

<https://doi.org/10.1038/s41534-025-01037-6>

Quantum ensemble learning with a programmable superconducting processor

Check for updates

Jiachen Chen^{1,5}, Yaozu Wu^{1,5}, Zhen Yang^{2,5}, Shibo Xu¹, Xuan Ye², Daili Li³, Ke Wang¹, Chuanyu Zhang¹, Feitong Jin¹, Xuhao Zhu¹, Yu Gao¹, Ziqi Tan¹, Zhengyi Cui¹, Aosai Zhang¹, Ning Wang¹, Yiren Zou¹, Tingting Li¹, Fanhao Shen¹, Jiarun Zhong¹, Zehang Bao¹, Zitian Zhu¹, Zixuan Song¹, Jinfeng Deng¹, Hang Dong¹, Pengfei Zhang¹, Wei Zhang³, Hekang Li¹, Qiujiang Guo¹, Zhen Wang¹, Ying Li⁴ ✉, Xiaoting Wang² ✉, Chao Song¹ ✉ & H. Wang¹

Quantum machine learning is among the most exciting potential applications of quantum computing. However, the vulnerability of quantum information to environmental noises and the consequent high cost for realizing fault tolerance has impeded the quantum models from learning complex datasets. Here, we introduce AdaBoost.Q, a quantum adaptation of the classical adaptive boosting (AdaBoost) algorithm designed to enhance learning capabilities of quantum classifiers. Based on the probabilistic nature of quantum measurement, the algorithm improves the prediction accuracy by refining the attention mechanism during the adaptive training and combination of quantum classifiers. We experimentally demonstrate the versatility of our approach on a programmable superconducting processor, where we observe notable performance enhancements across various quantum machine learning models, including quantum neural networks and quantum convolutional neural networks. With AdaBoost.Q, we achieve an accuracy above 86% for a ten-class classification task over 10,000 test samples, and an accuracy of 100% for a quantum feature recognition task over 1564 test samples. Our results demonstrate a foundational tool for advancing quantum machine learning towards practical applications, which has broad applicability to both the current noisy and the future fault-tolerant quantum devices.

The intersection between machine learning and quantum computing gives rise to the field of quantum machine learning that has attracted considerable attention. With the exponentially large Hilbert space, a quantum computer promises to offer representational power for recognizing complex data patterns that are challenging to recognize classically^{1–7}. Recently, the power of quantum learning models has been extensively studied in terms of quantum neural network (QNN) and quantum convolutional neural network (QCNN)^{8–14}, and, with the rapid advances of quantum technologies across various physical platforms^{15–19}, supported by a growing number of experiments^{20–24}. A crucial step towards practical applications is to test the quantum learning models on large-scale datasets. However, handling

extensive datasets presents significant challenges, necessitating the development of additional methods to enhance the performance of learning models. Ensemble learning is one such method, which leverages the power of joint decision and has achieved dramatic success in improving the accuracy of classical machine learning models over the last three decades^{25–29}. It is thus natural to contemplate the implementation of ensemble strategies in a quantum setting.

To date, research on quantum ensemble methods can be broadly divided into two categories depending on whether the base learners are implemented in a quantum superposition or not. A coherent superposition of base learners promises to speed up the ensemble learning procedure^{30–37},

¹School of Physics, ZJU-Hangzhou Global Scientific and Technological Innovation Center, and Zhejiang Key Laboratory of Micro-nano Quantum Chips and Quantum Control, Zhejiang University, 310027 Hangzhou, China. ²Institute of Fundamental and Frontier Sciences, University of Electronic Science and Technology of China, 610051 Chengdu, China. ³School of Physics and Key Laboratory of Quantum State Construction and Manipulation (Ministry of Education), Renmin University of China, 100872 Beijing, China. ⁴Graduate School of China Academy of Engineering Physics, 100193 Beijing, China. ⁵These authors contributed equally: Jiachen Chen, Yaozu Wu, Zhen Yang. ✉e-mail: yli@giscaep.ac.cn; xiaoting@uestc.edu.cn; chaosong@zju.edu.cn

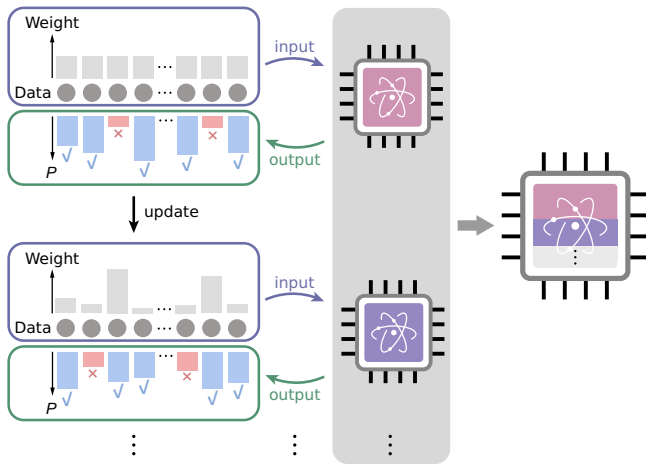


Fig. 1 | Schematic diagram of AdaBoost.Q. The algorithm is designed to generate a strong quantum classifier (right) by combining multiple weak quantum classifiers (middle). Each weak quantum classifier takes the same training data as an input, and it outputs the classification result of each data sample along with a probability P , which characterizes the confidence of the prediction. The weak quantum classifiers are trained iteratively, using reweighted versions of the training set, with the weights depending on the correctness of the predictions and finely tuned by the output probabilities of the previous classifier. This allows the subsequent quantum classifiers to focus on samples that were not well classified previously. The sample weights for the first classifier are assigned evenly among the training set.

which, however, is generally not feasible for near-term quantum devices, as it relies on subroutines that are only viable in fault-tolerant quantum computing regimes. On the other hand, classical ensemble methods have been explored to improve the performance of quantum classifiers from the perspective of error mitigation and resource savings^{38–40}, which can be well suited for current noisy quantum devices. However, directly applying classical ensemble techniques to quantum systems overlooks the unique properties of quantum classifiers. This naturally raises the following question: Can we harness the strengths of quantum classifiers to enhance ensemble methods while maintaining compatibility with the limitations of near-term quantum hardware?

