

Quantum-inspired weight-constrained neural network: Reducing variable numbers by 100× compared to standard neural networks

Shaozhi Li ^{*}*National Center for Transportation Cybersecurity and Resiliency (TracR), [Clemson University](#), South Carolina 29631, USA*M Sabbir Salek  and Mashrur Chowdhury*National Center for Transportation Cybersecurity and Resiliency (TracR), [Clemson University](#), South Carolina 29631, USA
and Glenn Department of Civil Engineering, [Clemson University](#), Clemson, South Carolina 29634, USA*Yao Wang [†]*Department of Chemistry, [Emory University](#), Atlanta, Georgia 30322, USA*

(Received 18 May 2025; accepted 12 January 2026; published 9 February 2026)

Although quantum machine learning has shown great promise, the practical application of quantum computers remains constrained in the noisy intermediate-scale quantum era. To take advantage of quantum machine learning, we investigate the underlying mathematical principles of these quantum models and find that the quantum neural network with amplitude encoding is equivalent to a weight-constrained neural network. Motivated by this discovery, we develop a classical weight-constrained neural network. We find that this approach can reduce the number of variables in a classical neural network by a factor of 135 while preserving its accuracy. In addition, we develop a dropout method to enhance the robustness of quantum machine learning models, which are highly susceptible to adversarial attacks. This technique can also be applied to improve the adversarial robustness of the classical weight-constrained neural network, which is essential for industry applications, such as self-driving vehicles. Our work offers an approach to reduce the complexity of large classical neural networks, addressing a critical challenge in machine learning.

DOI: [10.1103/wb1b-ntpc](https://doi.org/10.1103/wb1b-ntpc)

I. INTRODUCTION

Large artificial intelligence (AI) models are transforming our education and research, enabling groundbreaking advances in various domains [1–6]. For example, OpenAI’s GPT series and Google’s BERT have revolutionized our ability to generate humanlike texts, recognize complex image patterns, and interpret audio [7]. Similarly, AlphaFold has simplified the design and discovery of proteins and drugs [8,9]. The remarkable capabilities of these large AI models arise from their vast networks, often containing billions or even trillions of variables [10]. However, the large scale of these AI models induces many significant challenges, including high memory requirements, overfitting, convergence difficulties, and complex hyperparameter tuning [11,12]. Addressing these challenges requires the development of AI models that could maintain strong learnability while reducing the complexity of the models.

Emerging quantum machine learning (QML) techniques offer promising solutions to some challenges posed by large AI models [13–18]. The advantage of quantum speedups has been demonstrated in various machine learning tasks [14,19], including finding concise functions to fit exponentially large datasets [20], learning a particular data distribution [18], discovering unknown kernel functions [21], and extracting insights from experimental data points using exponentially fewer data compared to classical approaches [22]. These advances can significantly reduce the computational time required to train AI models. In addition to speed, quantum machine learning offers memory advantages by taking advantage of the exponential storage capability of quantum computers. For example, quantum neural networks (QNNs) with amplitude encoding require far fewer variables than their classical counterparts [23]. Consequently, the vast number of parameters in large AI models can be effectively reduced by employing quantum or hybrid quantum-classical models.

Driven by these advantages, QML has become an active area of research [24–34]. Although substantial progress has been made in the development of QML algorithms, there are still few real-world applications that leverage real quantum computers [35–39]. This gap largely results from the quantum noise issue, which makes results unreliable when many qubits are used [40,41]. Addressing this noise challenge is complex, and achieving a large-scale, noise-free quantum computer is expected to take considerable time. In light of these obstacles,

^{*}Contact author: lishaozhi@mail.neu.edu.cn[†]Contact author: yao.wang@emory.edu

Published by the American Physical Society under the terms of the [Creative Commons Attribution 4.0 International](#) license. Further distribution of this work must maintain attribution to the author(s) and the published article’s title, journal citation, and DOI.

the development of quantum-inspired classical algorithms that address the challenges of classical machine learning is a valuable direction to advance machine learning techniques [42–48].

To translate the advantages of quantum machine learning into classical methods, a deeper understanding of their mathematical foundations is required. Currently, such understanding is limited, and research often focuses on empirically demonstrating the success of QML across various datasets. For instance, quantum neural networks have been shown to succeed with fewer parameters in tasks like recovery rate prediction and medical image clarification; however, the reasons for this efficiency remain unclear [49,50]. Our work aims to clarify the mathematical principles of QNNs and explain the origin of these quantum advantages. We will then leverage this understanding to develop a classical neural network, achieving high performance with fewer parameters.

We first analyze the mathematical foundations of quantum neural networks with angle encoding and find that using a quantum circuit with angle encoding is mathematically equivalent to constructing a polynomial function, where each term is expressed as a continuous product of the cosine series [47]. In contrast, a quantum circuit with amplitude encoding generates a weight-constrained matrix, whose weight element is again expressed as a summation of the continuous product of the cosine series. The number of trainable variables in this weight-constrained matrix is much smaller than the dimensionality N of the matrix or the input space because the number of quantum gates is typically scaled as $\log_2 N$ not N . Therefore, a quantum circuit with amplitude encoding requires fewer variables than the classical network, whose variable number is typically scaled as N . Motivated by this discovery, we develop a classical weight-constrained neural network, which can reduce the number of variables by 100 times compared to the standard neural network (NN). We validate this weight-constrained method by examining its performance on four datasets: MNIST, Fashion MNIST (FMNIST), CIFAR, and TRAFFIC datasets. Our results demonstrate that the weight-constrained method maintains the accuracy (ACC) of the classical neural network while substantially reducing the number of variables. This approach holds promise for scaling down variable numbers in large AI models such as GPT and BERT. Furthermore, we develop a method to enhance the adversarial robustness of the quantum neural network and the classical weight-constrained neural network models by randomly dropping parameterized angles during prediction. We demonstrate that such a method can significantly increase robustness against adversarial attacks.

This paper is organized as follows. Section II presents the mathematical foundations of quantum neural networks with angle and amplitude encodings. Section III develops the theoretical framework of the proposed weight-constrained approach. Section IV demonstrates the effectiveness of applying this method to classical neural networks. Section V compares the performance of weight-constrained neural networks with that of networks employing low-rank matrices. Section VI examines the improved adversarial robustness achieved through our proposed dropout technique. Section VII analyzes the key benefits and broader implications of the weight-constrained methodology. Finally, Section VIII concludes the paper with

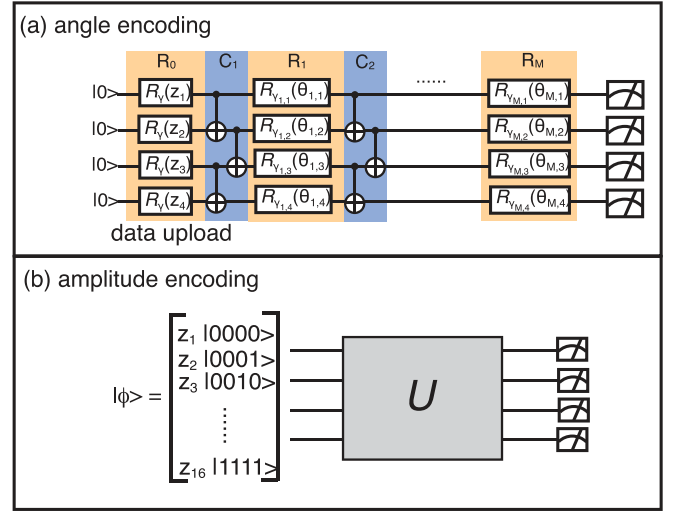


FIG. 1. Hybrid quantum-classical neural network. (a) A generic quantum circuit with angle encoding. (b) A generic quantum circuit with amplitude encoding.

a summary of findings and potential directions for future research.

II. QUANTUM NEURAL NETWORKS

To develop quantum-inspired classical neural networks, we explore the underlying mathematics of quantum neural networks with angle and amplitude encodings [1]. After building on this mathematical foundation, we develop a quantum-inspired classical neural network, which utilizes 100 times fewer variables than a standard classical neural network.

A. Quantum neural network with angle encoding

We start to analyze the quantum neural network with angle encoding, which adopts the input value as rotation angles in the quantum gates, as shown in Fig. 1(a). The nonlinear relationship between the output of a quantum circuit and the input data $\mathbf{z}$ is constructed by applying different rotation gates and cnot gates in this circuit. Without loss of generality, we consider a quantum circuit shown in Fig. 1(a), which has one data-upload layer $\tilde{R}_0$, M rotation layers, and M cnot layers. The rotation operator R_y in Fig. 1(a) is chosen from the collection of $\{R_x, R_y, R_z\}$. Then, the final quantum state of this quantum circuit is

$$|\psi\rangle = \tilde{R}_M \tilde{C}_M \cdots \tilde{R}_1 \tilde{C}_1 \tilde{R}_0 |0000\rangle, \quad (1)$$

where $\tilde{R}_i$ denotes a matrix constructed from the tensor product of different single-gate rotation matrices within one layer and $\tilde{C}_i$ denotes a matrix constructed from the tensor product of two-gate cnot matrices. Since $\tilde{R}_i$ is composed solely of rotation operators, its matrix elements are either trigonometric functions or zero, whereas the matrix elements of $\tilde{C}_i$ are either 0 or 1. Defining

$$U = \tilde{R}_M \tilde{C}_M \cdots \tilde{R}_1 \tilde{C}_1 \tilde{R}_0, \quad (2)$$

then the final state can be expressed as

$$|\psi\rangle = U|0000\rangle = \sum_i U_{i,1}|b_i\rangle, \quad (3)$$

where b_i denotes the computational basis states (0000, 0001, 0011, ...) and $U_{i,1}$ is the i th element of the first column of U , which is either a sum of products of trigonometric functions or zero due to the matrix multiplication. In general, $U_{i,j}$ can be written as

$$U_{i,j} = \sum_s p_s \prod_{i \in \mathbb{N}_s} \cos(\theta_i/2 + q_i) \prod_{n \in \mathbb{M}_s} \cos(z_n/2 + q_n), \quad (4)$$

where p_s belongs to $\{0, 1, -1, i, -i\}$. The symbol q_i denotes 0 or $\pi/2$. $\mathbb{N}_s$ and $\mathbb{M}_s$ are collections of integer numbers.

