classical cases, the computations were done using the SCIKIT-LEARN python package [45] and following predefined models.

In light of these results, it is possible to see that the performance of the quantum classifier is comparable to that of classical classifiers when the training dataset has a low feature dimension. On the contrary, when datasets with more features are considered, the performance degrades but is still comparable to simulated results. This limitation can be addressed by increasing the flexibility of the model, as was shown in Ref. [32]. Higher flexibility can be achieved by either adding more layers to the quantum classifier or changing the encoding scheme to one with more degrees of freedom.

IV. DISCUSSION

We have experimentally implemented a single-qubit quantum supervised classifier on a QPU. The QPU is based on an ion trap platform, which is known for high-fidelity gates [35,43,46]. This high fidelity allows us to implement a quantum classifier based on the recently proposed re-uploading algorithm [31,32]. The experiment reported herein is an implementation of problems of this kind in the field of classification with minimal quantum resources that surpasses the experimental results from previous work [31]. This work constitutes experimental proof that the re-uploading scheme may become part of QML strategies for classification.

To enhance the performance of the algorithm, the classifier is trained in two steps, first using a classical simulator, and then at the gate level of the quantum device. This second step enables us to mitigate possible systematic errors from the quantum device. With this method, the accuracy of the classifier on the quantum experiment is enhanced up to 5–10 % depending on the problem. The resultant high performance of the QPU allowed a comparable outcome to that of a classical simulator and classical classifiers of equivalent complexity in the number of parameters used.

The examples of the datasets provided here include rudimentary but nontrivial classification tasks that were successfully solved, including binary, multiclass, and high-dimensional datasets. To complete the benchmarking, we showed that our QPU can not only classify multiclass and higher-dimensional feature maps, but it also shows competing results in classifying nonconvex and linear feature maps. Both experimental and simulated results are benchmarked against well-established classical methods. The performances for quantum and classical cases are comparable; see Table I. These results are aligned with the analysis from Ref. [31], where it is demonstrated that single-qubit re-uploading circuits with N layers are formally equivalent to single-hidden-layer neural networks with N neurons in the hidden layer.

While other approaches explore the possibilities to classify complex datasets by implementing a variety of quantum models, the aim of the re-uploading scheme is to compare their performance with classical procedures of similar complexities. This work demonstrates experimentally that a minimalist quantum system, namely a single qubit, is able to perform nontrivial classification tasks. This may make quantum algorithms advantageous when used as a subroutine in QML tasks with minimal quantum resources.

ACKNOWLEDGMENTS

This research is supported by the Quantum Engineering Program under National Research Foundation (Award No. QEP-P6) and CQT Core grant. A.P.-S. acknowledges financial support from Secretaria d'Universitats i Recerca del Departament d'Empresa i Coneixement de la Generalitat de Catalunya, cofunded by the European Union Regional Development Fund within the ERDF Operational Program of Catalunya (project QuantumCat, Ref. 001-P-001644).

APPENDIX A: ERROR ANALYSIS

The accuracy of the data re-uploading algorithm relies primarily on the fidelity of an individual single-qubit rotation gate. The gate, as explained in the main text, is operated by controlling the laser phase and interaction time while maintaining the intensity at the ion position and the frequency of the laser constant. Therefore, the residual error in the gate operation is reflected in the accuracy of the classifier as define in Eq. (8). In Eq. (9), the rotation angles γ and δ are related to physical quantities as