In this work, we take a step forward by presenting a quantum version of the adaptive boosting (AdaBoost) ensemble algorithm, dubbed AdaBoost.Q. Our method uses the statistic information generated by quantum classifiers to improve the efficiency of AdaBoost algorithm by refining the attention mechanism during the adaptive training procedure. Using a superconducting quantum processor, we demonstrate the effectiveness of AdaBoost.Q in enhancing the performance of quantum models through two supervised learning experiments. In the first experiment, we perform a ten-class classification task on the MNIST handwritten digits dataset⁴¹ with a ten-qubit QNN classifier. By employing AdaBoost.Q, we improve the testing accuracy from 80% to above 86% over the full-size MNIST test dataset. Additionally, numerical simulations comparing AdaBoost.Q with the conventional AdaBoost.M1 algorithm⁴² (used in refs. 38,39) reveal the superior performance of our approach. The second experiment aims to classify three quantum phases of a spin chain model with a 15-qubit QCNN classifier. We show that the performance of the QCNN classifier can be significantly improved by AdaBoost.Q, with the testing accuracy enhanced from 77% to 100% over 1564 test samples. Our results provide a widely applicable method for pushing quantum machine learning towards practical applications.

Framework and experimental setup

It is a common human practice to aggregate and weigh different opinions to make a complex decision. The ensemble methodology extends this idea to the world of machine learning, aiming to construct a highly accurate classifier (referred to as a “strong” classifier) by combining multiple “weak”

classifiers, each of which may only slightly outperform a random guess. The AdaBoost algorithm is among the most prominent ensemble methods to generate a strong classifier^{25,26,28}. It works by training the weak classifiers sequentially on the same training set. At the core of the AdaBoost algorithm is an attention mechanism, where more attention is paid to data points that were previously misclassified when training the subsequent classifier. The level of attention paid is determined by a sample weight that is assigned to each training point. After the training procedure, each weak classifier is also assigned a weight for scoring its importance in forming the strong classifier. In the AdaBoost.Q algorithm, we extract these weights based on the probabilistic nature of the quantum classifiers.

We consider quantum classifiers that are built with parameterized quantum circuits. For a supervised K -class classification task, the training set consists of pairs of datasets, $\{x_i, y_i\}_{i=1}^N$, where x_i represents the data sample, y_i indexes the corresponding class label, and N is the training size. To classify the data samples, we select m qubits, with $m \geq \lceil \log_2 K \rceil$, from the quantum classifier and measure them in the computational basis, which can be described by a set of basis projectors $\{\Pi_j\}_{j=0}^{2^m-1}$ known as the projection-valued measures (PVM). We divide the PVM equally into K groups, with the k th group containing projectors indexed from $k \lfloor 2^m/K \rfloor$ to $(k+1) \lfloor 2^m/K \rfloor - 1$, while discarding the last $2^m - K \lfloor 2^m/K \rfloor$ projectors. The data sample is classified as the k th class if the measured state is located in the k th group.

According to Born’s rule, for an input sample x_i , the probability of measuring the basis state in the k th group is $P_k(x_i, \theta) = \sum_{j=k \lfloor 2^m/K \rfloor}^{(k+1) \lfloor 2^m/K \rfloor - 1} \text{Tr}(\rho(x_i, \theta) \Pi_j)$, where $\rho(x_i, \theta)$ denotes the reduced density matrix of the m measured qubits of the quantum classifier parameterized by θ . The corresponding predicted label of the quantum classifier is obtained as $\tilde{y}_i = \arg \max_k P_k(x_i, \theta)$. Note that the probability $P(x_i, \theta) = \max_k P_k(x_i, \theta)$ naturally characterizes the confidence of the prediction.

Using the probability information, we establish a refined criterion for calculating the weights of the data samples and classifiers following the spirit of the real AdaBoost algorithm^{26,43}. Specifically, we initialize all the sample weights to be $1/N$ when training the first classifier. During the iterative training of the subsequent classifiers, the sample weights are updated according to

$$w_{l+1,i} = \frac{w_{l,i}}{Z_{l+1}} \exp[P(x_i, \theta_l^*) \cdot (1 - 2\delta_{\tilde{y}_l, y_i})], \tag{1}$$

where i indexes the samples, l indexes the weak classifiers, θ_l^* is the optimal parameter for the l th classifier, δ denotes the Kronecker delta, and $Z_{l+1} = \sum_{i=1}^N w_{l,i} \exp[P(x_i, \theta_l^*) \cdot (1 - 2\delta_{\tilde{y}_l, y_i})]$ is the normalizing factor. The loss function of the l th classifier is given by $\mathcal{L}_l = \sum_{b=1}^B \mathcal{L}_{l,b}$, with B denoting the batch size and

$$\mathcal{L}_{l,b} = -w_{l,b} \ln [P_{y_b}(x_b, \theta)]. \tag{2}$$

After training the l th classifier, we calculate its weight as

$$\alpha_l = \ln \frac{c_l \mathcal{P}_l^{\text{true}}}{\mathcal{P}_l^{\text{false}}}, \tag{3}$$

where $\mathcal{P}_l^{\text{true}} = \sum_i w_{l,i} P(x_i, \theta_l^*) \delta_{\tilde{y}_l, y_i}$ and $\mathcal{P}_l^{\text{false}} = \sum_i w_{l,i} P(x_i, \theta_l^*) (1 - \delta_{\tilde{y}_l, y_i})$. The additional parameter c_l , which takes the value of unity by default, can be slightly tuned to optimize the training accuracy in practice. The ensemble classifier is constructed by combining all the trained weak classifiers (also referred to as base classifiers), which classifies x_i according to

$$\tilde{y}_i = \arg \max_k \sum_l \alpha_l P_k(x_i, \theta_l^*). \tag{4}$$

The workflow of AdaBoost.Q is illustrated in Fig. 1 and the pseudocode is summarized in Methods. In the conventional AdaBoost.M1 algorithm (see

Supplementary Sec. I), sample weights are determined exclusively by the correctness of the classification results—specifically, all misclassified samples have their weights increased uniformly. By incorporating classification probabilities as a measure of confidence into the weight calculation, our algorithm enables more precise reweighting of data samples. This refinement allows for a more nuanced adjustment of sample weights, which can enhance the overall performance of the ensemble model.

We experimentally demonstrate the effectiveness of our approach on a fully programmable superconducting quantum processor¹⁷. The qubits on the processor are of the frequency-tunable transmon type, which are arranged in an 11×11 square lattice, with the neighboring qubits connected by tunable couplers. For the ten-class classification of the MNIST dataset, we select 20 qubits to construct two copies of a ten-qubit QNN classifier, which run in parallel to accelerate training. For the quantum data classification task, we build a QCNN classifier with a carefully designed one-dimensional (1D) chain consisting of 15 qubits. All the classifiers are essentially variational quantum circuits compiled into the native gate sets, i.e., the parameterized single-qubit gates and two-qubit CZ gates between neighboring qubits. The median Pauli error rates of the parallel single- and two-qubit gates, characterized with the simultaneous cross-entropy benchmarking technique, are around 5×10^{-4} and 6×10^{-3} , respectively. See Supplementary Sec. IIA for details on device and gate performances.