For a measurement operator A composed of Pauli operators, the circuit output is

$$O = \langle \psi | A | \psi \rangle = \sum_{i,j} U_{i,1}^* A_{i,j} U_{j,1} = \sum_s f_s(\theta, \mathbf{z}), \quad (5)$$

$$f_s(\theta, \mathbf{z}) = c_s \prod_{i \in \mathbb{N}_{1,s}} \cos(\theta_i/2 + q_i) \prod_{j \in \mathbb{N}_{2,s}} \cos(\theta_j/2 + q_j) \\ \times \prod_{n \in \mathbb{M}_{1,s}} \cos(z_n/2 + q_n) \prod_{m \in \mathbb{M}_{2,s}} \cos(z_m/2 + q_m), \quad (6)$$

where c_s belongs to $1, -1$. It is straightforward to find that the output O of a quantum circuit is a polynomial function. Equation (5) shows that the quantum neural network with angle encoding generates a complicated activation function, which could select features better than the classical activation functions, such as ReLU and Tanh. We refer the reader to our previous work [47] for a deeper understanding of this quantum activation function.

B. Quantum neural network with amplitude encoding is a weight-constrained neural network

Unlike angle encoding, which transforms real-world data into rotation angles, amplitude encoding transforms real-world data into coefficients of pure quantum states. For the amplitude encoding, the initial state of a quantum circuit $|\phi\rangle$ is set as

$$|\phi\rangle_i = \frac{z_i}{|\mathbf{z}|} |b_i\rangle, \quad (7)$$

where $|b_i\rangle$ denotes a pure quantum state, i.e., $|0000\rangle$ and $|0001\rangle$. $|\mathbf{z}|$ is the norm value of the input vector $\mathbf{z}$. This initial quantum state can be transformed to other states by applying rotation and cnot gates. For simplicity, we can use an operator U to represent these operations as shown in Fig. 1(b). Then, the final quantum state is written as

$$|\psi\rangle = U|\phi\rangle. \quad (8)$$

If we set the measurement operator as A , the output O of this quantum circuit is

$$O = \langle \phi | U^\dagger A U | \phi \rangle = \langle \phi | W | \phi \rangle, \quad (9)$$

where W is a unitary matrix. It is easy to rewrite Eq. (9) as

$$O = \frac{1}{|\mathbf{z}|^2} \sum_{i,j} z_i W_{ij} z_j. \quad (10)$$

Equation (10) shows that the quantum circuit generates a correlation matrix between any two elements in the input data. For example, $W_{1,2}$ represents the correlation between the first and second input features, and $W_{1,100}$ denotes the correlation between the first and the one hundredth input features. If all elements in W are nonzero, the construction of features incorporates long-range and short-range correlations [21].

The weight matrix element W_{ij} is mathematically equivalent to the matrix U defined in Eq. (4). This weight element is given by

$$W_{i,j} = \sum_s p_s \prod_{k \in \mathbb{N}_s} \cos(\theta_k/2 + q_k), \quad (11)$$

where θ_k is a rotation angle, $p_s \in \{0, 1, -1, i, -i\}$, $q_k \in \{0, \pi/2\}$, and $\mathbb{N}_s$ is a set of integers.

Equation (11) shows that each weight element is a polynomial function, where each term is a continuous product of the cosine series. Therefore, the amplitude-encoded quantum neural network is a weight-constrained neural network. In a typical quantum circuit, the number of rotation angles is significantly smaller than the dimensionality of the Fock space it encodes. Consequently, an amplitude-encoded quantum neural network operates with fewer trainable variables than a standard neural network of equivalent representational capacity.

III. QUANTUM-INSPIRED CLASSICAL NETWORK: WEIGHT-CONSTRAINED NEURAL NETWORK

Motivated by the discovery that the amplitude-encoded quantum neural network is a weight-constrained neural network, we develop a classical counterpart of the weight-constrained neural network, in which the weight variables are constructed as follows. To generate K different weights, we first define N different variables, denoted as θ . Then, a combination process is performed, where r variables are chosen from these N variables, resulting in $C(N, r)$ possible combinations. Labeling the k th combination as $\theta^{(k)}$, the k th weight w_k is defined as

$$w_k = \prod_{i \in \text{even}, i \leq r} \cos(\theta_i^{(k)}) \prod_{j \in \text{odd}, j \leq r} \sin(\theta_j^{(k)}). \quad (12)$$

This combination strategy enables constructing thousands of weights using only a few dozen variables. We note that the weight correlation at the first moment is zero because

$$\int_{-\pi}^{\pi} d\theta w_k w_{k'} = 2^{N-r} \pi^N \delta_{k,k'}. \quad (13)$$

In addition, these weights are always correlated in higher moments, that is,

$$\int_{-\pi}^{\pi} d\theta w_k^s w_{k'}^s \neq 0, \quad (14)$$

when $s > 1$.

A critical question when adopting the weight-constrained method is how it impacts the performance of a neural network. To answer this question, we examine two distinct two-layer

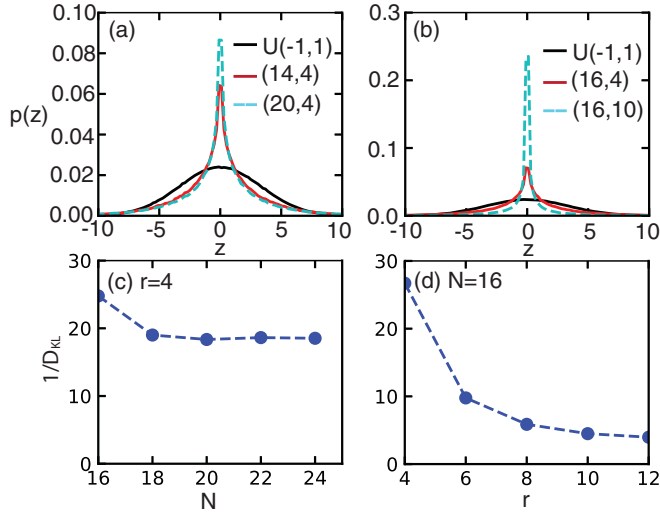


FIG. 2. Panels (a) and (b) plot the distribution $p(z)$ of z , defined as $z = \sum_i w_i x_i$, where w denotes the weight and x denotes the input features. (c) The inverse of Kullback-Leibler divergence D_{KL}^{-1} of the distribution $p(z)$ as a function of N at $r = 4$, where N denotes the number of variables. (d) D_{KL}^{-1} of the distribution $p(z)$ as a function of r at $N = 16$.

neural networks. In the first classical network, 100 input features, x , are connected to an output neuron, y , as follows:

$$y = f(z), \quad (15)$$

$$z = \sum_i w_i x_i, \quad (16)$$

where w_i represents the weight, sampled from the uniform distribution $\mathcal{U}(-1, 1)$. Here, $f(z)$ denotes a standard classical activation function, such as Tanh, Sigmoid, or ReLU. The second classical neural network has the same architecture as the first but employs the weight-constrained method to determine the weights. Since the variance of the constrained weights, as defined in Eq. (12), scales as $2^{-r/2}$, we adjust them to remove the impact of the variance. We redefine the weights as

$$w'_k = \sqrt{\frac{2^r}{3}} \prod_{i \in \text{even}, i \leq r} \cos(\theta_i^{(k)}) \prod_{j \in \text{odd}, j \leq r} \sin(\theta_j^{(k)}). \quad (17)$$

Then, the variance of w'_k is $\sqrt{1/3}$, matching the variance of the weights w_i drawn from $\mathcal{U}(-1, 1)$. Figures 2(a) and 2(b) display the distribution $p(z)$ of z for both classical neural networks. The input features x are sampled from a uniform distribution $\mathcal{U}(-1, 1)$. The black curve represents the distribution for the classical neural network with weights sampled from $w \in \mathcal{U}(-1, 1)$. The other curves show results for the weight-constrained network across different pairs of (N, r) used in constructing the combination sets. As shown in Figs. 2(a) and 2(b), both networks exhibit dome-shaped distributions centered at $z = 0$. Including a uniformly distributed bias term does not affect the central position of these distributions.

Although the mean and variance of the distributions for the weights from the uniform distribution and the weight-constrained method are the same, the weight-constrained

network displays a narrower distribution width and a more pronounced peak compared to the network using uniformly distributed weights. Moreover, we observe that the weight distribution of the weight-constrained network shows a strong dependence on r but a weak dependence of N . For instance, at $r = 4$, increasing the value of N from 14 to 20 only slightly raises $p(z = 0)$ from 0.06 to 0.08. In contrast, at $N = 16$, increasing r from 4 to 10 causes a significant enhancement of $p(z = 0)$. This result implies that the performance of the weight-constrained network has a strong dependence on r but a weak dependence of N .

To quantify the performance of a network, we evaluate its expressibility, which characterizes the ability of a network to represent or predict arbitrary real value. In the ideal case, the variable z can take any real value within the range $(-\infty, \infty)$, and its probability distribution $p(z)$ is uniform. Such a case corresponds to the strongest expressibility. However, when constraints are imposed on both the weights w and inputs x , the resulting distribution $p(z)$ becomes dependent on z , thereby reducing the expressibility. To quantify this deviation, we employ the Kullback-Leibler (KL) divergence, D_{KL} , which measures the difference between $p(z)$ of the network and $q(z)$ of the ideal case:

$$D_{KL} = \int_{-\infty}^{\infty} dz p(z) \log \left[\frac{p(z)}{q(z)} \right]. \quad (18)$$

In practical computations, we should restrict the integration domain to $(-R, R)$ with a large cutoff R , for which $q(z) = 1/(2R)$. Under this approximation, the KL divergence becomes

$$D_{KL} = \int_{-R}^R dz p(z) \log p(z) + \log(2R), \quad (19)$$

when $p(z)$ closely resembles the uniform distribution of the ideal case, $D_{KL} \approx 0$; conversely, when $p(z)$ collapses to a delta function $\delta(z)$, we obtain $D_{KL} = \log(2R)$. Therefore, D_{KL} serves as an inverse measure of expressibility—the smaller the KL divergence, the greater the expressibility of a network.