$$\gamma = (\Delta/\hbar)t_{\mathrm{op}} + \delta\phi, \tag{A1}$$

$$\delta = (\Omega/\hbar)t_{\mathrm{op}}, \tag{A2}$$

where Δ is the laser detuning, t_{op} is the operation time of gates, Ω' is the modified Rabi frequency, and $\delta\phi$ is the relative phase of the laser with respect to the qubit. The modified Rabi frequency $\Omega' = \sqrt{\Omega_0^2 + \Delta^2}$, with Ω_0 denoting the resonant Rabi frequency. Furthermore, the resonant Rabi frequency is proportional to the square root of the intensity, I_0, at the ion position. Therefore, each of the independent variables $\delta\phi$, t_{op}, Δ, and I_0 contributes to the error in a rotation gate, thus influencing the accuracy of the quantum data re-upload classifier. We have characterized these factors separately using the *circle* classification problem as a test-bed. The errors shown in Fig. 8 are obtained as an inaccuracy percentage of classification as compared to the simulation results. In the following, the details of the influence of each of these parameters on the accuracy of the classifier are discussed:

(i) *Phase:* The DDS controls the rf phase of the AOM, which determines the relative phase of the laser. Each DDS is synchronized to a rubidium atomic clock, which is accurate to one part in 10^{10} and thus contributes negligibly to the phase error. The DDS is, however, triggered by the FPGA, which