Ensemble learning with QNN

As a first demonstration, we apply our method to improve the performance of QNN-based classifiers, which have been intensively studied in recent years^{10,44–46}. We focus on a ten-class classification task with the MNIST handwritten digit dataset, which is widely used in benchmarking machine learning models. This task had been challenging for quantum hardware, and it was not until recently that an experimental testing accuracy of around 62% over 500 test samples was reported²³. We construct the QNN with a shallow circuit consisting of ten qubits, which contains three layers of single-qubit gates to encode 30 trainable parameters and two layers of CNOT gates to entangle all qubits (see Supplementary Section IIC for details about the circuit design). The training dataset contains 3600 28×28 -pixel images (360 for each digit) selected from the MNIST dataset. To encode the classical data, we adapt the encoding scheme from the end-to-end learning framework⁴⁷. Specifically, we first vectorize the image data and then transform it to an array of rotational angles \mathbf{x} with a transform matrix W , following which we encode them into the single-qubit rotation of the QNN circuit alternatively

with the trainable parameters θ (Fig. 2a). Both W and θ are trained simultaneously to minimize the loss function during the learning procedure. To reduce the runtime, we further parallelize quantum computing by constructing and running two copies of the QNN simultaneously on the processor.

The training procedure of a single QNN classifier is exemplified in Fig. 2c, where the loss function converges to about 0.007 after 840 training steps. At the end of the training procedure, we input all 10,000 samples of the MNIST test dataset to the trained classifier, obtaining an overall testing accuracy around 80.5%, which is consistent with the training accuracy, verifying the generalizability of the trained QNN model. See Methods for details about the training procedure.

With the performance of the base QNN classifier established, we proceed to AdaBoost.Q. In Fig. 3a, we plot the testing accuracies of the ensemble QNN classifiers during the implementation of AdaBoost.Q. A notable increase of the accuracy is observed, from 80.5% to 86.7% over the first two iterations, after which the accuracy saturates. The observation is consistent with the weight calculated for each base classifier, which dramatically drops to near zero for the fourth base classifier (Fig. 3b). In Fig. 3c, we compare the classification results with and without using AdaBoost.Q, observing an improvement of testing accuracy for the ensemble classifier across almost all digits. To assess the intrinsic performance of AdaBoost.Q, we conduct numerical simulation to compare our method with the conventional AdaBoost.M1 algorithm⁴². The ensemble classifiers are constructed using four base classifiers, each employing the same QNN circuit structure as utilized in the experiment. We record the accuracy improvements of the ensemble classifier over a single classifier with randomly initialized parameters across 500 trials. As depicted in Fig. 3d, our method achieves an average accuracy improvement of 5%, aligning with the experimental results and surpassing AdaBoost.M1, which attains only a 1.6% average accuracy improvement. The results highlight the efficiency gains achieved through the refined attention mechanism of AdaBoost.Q. See Supplementary Section I for details of the numerical simulation and the AdaBoost.M1 algorithm.

Ensemble learning with QCNN

To demonstrate the versatility of AdaBoost.Q, we further apply it to QCNN-based classifiers. The task is to classify the ground states of a cluster-Ising Hamiltonian⁴⁸, which can either belong to a symmetry-protected topological (SPT) phase, a paramagnetic (PM) phase, or an Ising phase (See

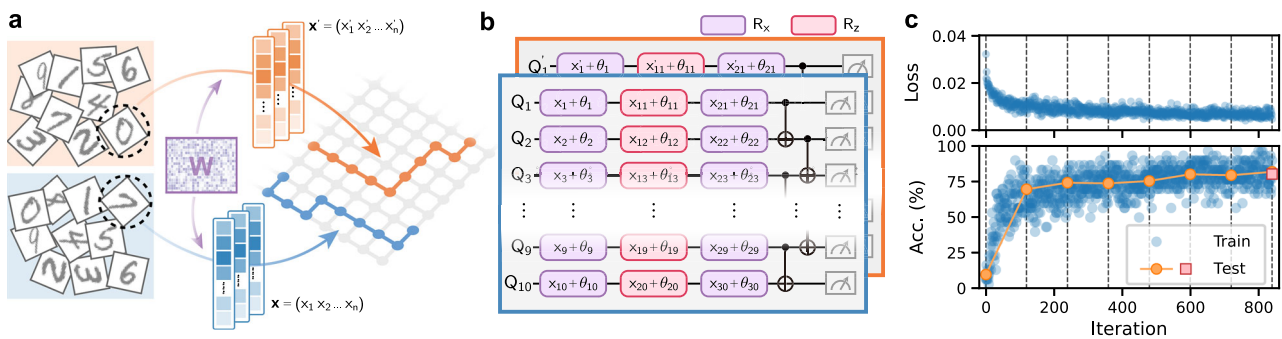
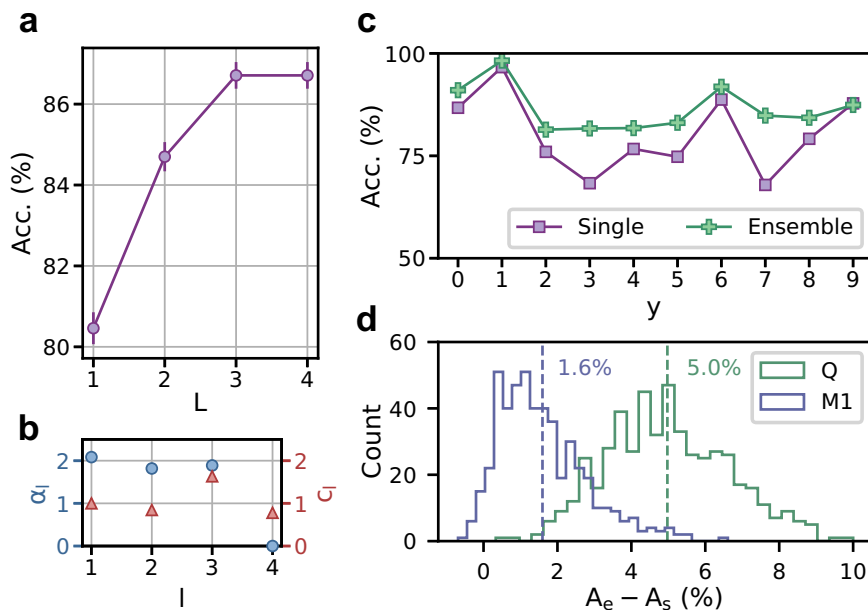