In our calculations, we use the output distribution $p(z)$ of a network with weights sampled from $U(-1, 1)$ as the reference distribution $q(z)$. This choice is justified because this unrestricted network possesses higher expressibility than the weight-constrained network (see Appendix F for details). Figures 2(c) and 2(d) plot the inverse KL divergence, $1/D_{KL}$ (blue circles), as a function of N (at $r = 4$) and r (at $N = 16$), respectively. The results indicate that $1/D_{KL}$ decreases slightly with increasing N for $N < 18$ and becomes independent of N for $N > 18$. In contrast, $1/D_{KL}$ decreases rapidly as r increases. This behavior suggests that the expressibility of the weight-constrained network depends weakly on N , while it increases significantly as r decreases.

IV. PRACTICALITY OF THE WEIGHT-CONSTRAINED METHOD

Having discussed the expressibility of classical neural networks employing the weight-constrained method, we now focus on its practical applications in classical fully connected and convolutional neural networks.

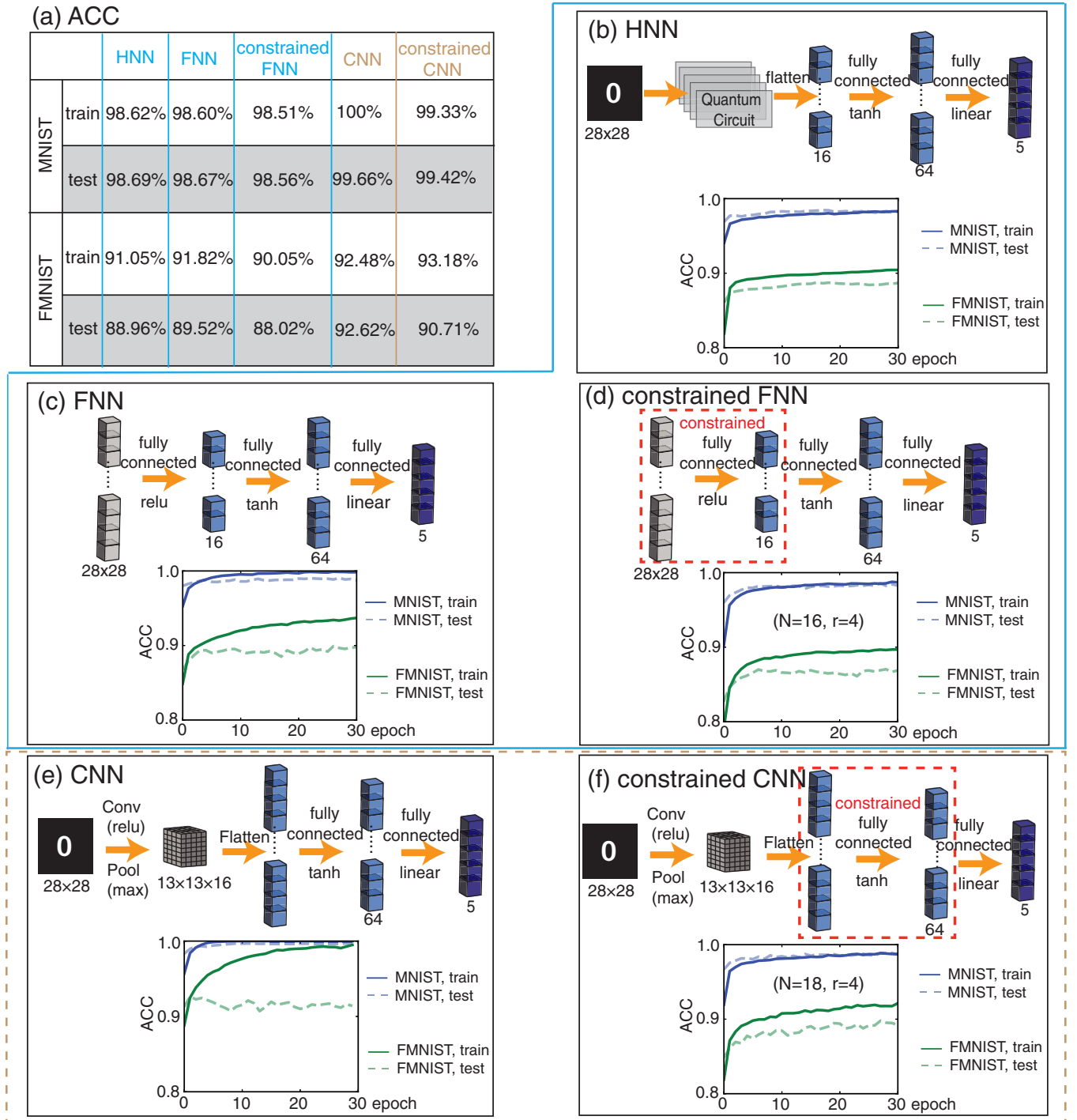


FIG. 3. Training results of different neural networks. (a) A summary of the ACC of these five models for the MNIST and FMNIST datasets. (b) The architecture of the HNN and the training results on the MNIST and FMNIST datasets. (c) The FNN architecture and the training results on the MNIST and FMNIST datasets. (d) The weight-constrained FNN architecture and the training results on the MNIST and FMNIST datasets. (e) The CNN architecture and the training results on the MNIST and FMNIST datasets. (f) The architecture of the weight-constrained CNN and the training results on the MNIST and FMNIST datasets.

We begin by comparing the performance of the weight-constrained neural network with that of a standard classical fully connected neural network (FNN) and a hybrid neural network (HNN). The architectures of these three networks are depicted in Figs. 3(b)–3(d), respectively. All three

networks consist of four layers, with 16 neurons in the first hidden layer and 64 neurons in the second hidden layer. Details on the quantum circuit design are provided in Appendix A. For the weight-constrained FNN, the weights in the first hidden layer are determined using Eq. (17), which

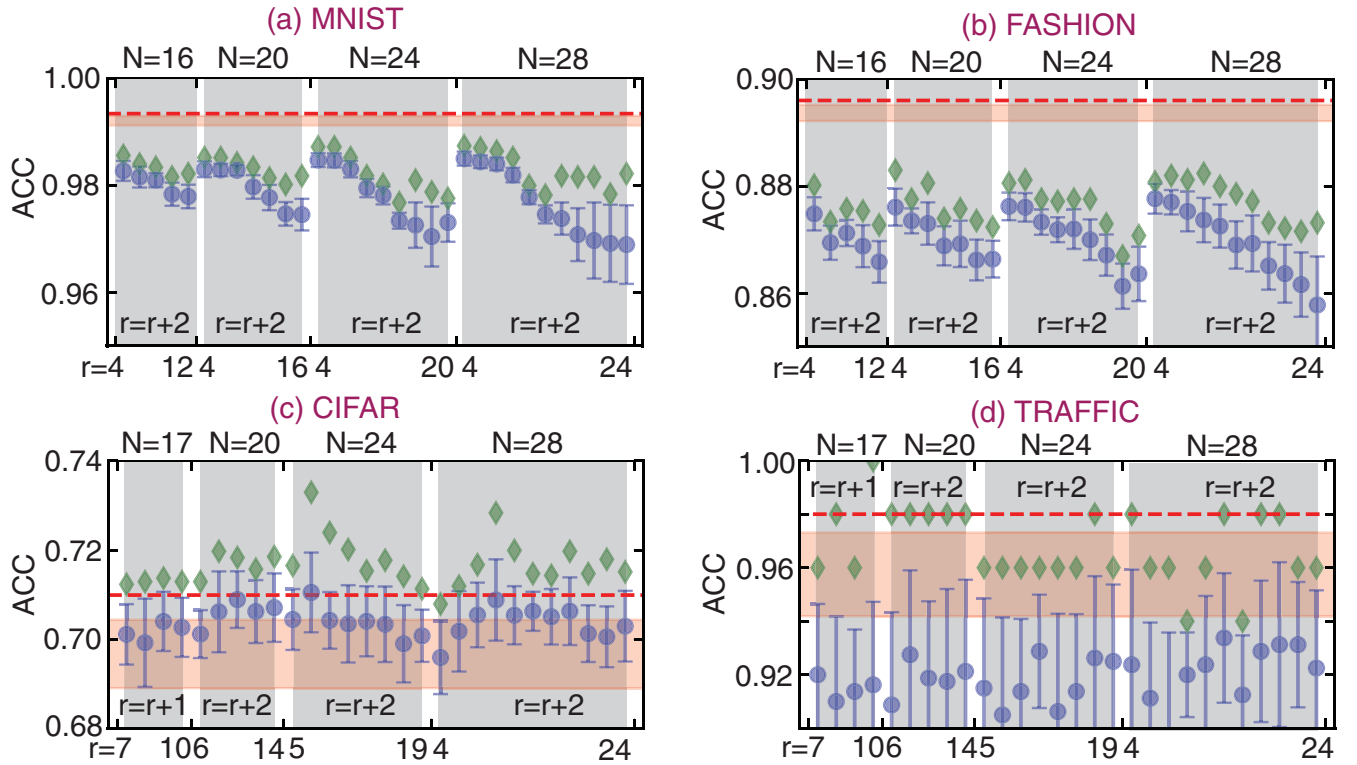


FIG. 4. The ACC of the weight-constrained FNN and the weight-constrained CNN for different values of N and r . Here, N and r represent variables in the combination formula $C(N, r)$. Panels (a) and (b) plot the results of the MNIST and the FMNIST datasets. Panels (c) and (d) plot the results of the CIFAR and the TRAFFIC datasets. The red dashed line and shaded region represent the optimal value and the ± 1 standard deviation interval of the standard network, respectively.

leads to the same variance for different r values. In addition to FNN, we evaluate the performance of the weight-constrained convolutional neural network (CNN) and its standard counterpart. Figure 3(e) illustrates a standard CNN architecture comprising one convolutional layer, one pooling layer, and two fully connected layers. The weight-constrained CNN, shown in Fig. 3(f), shares the same architecture as the standard CNN, but the weight-constrained method is applied to the first fully connected layer. The weights for the standard layer are initialized using the He method, and the variables for the weight-constrained method are initialized from a uniform distribution $\mathcal{U}(-\pi, \pi)$.