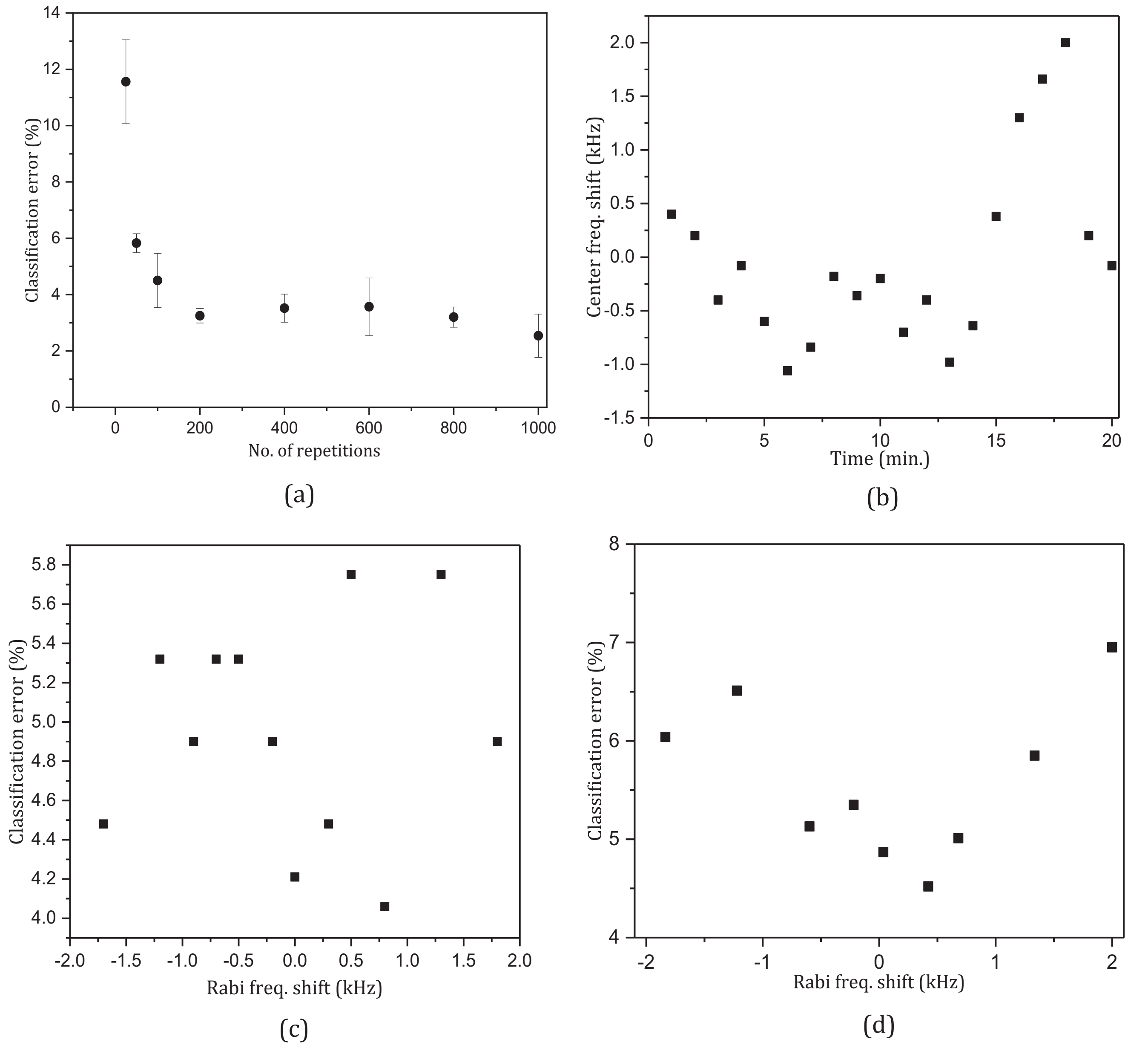

FIG. 8. Systematic error analysis: All the results shown here are related to the binary classification of a circle as in Fig. 5. The errors are classification errors. (a) The classifier error as a function of the number of repeated experiments. The error bar at each point corresponds to one standard deviation of a number of repeat measurements for the same number of repeated experiments under the same condition. The exact number of repeat measurements varies between 5 and 10. (b) Variation of the resonance frequency as a function of time. The range of Rabi frequency fluctuation within a typical experimental time of $\leqslant$10 min is about 2 kHz. (c) Error in binary classification of a circle feature with the variation of laser frequency detuning measured in terms of the Rabi frequency. The variation in the value of classification error is about 2% within the experimental time of ~10 min. (d) The same plot as in (c) but by varying the laser power measured in terms of the Rabi frequency.

has time jitter below 10 ns leading to phase noise on the qubit below 0.1% for a Rabi π time of 12 μs.

(ii) *Interaction time:* The laser-qubit interaction time is determined by the FPGA, which is precise to 1 ns. Therefore, the contribution to the accuracy of the classifier is less than 0.01%. However, due to the time jitter below 10 ns, its contribution to the accuracy is below 0.1%. Occasional collision with the residual background gas molecule during the interaction time leads to a projection to the state $|0\rangle$, thus losing the final-state information and hence error in the classification. Usually this error becomes smaller with larger statistics.

(iii) *Laser-qubit detuning:* The detuning of the laser with respect to the qubit frequency denoted by Δ modifies the Rabi frequency. To quantify the influence of any unwanted fluctuations of the detuning on the classifier accuracy, we first quantified the range of Rabi frequency fluctuation within the experimental time of about 10 min. However, to ensure that our classifier accuracy is limited by systematic errors rather