Fig. 2 | Ten-class classification of the MNIST dataset with QNN. **a** Experimental setup. The training set is composed of 3600 MNIST handwritten digit images, each with a size of 28×28 pixels. An image is transformed into a 30-dimensional vector \mathbf{x} using a trainable matrix W for further quantum encoding. At each training step, we select a batch of 30 images to train the classifier, which is split evenly into two sub-batches, denoted here by \mathbf{x} and \mathbf{x}' , and fed to two copies of the QNN in parallel. Each copy is constructed with a ten-qubit chain selected from the quantum processor. **b** The two QNN circuits, each of which is composed of three layers of single-qubit gates followed by two layers of CNOT gates. The rotation angles of single-qubit gates

are used to encode the data vector and trainable parameters θ . Here R_x and R_z denote the single-qubit rotation gates around the x - and z -axis, respectively. **c** Loss function (top) and accuracy for the test and training sets (bottom) at each training step. The training is carried out for 7 epochs, with each epoch consisting of 120 training steps as separated by the dashed lines. After each epoch, the QNN is monitored with 1000 images randomly selected from the MNIST test set (orange circle dots). At the end of the training, we measure the test accuracy over the whole MNIST test set containing 10,000 images (red square dot).

Fig. 3 | Experimental implementation of AdaBoost.Q. **a** Testing accuracy of the ensemble classifiers composed of L base classifiers, with $L = 1$ to 4. The error bars are obtained as the standard deviations from bootstrapping (resampled 1000 times from the original data sets). The accuracy is estimated based on the whole MNIST test set. **b** The experimental weight α_l and adjustment parameter c_l of each base classifier l . **c** Comparison of the classification performance on the whole MNIST test set with single and ensemble classifiers, where the accuracy of classifying each digit is displayed. **d** Comparison of the performance between the conventional AdaBoost.M1 and AdaBoost.Q with numerical simulation considering the same task, where both ensemble classifiers are composed of four base classifiers. The simulation is performed 500 times each. The histogram of the accuracy improvement, $A_e - A_s$ with A_e and A_s being the testing accuracy of the ensemble and single classifiers, is plotted, with the average values denoted by the dashed lines.



Methods). Previous studies^{12,21} have revealed the advantages of QCNN over the direct measurement of the string order parameter in identifying the SPT phase. Here, we consider a more complex task, i.e., the classification of all three phases.

The QCNN circuit is implemented on a 1D chain of 15 qubits on our device. QCNN can achieve an exponential reduction of trainable parameters compared with the generic QNN circuit¹², at the cost of involving long-range two-qubit gates along the chain. We circumvent this challenge by carefully designing the topology of the selected 1D chain, such that all necessary two-qubit gates can be directly implemented, as shown in Fig. 4a. A QCNN classifier is typically composed of convolutional, pooling, and fully connected layers. In our experiment, we construct the convolutional layer with two layers of convolutional kernels applied in a translationally invariant way. Each convolutional kernel is a parameterized two-qubit $U(4)$ unitary⁴⁹. A pooling layer applies parallel controlled-X (CNOT) gates and then discards the control qubits. After performing the convolutional and pooling layers for three rounds, we implement a CZ gate on the remaining two qubits as the fully connected layer. The two qubits are then readout to obtain the classification results.

We use variational quantum circuits to prepare approximate ground states for constructing the training and test datasets. The quantum data can be directly input into the QCNN circuit as initial states. Within the reach of variational quantum circuits, our training (test) dataset consists of 2204 (1564) points spanning the phase diagram of the cluster-Ising model, with the experimentally measured string order parameters shown in Fig. 4b. To examine the generalization of the QCNN classifier, we extend to the parameter regime in generating the test samples (red box in Fig. 4b). See Methods and Supplementary Section IIB for more details about the generation of training and test sets.

In Fig. 4c–e, we show the performance of AdaBoost.Q. For a single QCNN classifier, we find that the testing accuracy for the three-class classification task is limited to around 80%. A detailed analysis of the classification results reveals that the states in the SPT phase are most likely to be misclassified (Fig. 4e, top panel). By tuning the weights of the training states (Fig. 4c), the second classifier achieves a remarkable increase in the testing accuracy, reaching a value of 93%, with most of the misclassified samples

located in the Ising phase (Fig. 4e, middle panel). Finally, by combining only two base classifiers, the ensemble classifier achieves a testing accuracy of 100% on the whole test set (Fig. 4e, bottom panel), demonstrating the high efficiency of our algorithm.

Conclusion and discussion

We utilize the statistical property of quantum classifiers to enhance the performance of the AdaBoost algorithm by refining its core weight-update rule. We validate the efficacy of our approach through a prototypical ten-class classification task on the MNIST handwritten digit dataset, observing a significant improvement in testing accuracy. To the best of our knowledge, this represents the first experimental benchmark of a quantum classifier on the full-size MNIST test set, although the achieved testing accuracy of 86.7% remains modest compared to the classical benchmark of approximately 99%. To attain competitive accuracy, further advancements in quantum learning models—particularly in terms of system size and circuit structure—are necessary. In this context, AdaBoost.Q can alleviate the accuracy requirements for individual quantum classifiers. The resource efficiency of AdaBoost.Q compared with improving the performance of a single classifier is discussed in Supplementary Section IID.

On the other hand, quantum learning models operating on quantum data present a more promising route towards achieving a quantum advantage^{8–13}. Our method also demonstrates its utility in this direction, as evidenced by the results of the second task. While our results showcase the promise of this approach, we note that a comprehensive exploration of quantum advantage in quantum machine learning constitutes a distinct and significant research direction. A definitive demonstration of such advantage will likely require further advances in both algorithmic design and hardware capabilities, building upon the foundational insights provided by studies like those in ref. 6–10,20.

A caveat regarding the future use of AdaBoost.Q is its susceptibility to overfitting, particularly in the presence of noisy data, which is commonly observed within the AdaBoost algorithm family due to their iterative nature. To mitigate such risk, strategies such as early stopping⁵⁰, learning rate adjustment⁵¹, and data cleaning⁵² can be applied. For example, one can monitor the performance on a validation