The table in Fig. 3(a) summarizes the optimal results of the HNN, FNN, constrained FNN, CNN, and constrained CNN for the MNIST and FMNIST datasets. These results are determined based on the minimal loss function for the test datasets in one training simulation. In the HNN, each quantum circuit comprises 38 variables. In the weight-constrained FNN, the combination sets are constructed by selecting four elements from a pool of 16 variables ($N = 16$ and $r = 4$). This combination set is chosen randomly. With this parameter setting, the number of variables in the first layer of the HNN and the constrained FNN is reduced by factors of approximately 19 and 49, respectively, compared to the standard FNN. For the MNIST test dataset, the HNN and the FNN achieved similar accuracies, 98.7%, about 0.1% larger than the ACC of the weight-constrained FNN. For the FMNIST test dataset, the HNN and the FNN performed comparably, achieving an ACC of 89%, approximately 1% larger than that

of the weight-constrained FNN. These results indicate that introducing weight constraints does not significantly degrade the performance of neural networks. The slightly better performance of the HNN compared to the weight-constrained FNN can be attributed to the more effective weight construction provided by the quantum circuit.

Figures 3(e) and 3(f) plot training results of the CNN and the weight-constrained CNN on these two datasets. Here, the weight-constrained method is implemented by selecting four elements from a pool of 18 variables, reducing the number of variables by a factor of 150 compared to the standard CNN. Despite this substantial reduction in variables, the table in Fig. 3(a) shows that the weight-constrained CNN achieves an ACC comparable to the standard CNN on the MNIST test dataset. For the FMNIST test dataset, the ACC of the weight-constrained CNN is only 1.91% smaller than that of the standard CNN. Notably, we observe that the loss function of the weight-constrained CNN for the test dataset decreases monotonically, whereas that of the standard CNN first decreases and then increases as the number of epochs grows (see Appendix B for details). This behavior indicates that the weight-constrained method effectively mitigates overfitting, as the reduced expressibility of the constrained network limits its tendency to learn noise from the training data.

To further investigate the impact of the combination method in the weight-constrained strategy, we perform simulations using different values of N and r . Figure 4 presents the ACC for the MNIST, FMNIST, CIFAR, and TRAFFIC test datasets. In these simulations, the weight-constrained FNN

is applied to the MNIST and FMNIST datasets, whereas the weight-constrained CNN is used for the CIFAR and TRAFIC datasets. For each (N, r) pair, the green diamond and blue circle (with error bars) represent the optimal and mean ACC from 16 independent simulations, respectively. The red dashed line and shaded region represent the optimal value and the ± 1 standard deviation interval of the standard network, respectively. For the weight-constrained FNN, the mean ACC generally decreases as r increases, which is attributed to the fact that the expressibility is suppressed by r . In contrast, the mean ACC shows no significant dependence on N , consistent with our observation that the expressibility depends only weakly on N . We note that the impact of expressibility is also dependent on the activation function. For the ReLU activation function, $\text{relu}(z)$ is nonzero as $z \rightarrow \infty$, whereas $\tanh(z)$ becomes 0 as $z \gg 1$. Therefore, varying r has a less pronounced impact when the Tanh activation function is applied to the weight-constrained layer (see Appendix C for details). For the weight-constrained CNN, the mean ACC shows no clear relationship with r . This is because the neuron value x in the weight-constrained layer can be adjusted by the preceding layers, thereby mitigating the expressivity limitations imposed by the weight-constrained approach (see Appendix C for details).

The performance of the weight-constrained neural network, compared to the standard classical neural network, varies depending on the architecture of the neural network. We observe that the optimal ACC of the weight-constrained FNN (green diamonds) is slightly smaller than that of the standard ACC (red dashed line). In contrast, the optimal ACC of the weight-constrained CNN is comparable to or even larger. This result implies that the weight-constrained approach delivers superior performance in CNNs compared to FNNs.

V. COMPARISON: WEIGHT-CONSTRAINED VERSUS LOW-RANK NEURAL NETWORKS

Compared to standard neural network, the proposed weight-constrained method has fewer parameters by restricting weight variables. In machine learning, other techniques have been proposed to reduce variables, such as the low-rank matrix [51] and the weight sharing technique [52]. Here, we compare the performance of the weight-constrained method with low-rank neural networks.

A low-rank weight matrix is constructed as $W = A \times B$, where A has dimensions $s \times m$ and B has dimensions $m \times n$. The variable number of this low-rank weight matrix is $sm + mn$, which is smaller than the size of W , when $m \ll s$ and $m \ll n$; therefore, the low-rank weight matrix method reduces the variable number. To ensure a fair comparison with the weight-constrained neural network, in our study, the low-rank weight matrix is applied in the first hidden layer for the FNN and the second-to-last layer for the CNN.

Figure 5 shows the ACC for various neural networks across different test datasets. Results show the mean (blue circle) and standard deviation (green diamond) from 16 independent simulations. The red symbol denotes the result of the standard neural network. For the MNIST and FMNIST datasets, we compare a standard FNN against its low-rank matrix and weight-constrained variants. Similarly, for the CIFAR and TRAFIC datasets, we compare a standard CNN

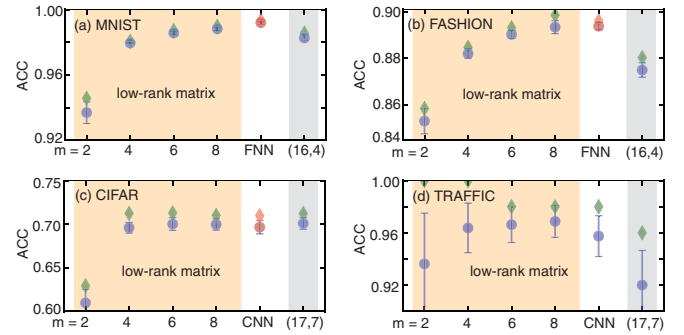


FIG. 5. Test ACC of neural networks with different weight parametrizations. (a) MNIST and (b) FMNIST results for FNNs. (c) CIFAR and (d) TRAFIC dataset results for CNNs. For each panel, performance is shown for a standard network, our proposed weight-constrained method, and a low-rank method with varying m . Markers indicate the mean ACC over 16 simulations (circles) and the optimal result (diamonds).

against its low-rank and weight-constrained counterparts. The weight-constrained FNN with $N = 16$ and $r = 4$ contains 256 parameters in the constrained layer, and the weight-constrained CNN with $N = 17$ and $r = 7$ includes 1088 parameters. In contrast, the low-rank FNNs with $m = 2, 4, 6$, and 8 contain 1600–6400 parameters in the low-rank matrix, while the corresponding low-rank CNNs contain 21 728–86 912 parameters. The results show that the mean ACC of the low-rank models increases with m , eventually approaching that of the standard NN. Despite using substantially fewer parameters, the weight-constrained networks achieve mean accuracies comparable to those of the low-rank models with $m = 4$ on all datasets except TRAFFIC. The deviation observed for the TRAFFIC dataset may result from its relatively small training dataset. Overall, these results demonstrate that the proposed weight-constrained method is more parameter efficient while maintaining competitive performance relative to the low-rank approach.

VI. DROPOUT ENHANCED ADVERSARIAL ROBUSTNESS

Adversarial machine learning is an emerging frontier that studies the vulnerability of machine learning systems and develops defense strategies against adversarial attacks. It was demonstrated that quantum machine learning methods for classification tasks are highly vulnerable to adversarial attacks as a result of the concentration of measure phenomenon, which refers to the fact that the distance from a (Haar) randomly sampled point in the Hilbert space to the nearest adversarial example vanishes as $\mathcal{O}(2^{-n})$, where n is the number of qubits [53]. Moreover, it is found that universal adversarial samples can deceive all classifiers with only a perturbation of strength $\mathcal{O}(\log(k)2^{-n})$, where k is the number of independent classifiers [54]. Recently, several approaches have been proposed to improve the robustness of quantum machine learning models, including applying quantum noise [55] and adversarial training [56]. In this work, we propose an approach to improve adversarial robustness.