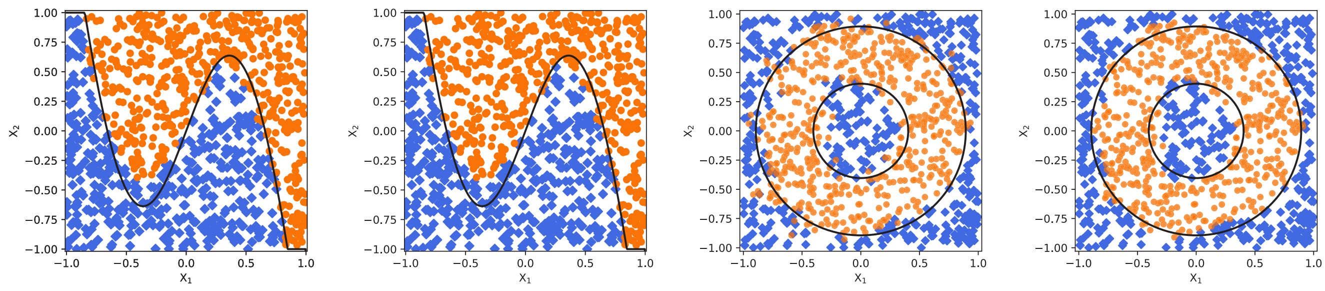

FIG. 9. Classifier test results for nonconvex and nontrivial topology problems. In each pair, the left plot corresponds to QPU and the right one to simulation. The ion-trap-based QPU classifier performed on 1000 random test data points is depicted by blue diamonds for points within and orange circles outside the boundary separating the circular feature shown as a solid line. The resulting *nonconvex* and *crown* datasets are computed using four layers, both from QPU and simulation. Notice that the border between classes in the experimental results is not as sharply defined as in the simulated classification. This difference is due to the uncertainty of the quantum measurements.

than by statistical ones, we measured the statistical error in the classification problem by repeating the experiment for each data point between 25 and 1000 times [see Fig. 8(a)]. The error (or inaccuracy) decreases from 12% to about 4% for 100 repetitions and then stays nearly the same, limited by systematic error. The fluctuation of the laser frequency with respect to the atomic resonance is captured over a time period of 20 min (twice the duration of an experiment) as plotted in Fig. 8(b). The random variation of the Rabi frequency over time is mostly caused by the magnetic field noise as we have separately measured the laser frequency drift to be $\leqslant 5$ kHz/24 h [35]. To minimize the impact of the residual magnetic field noise, we use electronic levels ($\Delta m = 0$) that are weakly sensitive to such noise. In addition, the detuning also indirectly influences the modified Rabi frequency. To check its influence, we varied the Rabi frequency, shown in Fig. 8(c), by varying the detuning within a 2 kHz range [as expected from Fig. 8(b)]. The result shows below 5% accuracy for the classifier when operating for 10 min.

(iv) *Laser intensity:* The Rabi frequency is fixed by setting the power and frequency of the laser at the start of the experiment. Any change in the Rabi frequency during the experiment, therefore, leads to error in the applied qubit rotation angle. Therefore, the accuracy of the classifier depends on the Rabi frequency fluctuations due to intensity fluctuation apart from detuning, as discussed earlier. The intensity is influenced by two factors: (a) laser power noise and (b) laser beam pointing error. In our experiment, the laser beam is tightly focused on the ion by a high numerical aperture (NA ~ 0.4) in-vacuum lens. To obtain high intensity at the ion position, the light is focused tightly to about 10 μm beam waist. To capture the influence of laser power variations on the classification error, we varied power (plotted in terms of modified Rabi frequency) in Fig. 8(d). Thus it is seen that the influence of intensity noise accounts to 5% error in accuracy. Thus, to avoid the influence of Rabi frequency fluctuation within the experimental time of $\sim$10 min, we reduce the Rabi frequency from 312 to 40 kHz such that the absolute error also reduces. This leads to an overall error of only 2% on the classifier output.

APPENDIX B: ADDITIONAL RESULTS

The results depicted in Fig. 9 correspond to additional binary classification problems. In both cases, classes are defined in such a way that each one fills half the total feature space. Notice that these classification problems are harder than the circle discussed above [44]. The nonconvexity feature