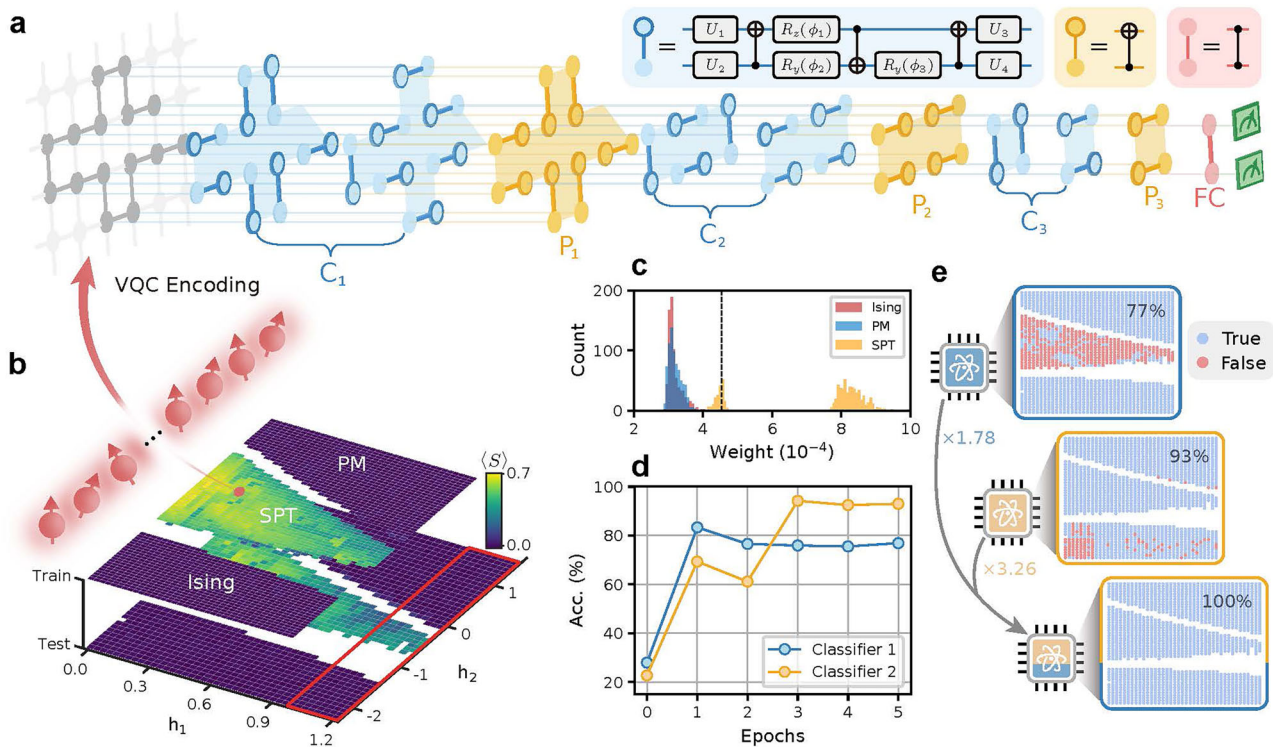


Fig. 4 | Ensemble learning of quantum data with QCNN. **a** Structure of the QCNN circuit, which consists of three alternating convolutional (C_i) and pooling (P_i) layers, followed by a fully-connected (FC) layer. Each convolutional layer contains two layers of convolutional kernels, each of which consists of seven single-qubit gates with 15 variational parameters and three two-qubit CNOT gates. We apply the convolutional kernels in a translationally invariant way, resulting in a total of 45 variational parameters. The pooling layer applies a layer of two-qubit CNOT gates and then passes the target qubits to the subsequent layer, thus reducing the qubit number by half. After three rounds of pooling, we are left with two qubits, on which we apply a CZ gate as a fully connected layer. The two qubits are then measured in the computational basis for the classification task. **b** The experimentally measured string order parameters (S) for the quantum states in the training and test sets, respectively. Each quantum state is the ground state of a cluster-Ising Hamiltonian with 15 spins,

which has two parameters h_1 and h_2 (see Methods). The states can be approximately prepared with variational quantum circuits and directly input into the QCNN classifier. The test set is generated in a parameter regime larger than that for the training set, and we highlight their difference with the red box. **c** Histogram of the sample weights for training the second QCNN base classifiers. The weights for the quantum states in different phases are plotted separately. The dashed line represents the initial weight that is assigned to all data samples when training the first base classifier. **d** Testing accuracy of the first (blue) and the second (yellow) base classifier at each epoch. **e** Classification results of the first (top), second (middle), and ensemble (bottom) classifiers on the test set. The classifier weights are 1.78 and 3.26, respectively.

set and stop training when the validation error starts to increase, even if the training error continues to decrease. This prevents the model from over-adapting to noise. Our approach is broadly compatible with various quantum learning models and can be implemented across different physical platforms, both in the current noisy intermediate-scale quantum era and the coming fault-tolerant quantum era, which we anticipate would benefit the future exploration of quantum learning advantages.

Methods

Data generation

The training and test sets used for the ten-class classification task originate from the MNIST handwritten digits dataset. To accelerate the training procedure, we use the k -means algorithm⁵³ to select 3600 representative images from the MNIST training dataset to form the training set. Specifically, we use KMeans function of the scikit-learn package⁵⁴ to divide each number into 360 clusters. We select one image from each cluster to form ten numbers totalling 3600 images. During the clustering process of KMeans, we perform 180 random initial centroids and select the case the clustering with the best inertia. The test set is constructed with all 10,000 images from the MNIST test dataset. During the training procedure, we also randomly select 1000 images, with 100 for each digit, from the test set to monitor the testing accuracy.

For the quantum phase recognition task, the quantum dataset consists of ground states of the cluster-Ising Hamiltonian:

$$H = - \sum_{i=1}^{N-2} Z_i X_{i+1} Z_{i+2} - h_1 \sum_{i=1}^N X_i - h_2 \sum_{i=1}^{N-1} X_i X_{i+1}, \quad (5)$$

where $N = 15$ in our system. $\{X_i, Z_i\}$ are the Pauli operators acting on the i th spin. Depending on the choice of the two model parameters $\{h_1, h_2\}$, the ground states can belong to either the SPT phase, the PM phase, or the Ising phase. The SPT phase can be distinguished by measuring the string order parameter:

$$S = Z_1 X_2 X_4 \dots X_{12} X_{14} Z_{15}, \quad (6)$$

as shown in Fig. 4b. To construct the training (test) set, we select 2, 204 (1, 564) ground states in the parameter regimes of $h_1 \in [0, 1)$ ($h_1 \in [0, 1.2)$) and $h_2 \in (-2.3, 1.6]$, respectively. For a given $\{h_1, h_2\}$, we use a variational quantum circuit (VQC) to prepare the corresponding ground state. The circuit is trained on a classical computer before being deployed on the quantum processor (See Supplementary Section IIB for details on the training procedure).

Algorithms for quantum ensemble learning

The AdaBoost.Q algorithm proposed in this work is shown in Algorithm 1.

Algorithm 1. AdaBoost.Q

Input:

Training set $\mathcal{T} = \{(x_i, y_i)\}_{i=1}^N$, where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y} \subseteq \{1, 2, \dots, K\}$;
 The number of weak classifier L ;
 A series of weak classifiers $\{G(x, \theta_l)\}_{l=1}^L$,

$$G(x, \theta_l) : \mathcal{X} \rightarrow \mathbf{R}^K, \quad G(x_i, \theta_l) = \{P_k(x_i, \theta_l)\}_{k=1}^K,$$

with $P_k(x_i, \theta_l)$ defined in the main text.