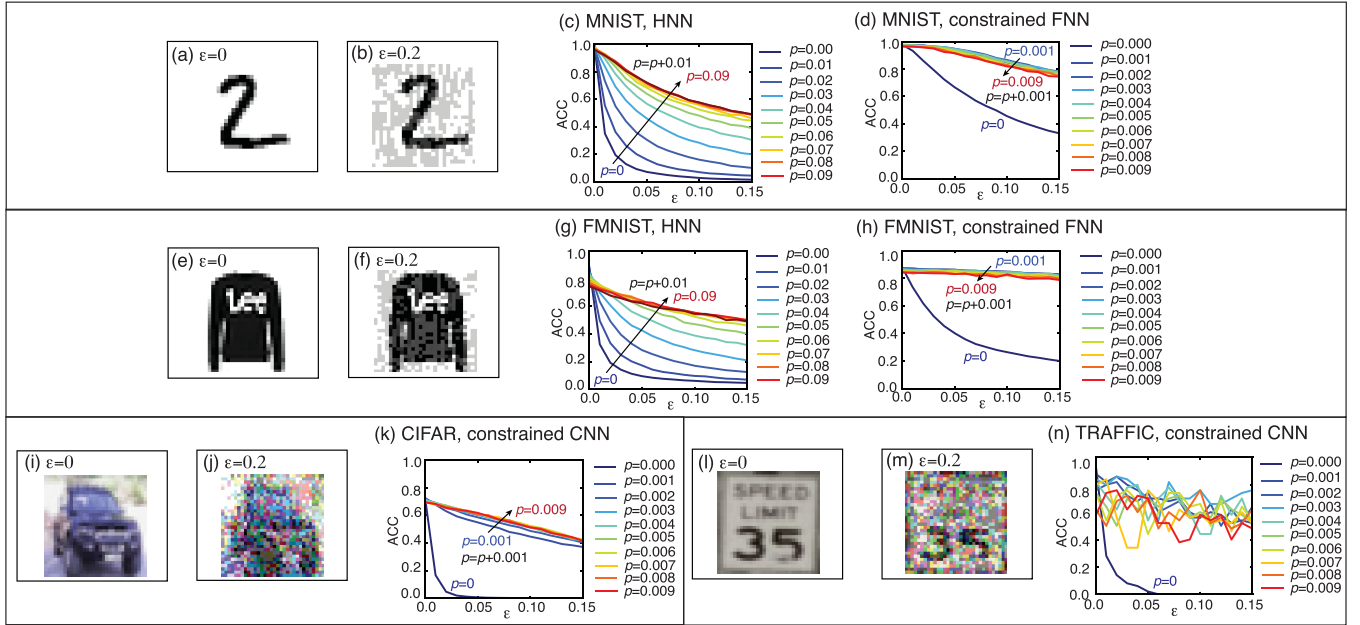


FIG. 6. The ACC under adversarial attack. Panels (a) and (b) show the MNIST image and the image under an adversarial attack with an attack intensity $\epsilon = 0.2$. Panels (c) and (d) plot the ACC of the HNN and the weight-constrained FNN with a dropout policy. Here, p denotes the dropout probability. Panels (e) and (f) show the FMNIST image and the image under an adversarial attack with an attack intensity $\epsilon = 0.2$. Panels (g) and (h) plot the ACC of the HNN and the weight-constrained FNN with a dropout policy. Panels (i) and (j) show the CIFAR image and the image under an adversarial attack with an attack intensity $\epsilon = 0.2$. Panel (k) plots the ACC of the weight-constrained CNN with a dropout policy. Panels (l) and (m) show the traffic image and the image under an adversarial attack with an attack intensity $\epsilon = 0.2$. Panel (n) plots the ACC of the weight-constrained CNN with a dropout policy.

In this study, we use the fast gradient sign method (FGSM) to generate adversarial samples [57]. For the FGSM method, each pixel is modified by $\epsilon \text{sgn}(\nabla_x L)$, where L is the loss function and ϵ represents the strength of the attack. Therefore, a positive value of ϵ increases the loss function, reducing accuracy. To visualize the adversarial attack, we plot original images and their corresponding adversarially perturbed versions in Figs. 6(a), 6(b), 6(e), 6(f), 6(i), 6(j), 6(l), and 6(m).

To enhance the robustness of the quantum machine learning model, we propose a randomized dropout strategy for R_z gates within the quantum circuit of a trained model. In this approach, dropout is applied after training, and attackers only interact with the model after dropout has been incorporated. Figures 6(c) and 6(g) plot the ACC of the HNN model under adversarial attack for the MNIST and FMNIST test datasets, where p denotes the probability of dropping out. Without the dropout policy ($p = 0$), the ACC for both datasets collapses to near zero at $\epsilon = 0.05$, revealing the inherent vulnerability of quantum models to attacks. In contrast, dropout maintains robust performance. For example, with $p = 0.08$, the ACC for the MNIST dataset declines from 0.968 to 0.567, and for the FMNIST dataset, it drops from 0.774 to 0.555 when ϵ increases from 0 to 0.1. It is also observed that the ACC for unattacked samples is suppressed by the dropout probability. For example, increasing the probability of dropping out from 0 to 0.08 reduces the ACC by approximately 0.02 for the MNIST dataset and by 0.112 for the FMNIST dataset. These results highlight that while the dropout method effectively enhances the adversarial robustness of quantum neural networks,

it is crucial to carefully select the dropout probability to strike a balance between maintaining ACC for unattacked samples and improving robustness under attack.

The dropout method can be extended to classical weight-constrained neural networks to enhance their adversarial robustness. Dropping an R_z gate in a quantum circuit analogously corresponds to changing elements of the weight matrix W shown in Eq. (10). In a weight-constrained neural network, the weight is constructed from the continuous product of the trigonometric function. Therefore, we implement dropout by randomly selecting rotation angles and setting their corresponding trigonometric terms to 1 (equivalent to removing their contributions). Figures 6(d) and 6(h) illustrate the ACC of the weight-constrained FNN on the MNIST and FMNIST test datasets under adversarial attack, respectively. It is found that a small nonzero dropout probability ($p = 0.001$) can significantly improve adversarial robustness and make the change of the ACC under attacks tiny. For example, when $p = 0.001$ is adopted, the ACC for the FMNIST dataset is only decreased by 0.04 as the attack strength ϵ increases from 0 to 0.15. Compared to the hybrid quantum-classical model, the weight-constrained FNN requires a smaller value of p to achieve enhanced adversarial robustness. This ensures that the ACC for unattacked samples remains almost unaffected when such a small p is adopted. Furthermore, Figs. 6(k) and 6(n) plot the ACC of the weight-constrained CNN, which is trained on the CIFAR and TRAFFIC datasets, respectively. Again, we observe that the adversarial robustness is enhanced by the dropout method. The oscillatory behavior observed in

Fig. 6(n) is attributed to the small sample size of the TRAFFIC test dataset (50 samples), which lacks sufficient statistical robustness.

We note that the success of our approach on the TRAFFIC dataset is critical to ensuring the safety of self-driving vehicles [58,59]. In these vehicles, CNN-based regression models are widely used for object recognition. However, CNNs have been shown to be highly susceptible to adversarial samples [60]. To address this issue, numerous studies have focused on improving the adversarial robustness of CNNs [61,62]. Our work introduces a method to improve CNN robustness, thus improving the safety and reliability of self-driving vehicles.

VII. OVERVIEW OF THE BENEFITS OF THE WEIGHT-CONSTRAINED NEURAL NETWORK

In machine learning, regularization and early stopping are commonly employed to mitigate overfitting. Our developed weight-constrained neural network introduces an alternative approach to addressing this issue. By limiting its expressibility, the weight-constrained neural network effectively filters out noise in training data while capturing essential signals, thereby suppressing overfitting (see details in Appendix B).

Beyond suppressing overfitting, the weight-constrained method significantly reduces memory and storage requirements. In standard neural networks, all weights must be stored in memory, causing memory usage to scale linearly with the number of weights. In addition, large AI models require substantial storage to save trained parameters. The weight-constrained method can reduce these costs. For example, memory usage can be optimized by temporarily constructing and releasing the weight matrix after use. In addition, the memory cost for storing gradients of weights can also be reduced by adopting the weight-constrained method. This reduction is particularly critical for large-scale AI models. For example, a model with one billion parameters requires 8 GB of memory for weights and corresponding gradients with float-32 precision. By applying the weight-constrained method, the number of variables can be reduced by a factor of 100, reducing the memory requirement to just 81 MB.

To effectively apply QML to industry challenges, ensuring the security of QML models is paramount. Although standard classical techniques, such as adversarial training [56], can enhance the robustness of QML models, our work introduces an alternative approach that takes advantage of the intrinsic properties of quantum circuits. Additionally, our developed dropout method, based on the weight-constrained method, enriches techniques to improve the robustness of classical machine learning models.

VIII. CONCLUSION

In this work, we investigate the fundamental mathematical principles of quantum neural networks with angle and amplitude encodings. We find that quantum neural network with amplitude encoding is equivalent to a weight-constrained neural network. Building on this insight, we develop a classical weight-constrained neural network that simultaneously addresses overfitting and reduces memory requirements. Furthermore, we develop a dropout approach to enhance the

adversarial robustness of both quantum neural networks and classical weight-constrained neural networks. These advancements hold significant promise for practical machine learning applications in industrial settings.

However, our study demonstrates the effectiveness of the weight-constrained approach for conventional neural architectures, we have not evaluated its performance on transformer models. Recent work by Cherrat *et al.* [63] has proposed a quantum transformer variant that employs amplitude encoding for attention coefficient computation, demonstrating quantum advantages in MedMNIST classification tasks. This suggests an important direction for future research: investigating how weight constraints might be integrated into transformer architectures. Moreover, we observe that the quantum neural network with amplitude encoding, which corresponds to a quantum weight-constrained neural network, outperforms our proposed classical weight-constrained neural network. This finding encourages further investigation into alternative and potentially more effective weight-constrained methods.

As noted in Ref. [64], there exists an inherent trade-off in algorithm design—better performance on one task often comes at the expense of another. Our proposed weight-constrained method effectively reduces memory usage; however, it increases computational time due to the dynamic memory allocation required for constructing weights. Striking a balance between memory consumption and computational efficiency requires determining the optimal number of weights to be constructed using this method. However, we note that the need for dynamically constructing weight matrices is not required on quantum computers; therefore, quantum machine learning has advantages in both reducing memory consumption and improving computational efficiency.

ACKNOWLEDGMENTS

This work was supported by the National Center for Transportation Cybersecurity and Resiliency (TraCR) (a U.S. Department of Transportation National University Transportation Center) headquartered at Clemson University, Clemson, South Carolina, USA. Y.W. acknowledges support from the U.S. Department of Energy, Office of Science, Basic Energy Sciences, under Early Career Award No. DE-SC0024524.

DATA AVAILABILITY

The data that support the findings of this article are not publicly available upon publication because it is not technically feasible and/or the cost of preparing, depositing, and hosting the data would be prohibitive within the terms of this research project. The data are available from the authors upon reasonable request.