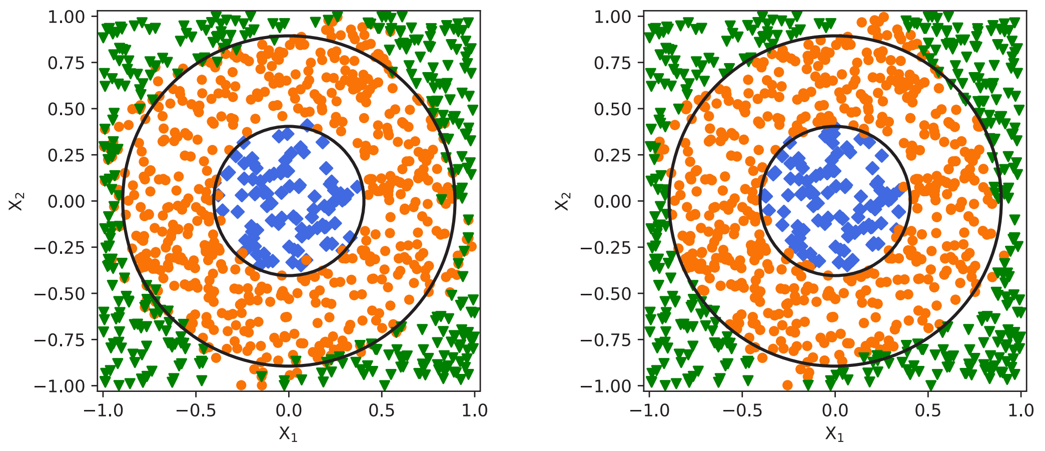

FIG. 10. Multiclass classification for the three-class *tricrown* problem on the QPU (left) and the simulation (right). Results include 1000 random data points; the classes are depicted with different colors and symbols, and the boundaries are defined by solid black lines. Notice that the border between classes in the experimental results is not as sharply defined as in the simulated classification. This difference is due to the uncertainty of the quantum measurements.