Output:

Combined classifier

$$\mathcal{G}(x) = \sum_{l=1}^L \alpha_l \cdot G(x, \theta_l^*),$$

where α_l and θ_l^* are the weight and optimal parameters of the l th classifier. The output label of the combined classifier is

$$\tilde{y}_i = \arg \max_k \mathcal{G}(x_i).$$

Initialize the weights of the training set,

$$\mathbf{w}_1 = \{w_{1,i}\}_{i=1}^N,$$

where $w_{1,i} = 1/N$.

for $l = 1 : L$ **do**

Train the classifier $G(x, \theta)$ with the weighted training set. Obtain

$$\tilde{y}_{l,i} = \arg \max_k G(x_i, \theta_l^*),$$

with probability:

$$P(x_i, \theta_l^*) = \max_k G(x_i, \theta_l^*).$$

Calculate the l th classifier's weight:

$$\alpha_l = \ln \frac{c_l \mathcal{P}_l^{\text{true}}}{\mathcal{P}_l^{\text{false}}},$$

with

$$c_l = \begin{cases} 1 & l = 1 \\ \arg \max_c \sum_i \delta(\tilde{y}_i(\alpha_l(c)), y_i) & l \neq 1. \end{cases}$$

Here $\tilde{y}_i(\alpha_l(c))$ is the prediction of the ensemble classifier composed of the first l classifiers. $\mathcal{P}_l^{\text{true}}$ and $\mathcal{P}_l^{\text{false}}$ are defined in the main text.

Update the weight of training set:

$$w_{l+1,i} = \frac{w_{l,i}}{Z_{l+1}} \exp[P(x_i, \theta_l^*) \cdot (1 - 2\delta_{\tilde{y}_{l,i}, y_i})],$$

with the normalization factor

$$Z_{l+1} = \sum_{i=1}^N w_{l,i} \exp[P(x_i, \theta_l^*) \cdot (1 - 2\delta_{\tilde{y}_{l,i}, y_i})].$$

end

Training the QNN classifier

We train the QNN classifier with epochs. At each epoch, we first shuffle the training set and then divide it evenly into 120 batches, with each batch

containing 30 images. During the training procedure, we iterate through the 120 batches of dataset for each epoch, and train the QNN classifier for 7 epochs. To encode an image sample x_p , we first flatten it into a 784-dimensional vector \mathbf{v}_p , which is then divided by 255^2 and transformed into a 30-dimensional vector \mathbf{x}_i by a trainable matrix \mathbf{W} . The data vector \mathbf{x}_i is encoded together with the trainable parameter θ into the QNN circuit, as shown in Fig. 2b. We initialize θ and \mathbf{W} by randomly sampling each of their element from the Gaussian distribution $\mathcal{N}(\pi, (\frac{\pi}{3})^2)$ and $\mathcal{N}(\frac{\pi}{60}, (\frac{\pi}{180})^2)$, respectively. \mathbf{W} and θ are trained by using the gradient based Adam optimizer⁵⁵, with the gradient experimentally measured by using the parameter shift rule⁵⁶. The updating rules for \mathbf{W} and θ are detailed in Algorithm 2.

Algorithm 2. Updating rules for \mathbf{W} and θ .

Input:

Trainable matrix \mathbf{W}_j ;
 Trainable parameter θ_j ;
 Batch of training data $\{(\mathbf{v}_b, y_b)\}_{b=1}^B$;
 Batch of corresponding sample weight $\mathbf{w} = \{w_b\}_{b=1}^B$.

Output:

Updated trainable matrix \mathbf{W}_{j+1} ;
 Updated trainable parameter θ_{j+1} .

$\mathbf{g}\theta_j, \mathbf{g}\mathbf{w}_j \leftarrow \mathbf{0}$

for $b = 1 : B$ **do**

Obtain the encoding data $\mathbf{x}_b = \mathbf{W}_j \cdot \mathbf{v}_b$. Measure the gradient

$$\begin{aligned} \nabla_{\mathbf{x}_b} \mathcal{L}_b &= \nabla_{\theta_j} \mathcal{L}_b = \left[\frac{\partial \mathcal{L}_b}{\partial \theta_{j,i}} \right]_{i=1}^{N_t} \\ &= -\frac{w_b}{P_{y_b}(\mathbf{x}_b, \theta_j)} \left[\frac{\partial P_{y_b}}{\partial \theta_{j,i}} \right]_{i=1}^{N_t} \\ &= -\frac{w_b}{P_{y_b}(\mathbf{x}_b, \theta_j)} \left[\frac{P_{y_b}(\mathbf{x}_b, \theta_{j,i}^+) - P_{y_b}(\mathbf{x}_b, \theta_{j,i}^-)}{2} \right]_{i=1}^{N_t}, \end{aligned}$$

where N_t is the number of trainable parameters and $\theta_{j,i}^{\pm} = [\theta_{j,1}; \theta_{j,2}; \dots; \theta_{j,i} \pm \pi/2; \dots; \theta_{j,N_t}]$. Calculate the gradient of transform matrix:

$$\nabla_{\mathbf{W}_j} \mathcal{L}_b = \nabla_{\mathbf{x}_b} \mathcal{L}_b \cdot \mathbf{v}_b^T.$$

$$\begin{aligned} \mathbf{g}\theta_j &= \mathbf{g}\theta_j + \nabla_{\theta_j} \mathcal{L}_b. \\ \mathbf{g}\mathbf{w}_j &= \mathbf{g}\mathbf{w}_j + \nabla_{\mathbf{W}_j} \mathcal{L}_b. \end{aligned}$$

end

Calculate:

$$\theta_{j+1} = \theta_j + \mathcal{A}[\beta_1, \beta_2, \gamma, j, \mathbf{g}\theta_j],$$

$$\mathbf{W}_{j+1} = \mathbf{W}_j + \mathcal{A}[\beta_1, \beta_2, \gamma, j, \mathbf{g}\mathbf{w}_j],$$

where \mathcal{A} is the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\gamma = 0.5$.

Training the QCNN classifier

The training procedure of the QCNN classifier is similar to that of the QNN classifier. Each QCNN classifier is trained for 5 epochs. At each epoch, the training set is shuffled and divided into 76 batches, with each batch containing 29 quantum states. The quantum states can be directly input into the QCNN classifier. The trainable parameter θ is initialized by randomly generating each of its element in a range of $[-\pi, \pi]$. The gradient of the loss function with respect to θ is evaluated by using the finite difference method, with the updating rules detailed in Algorithm 3. The robustness of the finite difference method used here is illustrated in Supplementary Section IIE.