APPENDIX A: QUANTUM CIRCUIT

Figure 7 plots the ten-qubit quantum circuit used in Fig. 3. This circuit includes a rotation layer with R_y and R_z operators, three convolutional layers, and three pooling layers. The quantum circuits for the convolutional $U(\theta)$ and pooling

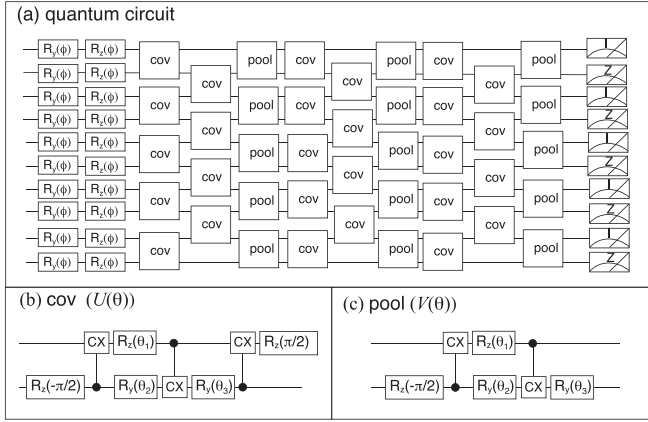


FIG. 7. Quantum circuit. (a) A quantum circuit used in Fig. 2. (b) The circuit for the convolutional (cov) layer. (c) The circuit for the pooling (pool) layer.

$V(\theta)$ operations are shown in Figs. 7(b) and 7(c), respectively. Variables are shared in each convolutional and pooling layer. These two operators are also used in Fig. 10.

APPENDIX B: TRAINING RESULTS ON THE FMNIST DATASET

Here, we compare the categorical cross-entropy loss functions of the CNN and the weight-constrained CNN for the FMNIST dataset, as shown in Fig. 8. The blue solid and dashed curves represent the training and test losses of the CNN, respectively, while the green solid and dashed curves correspond to those of the weight-constrained CNN. The parameters for the combination set are $N = 18$ and $r = 4$. The blue solid curve decreases continuously with increasing epoch number, whereas the blue dashed curve decreases initially and then increases, indicating overfitting in the CNN. In contrast, both the training and test losses of the weight-constrained CNN decrease monotonically, demonstrating that the overfit-

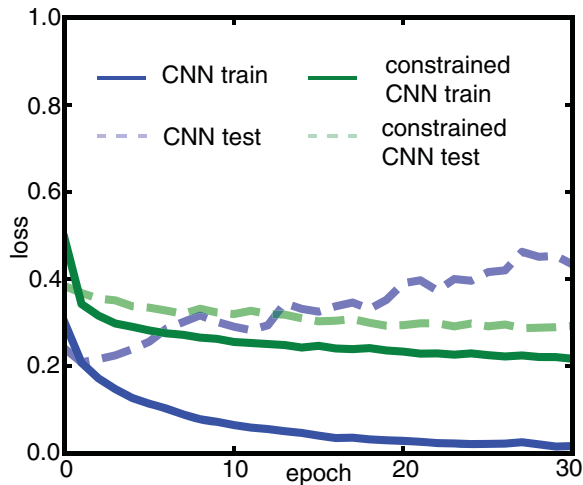


FIG. 8. Comparison of the categorical cross-entropy loss functions for the CNN and weight-constrained CNN on the FMNIST dataset.

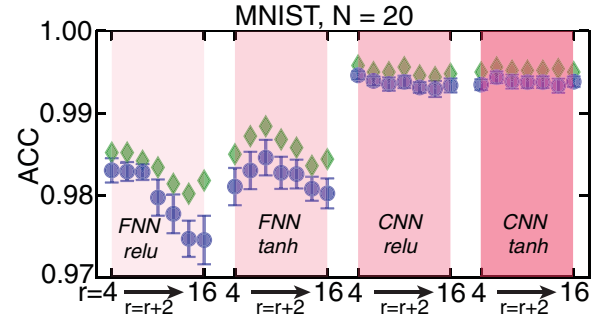


FIG. 9. Training results on the MNIST test dataset using different weight-constrained neural networks.

ting issue is effectively mitigated by the reduced expressibility of the weight-constrained network. Furthermore, we note that the minimal test loss of the CNN is smaller than the test loss of the weight-constrained CNN. Adopting the early-stop method, the optimal CNN model has a larger test ACC compared to the weight-constrained CNN, as shown in Fig. 3(a).

APPENDIX C: TRAINING RESULTS ON THE MNIST DATASET

Figure 9 compares the training results on the MNIST test dataset for four different weight-constrained networks: FNNs and CNNs employing either ReLU or Tanh activation functions. In the figure, the blue circle denotes the mean ACC, and the green diamond represents the optimal ACC. For the weight-constrained FNN with ReLU, the mean ACC decreases as r increases. In contrast, for the weight-constrained FNN with Tanh, the mean ACC initially increases and then decreases with r . However, as these variations fall within the range of static error, we conclude that the mean ACC for the Tanh FNN is effectively independent of r . For the weight-constrained CNN, the mean ACC is independent of r for both ReLU and Tanh functions. This is because the preceding layers can adjust the neuron values x in the weight-constrained layer, thereby mitigating the expressivity limitations imposed by the approach.

APPENDIX D: DIFFERENT NUMBERS OF FEATURES

Figure 10 plots the test ACC as a function of the number of features in the first hidden layer for both standard and weight-constrained FNNs. Blue circles and green diamonds represent the mean and optimal results from 16 independent simulations. For the weight-constrained FNN, the weight-constrained matrix is constructed by setting $N = 16$ and $r = 4$. The results on the MNIST and FMNIST datasets show that ACC increases as the number of features increases for both networks. Compared to the standard FNN, the weight-constrained FNN requires more features to achieve a similar ACC. For example, on MNIST, the standard FNN attains a mean ACC of about 0.98 with only four features, while the weight-constrained FNN needs 16 features to reach comparable performance. Nevertheless, the weight-constrained FNN uses fewer parameters overall. The FNN with four features in the first hidden layer has 3712 weights, while the

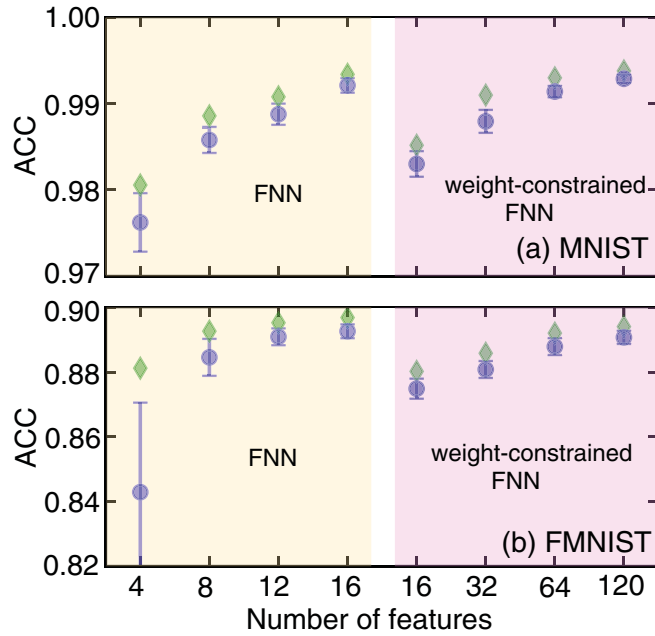


FIG. 10. The test ACC as a function of the number of features in the first hidden layer for both the standard and weight-constrained FNNs. Results on the MNIST and FMNIST datasets are presented in panels (a) and (b), respectively. The mean and optimal results are represented by blue circles and green diamonds, respectively. The weight-constrained matrix is constructed by setting $N = 16$ and $r = 4$.

weight-constrained FNN with 16 features has 1600 weight variables. A similar trend is observed for the FMNIST dataset: The standard FNN with 12 features achieves a mean ACC of about 0.89 using 10 496 weights, while the weight-constrained FNN with 64 features reaches a similar mean ACC with 5440 weight variables.

APPENDIX E: AMPLITUDE ENCODING

The QNN with amplitude encoding inherently limits the nonlinear relationship between input and output to a quadratic order, regardless of the complexity of the quantum circuit. This constraint makes it challenging for the QNN with amplitude encoding to extract features that involve three-point or four-point correlations. In contrast, a QNN with angle encoding can more effectively capture these higher-order correlations, as shown in Eq. (5). To illustrate this limitation, we consider the task of identifying whether a data point is above or below a boundary.

Figure 11(a) shows an HNN that we adopt to perform the task, classifying whether a data point is above or below a boundary. In this setup, the quantum circuit is responsible for feature extraction, while a fully connected classical neural network performs the classification. The quantum circuits for angle encoding and amplitude encoding are depicted in Figs. 11(b) and 11(c), respectively. The two-qubit operators U and V are sketched in Appendix A. As shown in Figs. 11(d)–11(g), we classify two different boundaries: $y_1 = x/2$ and $y_2 = x(x - 0.5)(x + 0.5)/2$, using angle and amplitude encodings. The black curve