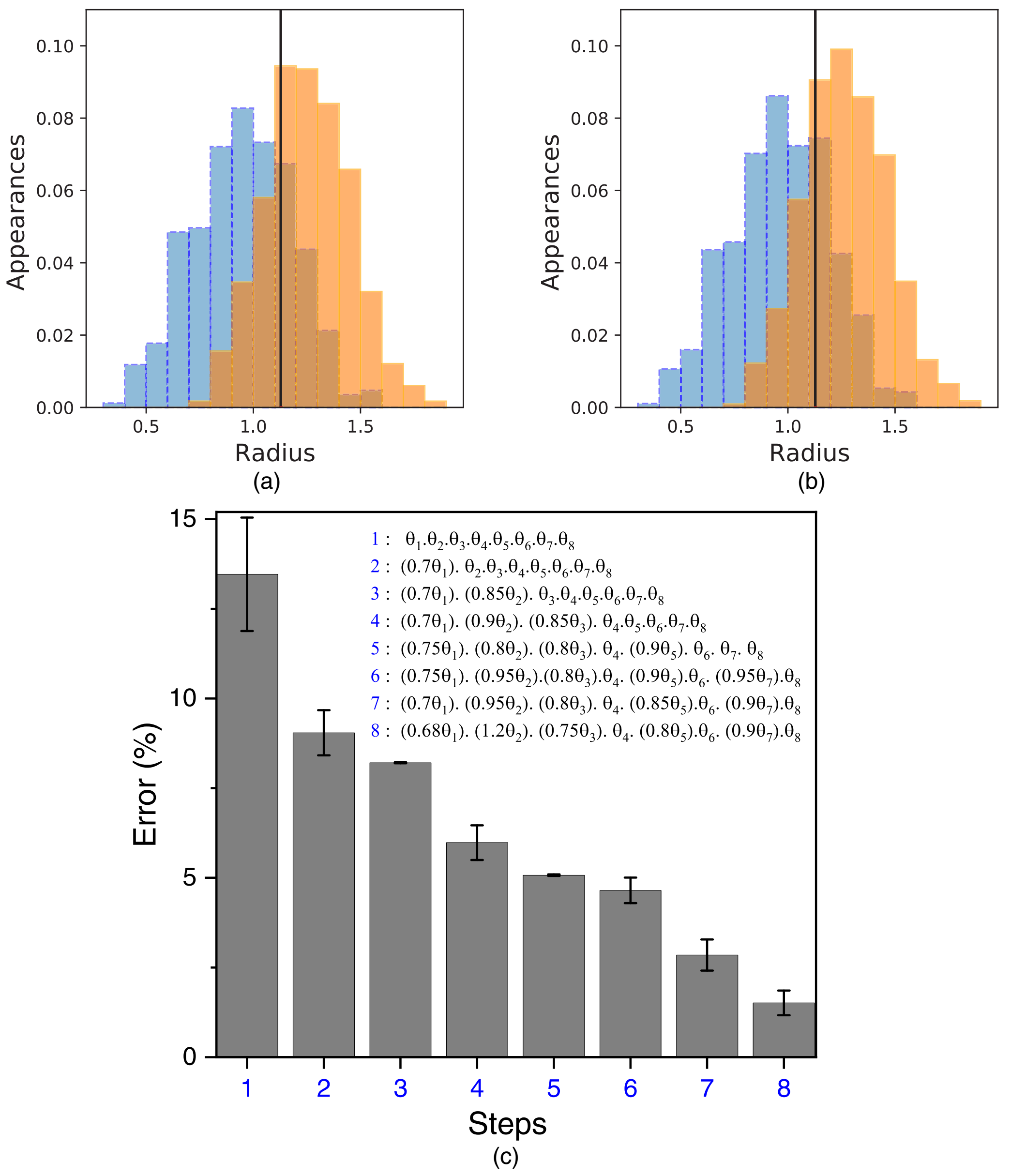

FIG. 11. Binary classification of the *hypersphere* dataset. The histograms in (a) and (b) represent the class association of points within a hypershell (given by the bin width) denoted by blue with a dashed outline (dark gray) in the left part (classified as within the hypersphere) and orange (light gray) in the right part (outside the hypersphere) for QPU (a) and simulation (b). The boundaries are defined by solid black lines. The overlap region (dark shaded area around the boundary lines) shows the ambiguity in classifying the points within a certain hyperradius. The accuracy of the QPU is improved by performing the experimental optimization near the vicinity of the simulated optima in a series of ten training steps. The reduction in the error with respect to the simulated results (b) is shown in (c).

of the *nonconvex* dataset presents a nontrivial challenge. In the *crown* case, the different classes are not only nonconvex but also the in-out class is disjoint. To solve the problem, the mapping performed by the circuit must be able to reflect this property. Nevertheless, the final result shows that all problems can be solved in the ion-trap QPU using the re-uploading scheme. The final accuracies obtained $(\mathcal{A}^{\text{sim}})/(\mathcal{A}^{*})$ are *nonconvex*: (92%)/(95%); *crown*: (87%)/(92%).

For the *tricrown* example depicted in Fig. 10, the difficulty is to add a third class to the *crown* problem. This, however, makes all regions in the feature space joint, but a nontrivial mapping is needed to transform the topology of the dataset to the target states in the Bloch sphere [32].

For Fig. 11 the classification results of the hypersphere are depicted both for QPU and simulation. In this case, only the radius and not the full description of the point is depicted for a graphical representation. Notice that the boundary lies partially outside the feature space, $r > 1$. This is due to geometrical reasons. For four and more dimensions, the n-ball with a volume half that of the $[-1, 1]^n$ hypercube does not fit into the hypercube itself. This changes the accuracy of the random classifier, but it is not taken into account in this analysis.

[1] E. Bernstein and U. Vazirani, Quantum complexity theory, SIAM J. Comput. **26**, 1411 (1997).

[2] P. W. Shor, Algorithms for quantum computation: discrete logarithms and factoring, in *Proceedings 35th Annual Symposium on Foundations of Computer Science* (IEEE, Piscataway, NJ, 1994), pp. 124–134

[3] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang, Experimental realization of shor's quantum factoring algorithm using nuclear magnetic resonance, Nature (London) **414**, 883 (2001).

[4] S. Debnath, N. M. Linke, C. Figgatt, K. A. Landsman, K. Wright, and C. Monroe, Demonstration of a small programmable quantum computer with atomic qubits, Nature (London) **536**, 63 (2016).