Algorithm 3. Updating rules for θ

Input:

weights $\mathbf{w} = \{w_b\}_{b=1}^B$; Trainable parameter θ_j ;
 Batch of training data $\{(x_b, y_b)\}_{b=1}^B$;
 Batch of corresponding sample weight $\mathbf{w} = \{w_b\}_{b=1}^B$.

Output:

Updated trainable parameter θ_{j+1} .

$g_{\theta_j} \leftarrow \mathbf{0}$

for $b = 1 : B$ do

 Measure the gradient

$$\begin{aligned} \nabla_{\theta_j} \mathcal{L}_b &= \left[\frac{\partial \mathcal{L}_b}{\partial \theta_{j,i}} \right]_{i=1}^{N_t} \\ &= -w_b \left[\frac{\partial \ln(P_{y_b})}{\partial \theta_{j,i}} \right]_{i=1}^{N_t} \\ &= -w_b \left[\frac{\ln(P_{y_b}(x_b, \theta_{j,i}^+)) - \ln(P_{y_b}(x_b, \theta_{j,i}^-))}{2\epsilon} \right]_{i=1}^{N_t}, \end{aligned}$$

 where N_t is the number of trainable parameters,
 $\theta_{j,i}^\pm = [\theta_{j,1}; \theta_{j,2}; \dots; \theta_{j,i} \pm \epsilon; \dots; \theta_{j,N_t}]$, and $\epsilon = 0.2$.
 $g_{\theta_j} = g_{\theta_j} + \nabla_{\theta_j} \mathcal{L}_b$.

end

Calculate:

$$\theta_{j+1} = \theta_j + \mathcal{A}[\beta_1, \beta_2, \gamma, j, g_{\theta_j}],$$

where \mathcal{A} is the Adam optimizer with $\beta_1 = 0.9$,
 $\beta_2 = 0.999$, and $\gamma = 0.02$ is the learning rate.

Data availability

The data presented in the figures and that support the other findings of this study are available on Github: <https://github.com/wuyaoz/Data-for-Ensemble-learning>.

Received: 31 January 2025; Accepted: 2 May 2025;
 Published online: 21 May 2025

References

1. Stoudenmire, E. M. & Schwab, D. J. Supervised learning with tensor networks. In: *Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS'16* 4806–4814 (Curran Associates Inc., Red Hook, NY, USA, 2016).
2. Biamonte, J. et al. Quantum machine learning. *Nature* **549**, 195 (2017).
3. Dunjko, V. & Briegel, H. J. Machine learning & artificial intelligence in the quantum domain: a review of recent progress. *Rep. Prog. Phys.* **81**, 074001 (2018).
4. Das Sarma, S., Deng, D.-L. & Duan, L.-M. Machine learning meets quantum physics. *Phys. Today* **72**, 48 (2019).
5. Cerezo, M., Verdon, G., Huang, H.-Y., Cincio, L. & Coles, P. J. Challenges and opportunities in quantum machine learning. *Nat. Comput. Sci.* **2**, 567 (2022).
6. Havlíček, V. et al. Supervised learning with quantum-enhanced feature spaces. *Nature* **567**, 209 (2019).
7. Liu, Y., Arunachalam, S. & Temme, K. A rigorous and robust quantum speed-up in supervised machine learning. *Nat. Phys.* **17**, 1013 (2021).
8. Dilip, R., Liu, Y.-J., Smith, A. & Pollmann, F. Data compression for quantum machine learning. *Phys. Rev. Res.* **4**, 043007 (2022).
9. Huang, H.-Y. et al. Power of data in quantum machine learning. *Nat. Commun.* **12**, 2631 (2021).
10. Jerbi, S. et al. Quantum machine learning beyond kernel methods. *Nat. Commun.* **14**, 517 (2023).
11. Grant, E. et al. Hierarchical quantum classifiers. *npj Quantum Inf.* **4**, 65 (2018).
12. Cong, I., Choi, S. & Lukin, M. D. Quantum convolutional neural networks. *Nat. Phys.* **15**, 1273 (2019).
13. Liu, Y.-J., Smith, A., Knap, M. & Pollmann, F. Model-independent learning of quantum phases of matter with quantum convolutional neural networks. *Phys. Rev. Lett.* **130**, 220603 (2023).
14. Tolstobrov A, et al. Hybrid quantum learning with data reuploading on a small-scale superconducting quantum simulator. *Phys. Rev. A.* **109**, 012411 (2024).
15. Arute, F., Arya, K. & Babbush, R. et al. Quantum supremacy using a programmable superconducting processor. *Nature* **574**, 505 (2019).
16. Kim, Y. et al. Evidence for the utility of quantum computing before fault tolerance. *Nature* **618**, 500 (2023).
17. Xu, S., Sun, Z.-Z. & Wang, K. et al. Digital simulation of projective non-abelian anyons with 68 superconducting qubits. *Chin. Phys. Lett.* **40**, 060301 (2023).
18. Bluvstein, D. et al. Logical quantum processor based on reconfigurable atom arrays. *Nature* **626**, 58 (2024).
19. Iqbal, M. et al. Non-abelian topological order and anyons on a trapped-ion processor. *Nature* **626**, 505 (2024).
20. Huang, H.-Y. et al. Quantum advantage in learning from experiments. *Science* **376**, 1182 (2022).
21. Herrmann, J. et al. Realizing quantum convolutional neural networks on a superconducting quantum processor to recognize quantum phases. *Nat. Commun.* **13**, 4144 (2022).
22. Gong, M. et al. Quantum neuronal sensing of quantum many-body states on a 61-qubit programmable superconducting processor. *Sci. Bull.* **68**, 906 (2023).
23. Kornjača, M. et al. Large-scale quantum reservoir learning with an analog quantum computer. Preprint at <https://arxiv.org/abs/2407.02553> (2024).
24. Zhang, C. et al. Quantum continual learning on a programmable superconducting processor. Preprint at <https://arxiv.org/abs/2409.09729> (2024).
25. Freund, Y. & Schapire, R. E. in *Computational Learning Theory* (ed Vitányi, P.) 23–37 (Springer Berlin Heidelberg, 1995) <https://doi.org/10.1006/jcss.1997.1504>.
26. Schapire, R. E. & Singer, Y. Improved boosting algorithms using confidence-rated predictions. *Mach. Learn.* **37**, 297 (1999).
27. Hastie, T., Rosset, S., Zhu, J. & Zou, H. Multi-class adaboost. *Stat. Interface* **2**, 349 (2009).
28. Sagi, O. & Rokach, L. Ensemble learning: a survey. *WIREs Data Min. Knowl. Discov.* **8**, e1249 (2018).
29. Viola, P. & Jones, M. J. Robust real-time face detection. *Int. J. Comput. Vis.* **57**, 137 (2004).
30. Schuld, M. & Petruccione, F. Quantum ensembles of quantum classifiers. *Sci. Rep.* **8**, 2772 (2018).
31. Abbas, A., Schuld, M. & Petruccione, F. On quantum ensembles of quantum classifiers. *Quantum Mach. Intell.* **2**, 6 (2020).
32. Arunachalam, S. & Maity, R. *Quantum Boosting* (JMLR.org, 2020).
33. Wang, X., Ma, Y., Hsieh, M.-H. & Yung, M.-H. Quantum speedup in adaptive boosting of binary classification. *Sci. China Phys. Mech. Astron.* **64**, 220311 (2020).
34. Leal, D., De Lima, T. & Da Silva, A. J. Training ensembles of quantum binary neural networks. In: *2021 International Joint Conference on Neural Networks (IJCNN)* 1–6. (2021). <https://ieeexplore.ieee.org/document/9534253>.
35. Ohno, H. Boosting for quantum weak learners. *Quantum Inform. Process.* **21**, 199 (2022).
36. Macaluso, A., Clissa, L., Lodi, S. & Sartori, C. An efficient quantum algorithm for ensemble classification using bagging. *IET Quantum Commun.* **5**, 253 (2024).
37. Yu, Z., Chen, Z. & Xue, C. Quantum ensemble learning with quantum support vector machine. In: *2024 7th International Conference on Advanced Algorithms and Control Engineering (ICAACE)* 1359–1363. <https://doi.org/10.1109/ICAACE61206.2024.10548133> (2024).