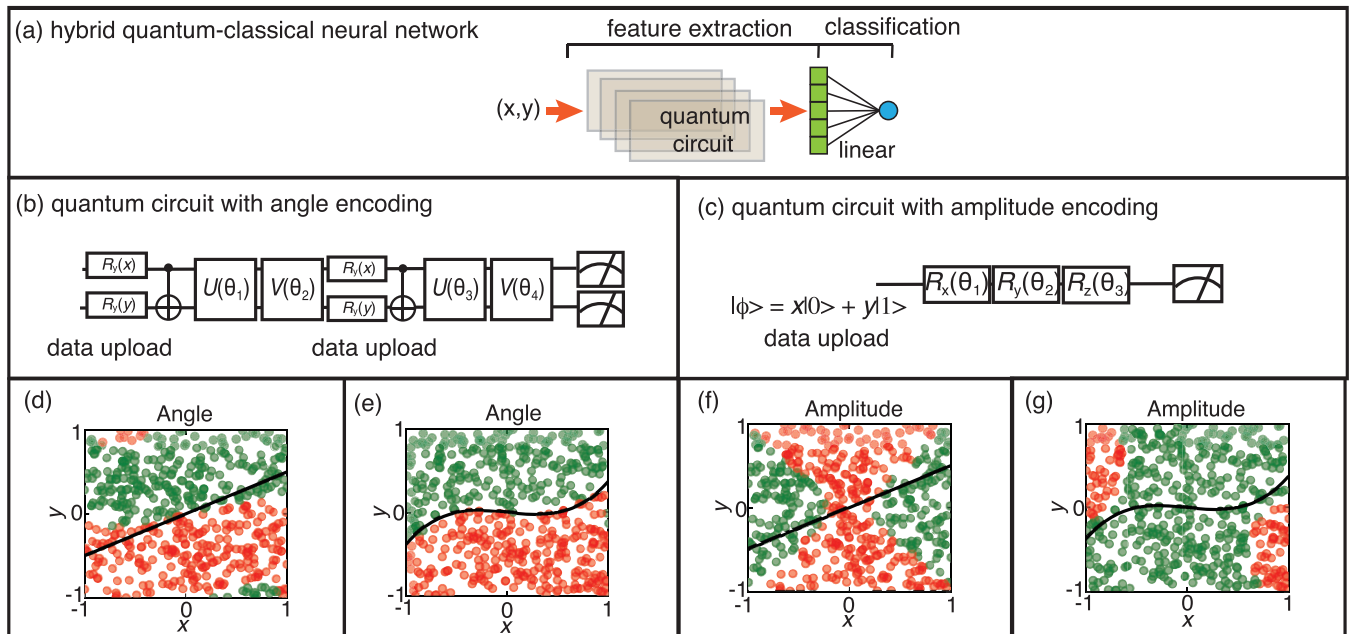


FIG. 11. Hybrid quantum-classical neural network. (a) A hybrid quantum-classical neural network. (b) A two-qubit quantum circuit using angle encoding. The operators U and V are illustrated in Appendix A. (c) A one-qubit quantum circuit using amplitude encoding. (d) Training results of the hybrid network with angle encoding for classifying whether a data point lies above or below the boundary $y = x/2$, shown as a black line. The green color denotes the predicted value above the boundary, while the red indicates the prediction below the boundary. (e) Training results of the hybrid network with angle encoding for classifying whether a data point lies above or below the boundary $y = x(x - 0.5)(x + 0.5)/2$. (f) Training results of the hybrid network with amplitude encoding, addressing the same task as in panel (d). (g) Training results of the hybrid network with amplitude encoding, addressing the same task as in panel (e).

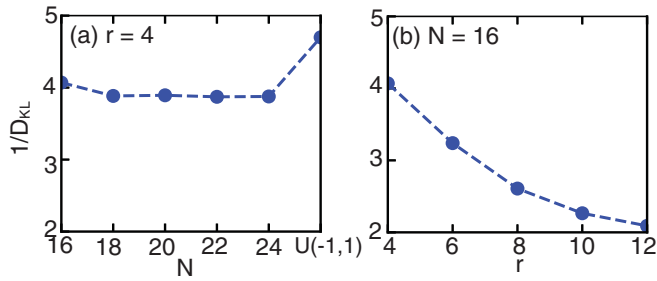


FIG. 12. (a) The inverse of Kullback-Leibler divergence D_{KL}^{-1} of the distribution $p(z)$ as a function of N at $r = 4$, where N denotes the number of variables. (b) D_{KL}^{-1} of the distribution $p(z)$ as a function of r at $N = 16$. Here, the reference distribution is set as the output distribution of a network with uniformly distributed weights.

denotes the boundary, the green dot denotes the predicted value above the boundary, and the red dot denotes the predicted value below the boundary. The QNN with amplitude encoding struggles to identify both boundaries because the features derived from terms like x^2 , xy , and y^2 are insufficient to characterize y_1 and y_2 . However, the QNN with angle encoding successfully identifies the boundary for y_2 due to its ability to capture the third-order term (i.e., x^3) inherent in the problem. The higher-order term can be obtained from the Taylor expansion of the cosine function. The subpar performance

of angle encoding for y_1 can be attributed to the excessive nonlinearity imposed by the QNN.

APPENDIX F: EXPRESSIBILITY

In this section, we present the inverse of Kullback-Leibler divergence D_{KL}^{-1} with $q(z) = 1/(2R)$ in Fig. 12. Here, R is set as 100, which is large enough to ensure $p(R) \approx 0$. The result shows that $1/D_{KL}$ is nearly independent of N but decreases rapidly as r increases. Furthermore, we observe that $1/D_{KL}$ of uniformly distributed weights is larger than that of the weight-constrained network, indicating that the standard network exhibits higher expressibility than the weight-constrained network.

APPENDIX G: DATASET

In our study, we adopted four different datasets: MNIST, Fashion MNIST, CIFAR, and a real-world traffic sign dataset derived from the LISA Traffic Sign Dataset [65] and the Mapillary Traffic Sign Dataset [66]. For each dataset, only five classes were selected as samples. For the MNIST, FMNIST, and CIFAR datasets, we use 30 000 samples for training and 5000 samples for testing. The traffic sign dataset has fewer samples, so we use 200 samples for training and 50 samples for testing.

- [1] M. Soori, B. Arezoo, and R. Dastres, Artificial intelligence, machine learning and deep learning in advanced robotics, a review, *Cognit. Rob.* **3**, 54 (2023).
- [2] R. Gozalo-Brizuela and E. C. Garrido-Merchan, ChatGPT is not all you need. A state of the art review of large generative AI models, [arXiv:2301.04655](https://arxiv.org/abs/2301.04655).
- [3] A. J. Thirunavukarasu, D. S. J. Ting, K. Elangovan, L. Gutierrez, T. F. Tan, and D. S. W. Ting, Large language models in medicine, *Nat. Med.* **29**, 1930 (2023).
- [4] B. Min, H. Ross, E. Sulem, A. P. B. Veyseh, T. H. Nguyen, O. Sainz, E. Agirre, I. Heintz, and D. Roth, Recent advances in natural language processing via large pre-trained language models: A survey, *ACM Comput. Surv.* **56**, 1 (2023).
- [5] Y. Liu, T. Han, S. Ma, J. Zhang, Y. Yang, J. Tian, H. He, A. Li, M. He, Z. Liu, Z. Wu, L. Zhao, D. Zhu, X. Li, N. Qiang, D. Shen, T. Liu, and B. Ge, Summary of ChatGPT-related research and perspective towards the future of large language models, *Meta-Radiology* **1**, 100017 (2023).
- [6] J. Qiu, L. Li, J. Sun, J. Peng, P. Shi, R. Zhang, Y. Dong, K. Lam, F. P.-W. Lo, B. Xiao, W. Yuan, N. Wang, D. Xu, and B. Lo, Large AI models in health informatics: Applications, challenges, and the future, *IEEE J. Biomed. Health. Inf.* **27**, 6074 (2023).
- [7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota, USA (Association for Computational Linguistics, 2019), pp. 4171–4186.
- [8] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Židek, A. Potapenko, *et al.*, Highly accurate protein structure prediction with AlphaFold, *Nature(London)* **596**, 583 (2021).
- [9] M. Varadi, D. E. V. Pires, S. Nair, D. Yeh, K. Tunyasuvunakool, D. Ochoa, J. Leman, G. R. Bickerton, *et al.*, AlphaFold protein structure database in 2024: Providing structure coverage for over 214 million protein sequences, *Nucl. Acids Res.* **52**, D368 (2024).
- [10] Z. Allen-Zhu, Y. Li, and Y. Liang, Learning and generalization in overparameterized neural networks, going beyond two layers, in *Advances in Neural Information Processing Systems* (British Columbia, Vancouver, Canada, 2019), Vol. 32.
- [11] D. Erhan, P.-A. Manzagol, Y. Bengio, S. Bengio, and P. Vincent, The difficulty of training deep architectures and the effect of unsupervised pre-training, in *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, edited by D. van Dyk and M. Welling (PMLR, Hilton Clearwater Beach Resort, Clearwater Beach, Florida, 2009), Vol. 5, pp. 153–160.
- [12] T. Bachlechner, B. P. Majumder, H. Mao, G. Cottrell, and J. McAuley, ReZero is all you need: Fast convergence at large depth, in *Proceedings of the 37th Conference on Uncertainty in Artificial Intelligence*, edited by C. de Campos and M. H. Maathuis (PMLR, 2021), Vol. 161, pp. 1352–1361.
- [13] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell,