[5] D. Maslov, J.-S. Kim, S. Bravyi, T. J. Yoder, and S. Sheldon, Quantum advantage for computations with limited space, Nat. Phys. **17**, 894 (2021).

[6] F. Arute, K. Arya, J. M. Martinis *et al.*, Quantum supremacy using a programmable superconducting processor, Nature (London) **574**, 505 (2019).

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

[8] Y. Wu, W.-S. Bao, S. Cao, F. Chen, M.-C. Chen, X. Chen, T.-H. Chung, H. Deng, Y. Du, D. Fan *et al.*, Strong Quantum Computational Advantage Using a Superconducting Quantum Processor, Phys. Rev. Lett. **127**, 180501 (2021).

[9] C. Cortes and V. Vapnik, Support-vector networks, Mach. Learn. **20**, 273 (1995).

[10] *On-Line Learning in Neural Networks*, edited by D. Saad (Cambridge University Press, Cambridge, USA, 1999).

[11] C. M. Bishop, Neural networks and their applications, Rev. Sci. Instrum. **65**, 1803 (1994).

[12] S. Russell, *Artificial Intelligence: A Modern Approach* (Prentice Hall, Upper Saddle River, NJ, 2010).

[13] M. Mohri, *Foundations of Machine Learning* (MIT Press, Cambridge, MA, 2012).

[14] Y. Liu, S. Arunachalam, and K. Temme, A rigorous and robust quantum speed-up in supervised machine learning, Nat. Phys. **17**, 1013 (2021).

[15] V. Dunjko and H. J. Briegel, Machine learning & artificial intelligence in the quantum domain: a review of recent progress, Rep. Prog. Phys. **81**, 074001 (2018).

[16] M. Schuld, A. Bocharov, K. M. Svore, and N. Wiebe, Circuit-centric quantum classifiers, Phys. Rev. A **101**, 032308 (2020).

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

[18] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, W.-K. Mok, S. Sim, L.-C. Kwek, and A. Aspuru-Guzik, Noisy intermediate-scale quantum (NISQ) algorithms, Rev. Mod. Phys. **94**, 015004 (2022).

[19] J. Preskill, Quantum computing in the NISQ era and beyond, Quantum **2**, 79 (2018).

[20] X.-D. Cai, D. Wu, Z.-E. Su, M.-C. Chen, X.-L. Wang, L. Li, N.-L. Liu, C.-Y. Lu, and J.-W. Pan, Entanglement-Based Machine Learning on a Quantum Computer, Phys. Rev. Lett. **114**, 110504 (2015).

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

[22] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, Supervised learning with quantum-enhanced feature spaces, Nature (London) **567**, 209 (2019).

[23] C. Hempel, C. Maier, J. Romero, J. McClean, T. Monz, H. Shen, P. Jurcevic, Ben P. Lanyon, P. Love, R. Babbush, Alán Aspuru-Guzik, R. Blatt, and C. F. Roos, Quantum Chemistry Calculations on a Trapped-Ion Quantum Simulator, Phys. Rev. X **8**, 031022 (2018).

[24] S. Resch and U. R. Karpuzcu, Quantum computing: An overview across the system stack, arXiv:1905.07240 [quant-ph].

[25] C. Figgatt, D. Maslov, K. A. Landsman, N. M. Linke, S. Debnath, and C. Monroe, Complete 3-qubit grover search on a programmable quantum computer, Nat. Commun. **8**, 1918 (2017).

[26] Y. Nam, J.-S. Chen, J. Kim *et al.*, Ground-state energy estimation of the water molecule on a trapped-ion quantum computer, npj Quantum Inf. **6**, 33 (2020).

[27] S. Johri, S. Debnath, A. Mocherla, A. Singh, A. Prakash, J. Kim, and I. Kerenidis, Nearest centroid classification on a trapped ion quantum computer, npj Quantum Inf. **7**, 122 (2021).