38. Li, Q. et al. Ensemble-learning error mitigation for variational quantum shallow-circuit classifiers. *Phys. Rev. Res.* **6**, 013027 (2024).
39. Wang, Y., Wang, X., Qi, B. & Dong, D. Supervised-learning guarantee for quantum adaboost. *Phys. Rev. Appl.* **22**, 054001 (2024).
40. Incudini, M. et al. Resource saving via ensemble techniques for quantum neural networks. *Quantum Mach. Intell.* **5**, 39 (2023).
41. Deng, L. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Process. Mag.* **29**, 141 (2012).
42. Freund, Y. & Schapire, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Computer Syst. Sci.* **55**, 119 (1997).
43. Schapire, R. E. The boosting approach to machine learning: An overview, https://doi.org/10.1007/978-0-387-21579-2_9 *Nonlinear Estimation and Classification*, 149 (2003).
44. Schuld, M., Bocharov, A., Svore, K. M. & Wiebe, N. Circuit-centric quantum classifiers. *Phys. Rev. A* **101**, 032308 (2020).
45. Ren, W. et al. Experimental quantum adversarial learning with programmable superconducting qubits. *Nat. Comput. Sci.* **2**, 711 (2022).
46. Li, W., Lu, Z. & Deng, D.-L. Quantum neural network classifiers: a tutorial. *SciPost Phys. Lect. Notes*, **61** <https://doi.org/10.21468/SciPostPhysLectNotes.61> (2022).
47. Pan, X. et al. Experimental quantum end-to-end learning on a superconducting processor. *npj Quantum Inf.* **9**, 18 (2023).
48. Smacchia, P. et al. Statistical mechanics of the cluster ising model. *Phys. Rev. A* **84**, 022304 (2011).
49. Vatan, F. & Williams, C. Optimal quantum circuits for general two-qubit gates. *Phys. Rev. A* **69**, 032315 (2004).
50. Zhang, T. & Yu, B. Boosting with early stopping: convergence and consistency. *Ann. Stat.* **33**, 1538 (2005).
51. Bühlmann, P. & Yu, B. Boosting with the l2 loss. *J. Am. Stat. Assoc.* **98**, 324 (2003).
52. Côté, P.-O., Nikanjam, A., Ahmed, N., Humeniuk, D. & Khomh, F. Data cleaning and machine learning: a systematic literature review. *Automated Softw. Eng.* **31**, 54 (2024).
53. MacQueen, J. Some methods for classification and analysis of multivariate observations. In: *Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability/University of California Press* (eds Le Cam, L. M. & Neyman, J.) 281–297 (University of California Press, 1967).
54. Buitinck, L. et al. API design for machine learning software: experiences from the scikit-learn project. In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning* (eds Bruno, C. Luc, D. R. Paolo, F. & Tias, G.) 108–122, Prague, Czech Republic (Springer, 2013).
55. Kingma, D. P. & Ba, J. Adam: a method for stochastic optimization. Proceedings of the 3rd International Conference on Learning Representations (ICLR). arXiv preprint arXiv:1412.6980. (2015).
56. Mitarai, K., Negoro, M., Kitagawa, M. & Fujii, K. Quantum circuit learning. *Phys. Rev. A* **98**, 032309 (2018).
- China (Grant No. 2023YFB4502600), the National Natural Science Foundation of China (Grant Nos. 12174342, 92365301, 12274367, 12322414, 12274368, 12225507, 12088101, 12074428 and 92265208), and the Zhejiang Provincial Natural Science Foundation of China (Grant Nos. LDQ23A040001, LR24A040002). Y.L. is also supported by NSAF (Grant No. U1930403).

Author contributions

X.W. and Y.L. initiated the study. J.C. and Y.W. carried out the experiments under the supervision of C.S., J.C. and X.Z. designed the device and H.L. fabricated the device, supervised by H.W., J.C., Z.Y., Y.L., and D.L. performed the numerical simulation, supervised by W.Z., X.W., and C.S.. J.C., Z.Y., X.Y., D.L., Y.L., and X.W. conducted the theoretical analysis. C.S., Y.W., J.C., and Y.L. co-wrote the manuscript. J.C., Y.W., S.X., K.W., C.Z., F.J., X.Z., Y.G., Z.T., Z.C., A.Z., N.W., Y.Z., T.L., F.S., J.Z., Z.B., Z.Z., Z.S., J.D., H.D., P.Z., H.L., Q.G., Z.W., C.S., and H.W. contributed to the experimental setup. All authors contributed to the analysis of data, the discussions of the results, and the writing of the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at <https://doi.org/10.1038/s41534-025-01037-6>.

Correspondence and requests for materials should be addressed to Ying Li, Xiaoting Wang or Chao Song.

Reprints and permissions information is available at <http://www.nature.com/reprints>

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

© The Author(s) 2025

Acknowledgement

The device was fabricated at the Micro-Nano Fabrication Center of Zhejiang University. We acknowledge support from the National Key R&D Program of