- et al.*, Quantum supremacy using a programmable superconducting processor, *Nature (London)* **574**, 505 (2019).
- [14] M. Schuld and N. Killoran, Is quantum advantage the right goal for quantum machine learning? *PRX Quantum* **3**, 030101 (2022).
- [15] Y. Du, M.-H. Hsieh, T. Liu, and D. Tao, Expressive power of parametrized quantum circuits, *Phys. Rev. Res.* **2**, 033125 (2020).
- [16] J. Alcazar, M. G. Vakili, C. B. Kalayci, and A. Perdomo-Ortiz, Enhancing combinatorial optimization with classical and quantum generative models, *Nat. Commun.* **15**, 2761 (2024).
- [17] M. Cerezo, G. Verdon, H.-Y. Huang, L. Cincio, and P. J. Coles, Challenges and opportunities in quantum machine learning, *Nat. Comput. Sci.* **2**, 567 (2022).
- [18] Y. Liu, S. Arunachalam, and K. Temme, A rigorous and robust quantum speed-up in supervised machine learning, *Nat. Phys.* **17**, 1013 (2021).
- [19] G. D. Paparo, V. Dunjko, A. Makmal, M. A. Martin-Delgado, and H. J. Briegel, Quantum speedup for active learning agents, *Phys. Rev. X* **4**, 031002 (2014).
- [20] N. Wiebe, D. Braun, and S. Lloyd, Quantum algorithm for data fitting, *Phys. Rev. Lett.* **109**, 050505 (2012).
- [21] H.-Y. Huang, M. Broughton, M. Mohseni, R. Babbush, S. Boixo, H. Neven, and J. R. McClean, Power of data in quantum machine learning, *Nat. Commun.* **12**, 2631 (2021).
- [22] H.-Y. Huang, M. Broughton, J. Cotler, S. Chen, J. Li, M. Mohseni, H. Neven, R. Babbush, R. Kueng, J. Preskill, and J. R. McClean, Quantum advantage in learning from experiments, *Science* **376**, 1182 (2022).
- [23] M. S. Salek, S. Li, and M. Chowdhury, A hybrid quantum-classical AI-based detection strategy for generative adversarial network-based deepfake attacks on an autonomous vehicle traffic sign classification system, *arXiv:2409.17311*.
- [24] P. Rebentrost, M. Mohseni, and S. Lloyd, Quantum support vector machine for big data classification, *Phys. Rev. Lett.* **113**, 130503 (2014).
- [25] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, *Nature (London)* **549**, 195 (2017).
- [26] C. Ciliberto, M. Herbster, A. D. Ialongo, M. Pontil, A. Rocchetto, S. Severini, and L. Wossnig, Quantum machine learning: A classical perspective, *Proc. R. Soc. A* **474**, 20170551 (2018).
- [27] K. Beer, D. Bondarenko, T. Farrelly, T. J. Osborne, R. Salzmann, D. Scheiermann, and R. Wolf, Training deep quantum neural networks, *Nat. Commun.* **11**, 808 (2020).
- [28] M. Schuld, Supervised quantum machine learning models are kernel methods, *arXiv:2101.11020*.
- [29] C. Blank, D. K. Park, J.-K. K. Rhee, and F. Petruccione, Quantum classifier with tailored quantum kernel, *npj Quantum Inf.* **6**, 41 (2020).
- [30] C. Zoufal, A. Lucchi, and S. Woerner, Quantum generative adversarial networks for learning and loading random distributions, *npj Quantum Inf.* **5**, 103 (2019).
- [31] 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).
- [32] K. Huang, Z.-A. Wang, C. Song, K. Xu, H. Li, Z. Wang, Q. Guo, Z. Song, Z.-B. Liu, D. Zheng, D.-L. Deng, H. Wang, J.-G. Tian, and H. Fan, Quantum generative adversarial networks with multiple superconducting qubits, *npj Quantum Inf.* **7**, 165 (2021).
- [33] D. Tancara, H. T. Dinani, A. Norambuena, F. F. Fanchini, and R. Coto, Kernel-based quantum regressor models learning non-Markovianity, *Phys. Rev. A* **107**, 022402 (2023).
- [34] L. Slattey, R. Shaydulin, S. Chakrabarti, M. Pistoia, S. Khairy, and S. M. Wild, Numerical evidence against advantage with quantum fidelity kernels on classical data, *Phys. Rev. A* **107**, 062417 (2023).
- [35] M. Benedetti, J. Realpe-Gómez, and A. Perdomo-Ortiz, Quantum-assisted Helmholtz machines: A quantum-classical deep learning framework for industrial datasets in near-term devices, *Quantum Sci. Technol.* **3**, 034007 (2018).
- [36] A. Perdomo-Ortiz, M. Benedetti, J. Realpe-Gómez, and R. Biswas, Opportunities and challenges for quantum-assisted machine learning in near-term quantum computers, *Quantum Sci. Technol.* **3**, 030502 (2018).
- [37] 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, *Phys. Rev. X* **12**, 031010 (2022).
- [38] Z. Krunić, F. Flother, G. Seegan, N. Earnest-Noble, and S. Omar, Quantum kernels for real-world predictions based on electronic health records, *IEEE Trans. Quantum Eng.* **3**, 2500311 (2022).
- [39] S. Moradi, C. Brandner, M. Coggins, R. Wille, W. Drexler, and L. Papp, Error mitigation for quantum kernel based machine learning methods on IonQ and IBM quantum computers, *arXiv:2206.01573*.
- [40] B. M. Terhal, Quantum error correction for quantum memories, *Rev. Mod. Phys.* **87**, 307 (2015).
- [41] E. Knill, R. Laflamme, and L. Viola, Theory of quantum error correction for general noise, *Phys. Rev. Lett.* **84**, 2525 (2000).
- [42] E. Tang, A quantum-inspired classical algorithm for recommendation systems, in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019* (Association for Computing Machinery, New York, 2019), pp. 217–228.
- [43] T. Felser, M. Trenti, L. Sestini, A. Gianelle, D. Zuliani, D. Lucchesi, and S. Montangero, Quantum-inspired machine learning on high-energy physics data, *npj Quantum Inf.* **7**, 111 (2021).
- [44] T. Q. Duong, J. A. Ansere, B. Narottama, V. Sharma, O. A. Dobre, and H. Shin, Quantum-inspired machine learning for 6G: Fundamentals, security, resource allocations, challenges, and future research directions, *IEEE Open J. Veh. Technol.* **3**, 375 (2022).
- [45] C. Ding, T.-Y. Bao, and H.-L. Huang, Quantum-inspired support vector machine, *IEEE Trans. Neural Networks Learn. Syst.* **33**, 7210 (2022).
- [46] M. L. Wall and G. D’Aguanno, Tree-tensor-network classifiers for machine learning: From quantum inspired to quantum assisted, *Phys. Rev. A* **104**, 042408 (2021).
- [47] S. Li, M. S. Salek, Y. Wang, and M. Chowdhury, Quantum-inspired activation functions and quantum Chebyshev-polynomial network, *arXiv:2404.05901*.

- [48] P. Tiwari and M. Melucci, Towards a quantum-inspired binary classifier, [IEEE Access 7, 42354 \(2019\)](#).
- [49] Y. Chen, P. Griffin, P. Recchia, L. Zhou, and H. Zhang, Hybrid quantum neural networks with amplitude encoding: Advancing recovery rate predictions, [arXiv:2501.15828](#).
- [50] E. Hassan, M. S. Hossain, A. Saber, S. Elmougy, A. Ghoneim, and G. Muhammad, A quantum convolutional network and ResNet (50)-based classification architecture for the MNIST medical dataset, [Biomed. Signal Process. Control 87, 105560 \(2024\)](#).
- [51] K. Sun, M. Li, D. Liu, and J. Wang, IGCv3: Interleaved low-rank group convolutions for efficient deep neural networks, in [Proceedings of the British Machine Vision Conference 2018 \(BMVC 2018\), Newcastle upon Tyne, UK](#) (BMVA Press, 2018), p. 101.
- [52] Y. Lin, M. Wang, Z. Zhang, X. Wang, T. Xiao, and J. Zhu, Understanding parameter sharing in transformers, [arXiv:2306.09380](#).
- [53] N. Liu and P. Wittek, Vulnerability of quantum classification to adversarial perturbations, [Phys. Rev. A 101, 062331 \(2020\)](#).
- [54] W. Gong and D.-L. Deng, Universal adversarial examples and perturbations for quantum classifiers, [Natl. Sci. Rev. 9, nwab011 \(2022\)](#).
- [55] Y. Du, M.-H. Hsieh, T. Liu, D. Tao, and N. Liu, Quantum noise protects quantum classifiers against adversaries, [Phys. Rev. Res. 3, 023153 \(2021\)](#).
- [56] S. Lu, L.-M. Duan, and D.-L. Deng, Quantum adversarial machine learning, [Phys. Rev. Res. 2, 033212 \(2020\)](#).
- [57] I. J. Goodfellow, J. Shlens, and C. Szegedy, Explaining and harnessing adversarial examples, [arXiv:1412.6572](#).
- [58] A. Chowdhury, G. Karmakar, J. Kamruzzaman, A. Jolfaei, and R. Das, Attacks on self-driving cars and their countermeasures: A survey, [IEEE Access 8, 207308 \(2020\)](#).
- [59] Y. Deng, X. Zheng, T. Zhang, C. Chen, G. Lou, and M. Kim, An analysis of adversarial attacks and defenses on autonomous driving models, in [Proceedings of the 2020 IEEE International Conference on Pervasive Computing and Communications, PerCom, Austin, TX](#) (IEEE, Piscataway, NJ, 2020), pp. 1–10.
- [60] I. J. Goodfellow, J. Shlens, and C. Szegedy, Explaining and harnessing adversarial examples, in [Proceedings of the 3rd International Conference on Learning Representations \(ICLR, San Diego, CA, 2015\)](#).
- [61] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, Distillation as a defense to adversarial perturbations against deep neural networks, in [Proceedings of the 2016 IEEE Symposium on Security and Privacy \(SP\), San Jose, California](#) (IEEE, Piscataway, NJ, 2016), pp. 582–597.
- [62] E. Wong and Z. Kolter, Provable defenses against adversarial examples via the convex outer adversarial polytope, in [Proceedings of the 35th International Conference on Machine Learning](#), edited by J. Dy and A. Krause (PMLR, Stockholm, Sweden, 2018), Vol. 80, pp. 5286–5295.
- [63] E. A. Cherrat, I. Kerenidis, N. Mathur, J. Landman, M. Strahm, and Y. Y. Li, Quantum vision transformers, [Quantum 8, 1265 \(2024\)](#).
- [64] D. H. Wolpert, The lack of *a priori* distinctions between learning algorithms, [Neural Comput. 8, 1341 \(1996\)](#).
- [65] A. Mogelmose, M. M. Trivedi, and T. B. Moeslund, Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey, [IEEE Trans. Intell. Transp. Syst. 13, 1484 \(2012\)](#).
- [66] C. Ertler, J. Mislej, T. Ollmann, L. Porzi, G. Neuhold, and Y. Kuang, The Mapillary traffic sign dataset for detection and classification on a global scale, in [Computer Vision ECCV 2020](#), edited by A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Lecture Notes in Computer Science, Vol. 12368 (Springer, Cham, 2020), pp. 68–84.