[28] M. S. Rudolph, N. B. Toussaint, A. Katabarwa, S. Johri, B. Peropadre, and A. Perdomo-Ortiz, Generation of high-resolution handwritten digits with an ion-trap quantum computer, arXiv:2012.03924 [quant-ph].

[29] K. R. Brown, A. C. Wilson, Y. Colombe, C. Ospelkaus, A. M. Meier, E. Knill, D. Leibfried, and D. J. Wineland, Single-qubit-gate error below 10^{-4} in a trapped ion, Phys. Rev. A **84**, 030303(R) (2011).

[30] P. Wang, C.-Y. Luan, M. Qiao, M. Um, J. Zhang, Y. Wang, X. Yuan, M. Gu, J. Zhang, and K. Kim, Single ion qubit with estimated coherence time exceeding one hour, Nat. Commun. **12**, 233 (2021).

[31] A. Pérez-Salinas, D. López-Núñez, A. García-Sáez, P. Forn-Díaz, and J. I. Latorre, One qubit as a universal approximant, Phys. Rev. A **104**, 012405 (2021).

[32] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, and J. I. Latorre, Data re-uploading for a universal quantum classifier, Quantum **4**, 226 (2020).

[33] M. Schuld, R. Sweke, and J. J. Meyer, The effect of data encoding on the expressive power of variational quantum machine learning models, Phys. Rev. A **103**, 032430 (2021).

[34] P. Easom-Mccaldin, A. Bouridane, A. Belatreche, and R. Jiang, On depth, robustness and performance using the data re-uploading single-qubit classifier, IEEE Access **9**, 65127 (2021).

[35] D. Yum, D. D. Munshi, T. Dutta, and M. Mukherjee, Optical barium ion qubit, J. Opt. Soc. Am. B **34**, 1632 (2017).

[36] T. Dutta and M. Mukherjee, A single atom noise probe operating beyond the Heisenberg limit, npj Quantum Inf. **6**, 3 (2020).

[37] N. Van Horne, D. Yum, T. Dutta, P. Hänggi, J. Gong, D. Poletti, and M. Mukherjee, Single-atom energy-conversion device with a quantum load, npj Quantum Inf. **6**, 37 (2020).

[38] H. G. Dehmelt, Slow spin relaxation of optically polarized sodium atoms, Phys. Rev. **105**, 1487 (1957).

[39] N. Hansen, *The CMA Evolution Strategy: A Comparing Review* (Springer, 2006).

[40] R. Byrd, P. Lu, J. Nocedal, and C. Zhu, A limited memory algorithm for bound constrained optimization, SIAM J. Sci. Comput. **16**, 1190 (1995).

[41] D. C. McKay, C. J. Wood, S. Sheldon, J. M. Chow, and J. M. Gambetta, Efficient z gates for quantum computing, Phys. Rev. A **96**, 022330 (2017).

[42] D. Maslov, Basic circuit compilation techniques for an ion-trap quantum machine, New J. Phys. **19**, 023035 (2017).

[43] E. Knill, D. Leibfried, R. Reichle, J. Britton, R. B. Blakestad, J. D. Jost, C. Langer, R. Ozeri, S. Seidelin, and D. J. Wineland, Randomized benchmarking of quantum gates, Phys. Rev. A **77**, 012307 (2008).

[44] T. K. Ho and M. Basu, Measuring the complexity of classification problems, in *Proceedings of the 15th International Conference on Pattern Recognition, ICPR-2000* (IEEE, 2000), Vol. 2, pp. 43–47.

[45] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. **12**, 2825 (2011).

[46] J. P. Gaebler, T. R. Tan, Y. Lin, Y. Wan, R. Bowler, A. C. Keith, S. Glancy, K. Coakley, E. Knill, D. Leibfried, and D. J. Wineland, High-Fidelity Universal Gate Set for $^9\mathrm{Be}^+$ Ion Qubits, Phys. Rev. Lett. **117**, 060505 (2016).