The Dilemma of Quantum Neural Networks

Yang Qian, Xinbiao Wang, Yuxuan Du, Xingyao Wu, and Dacheng Tao, Fellow, IEEE

Abstract—The core of quantum machine learning is to devise quantum models with good trainability and low generalization error bounds than their classical counterparts to ensure better reliability and interpretability. Recent studies confirmed that quantum neural networks (QNNs) have the ability to achieve this goal on specific datasets. In this regard, it is of great importance to understand whether these advantages are still preserved on real-world tasks. Through systematic numerical experiments, we empirically observe that current QNNs fail to provide any benefit over classical learning models. Concretely, our results deliver two key messages. First, QNNs suffer from the severely limited effective model capacity, which incurs poor generalization on real-world datasets. Second, the trainability of QNNs is insensitive to regularization techniques, which sharply contrasts with the classical scenario. These empirical results force us to rethink the role of current QNNs and to design novel protocols for solving real-world problems with quantum advantages.

Index Terms—Generalization, quantum neural network (QNN), trainability, variational quantum algorithm.

I. INTRODUCTION

The theme of deep learning is efficiently optimizing a good neural network architecture with low generalization error such that it can extrapolate the routerate from the training data to new unseen data [1], [2], [3]. During the past decades, deep neural networks (DNNs) with diverse architectures have been carefully designed to accomplish different tasks with both low train and test error. Moreover, these DNNs have achieved the state-of-the-art performance compared with conventional machine learning models such as support vector machines [2]. Concrete examples include the exploitation of convolutional neural networks (CNNs) to tackle computer vision tasks [4], [5] and the employment of recurrent neural networks to solve natural language processing tasks [6], [7]. Alongside the huge empirical success of deep learning, numerous studies have been dedicated to investigating the excellent trainability and generalization ability of DNNs [8], [9], [10]. A good understanding of these two properties does not only contribute to make DNNs more interpretable but also might lead to more reliable model architecture design.

A milestone in the regime of quantum computing is Google’s experimental demonstration that modern quantum machines can solve certain computational tasks faster than classical computers [11], [12]. Such a superior power fuels a growing interest of designing quantum machine learning (QML) models, which can be effectively executed on both noisy intermediate-scale quantum (NISQ) and fault-tolerant quantum machines with provable advantages [13], [14], [15], [16], [17], [18], [19], [20]. Following this routine, the quantum neural networks (QNNs), as the quantum extension of DNNs, have been extensively investigated [21], [22], [23], [24], [25], [26], [27], [28]. Celebrated by their flexible structures, experimental studies have implemented QNNs on different NISQ platforms to accomplish various learning tasks such as data classification [24], [29], [30], image generation [31], [32], [33], and electronic-structure problems in material science and condensed matter physics [34], [35], [36], [37].

Driven by the promising empirical achievements of QML and the significance of understanding the power of QNNs, initial studies have been conducted to explore the trainability and the generalization ability of QNNs [38], [39], [40], [41], [42], [43] by leveraging varied model complexity measures developed in statistical learning theory [44] and advanced tools in optimization theory [45]. Notably, the obtained results transmitted both positive and negative signals, as shown in Fig. 1. To be more concrete, theoretical evidence validated that QNNs can outperform DNNs for specific learning tasks, i.e., quantum synthetic data classification [24] and discrete logarithm problem [42]. However, McClean et al. [46] revealed the barren plateaus’ issue of QNNs, which challenges the applicability of QNNs on large-scale problems. Considering that an ambitious aim of QNNs is providing computational advantages over DNNs on real-world tasks, it is important to answer: “are current QNNs sufficient to solve certain real-world problems with potential advantages?” If the response is negative, it is necessary to figure out “what is the gap between QNNs and DNNs?”

Problem Setup: We inherit the tradition in DNNs to understand the trainability and generalization of QNNs [47]. In particular, the explicit form of the measure of the generalization error bound is

\[
\hat{R}_S(\hat{\theta}) - R(\hat{\theta}) := \frac{1}{n} \sum_{i=1}^{n} C(h(\hat{\theta}, x^{(i)}), y^{(i)}) - \mathbb{E}_{x,y}[C(h(\hat{\theta}, x), y)] \tag{1}
\]
where $S = \{(x^{(i)}, y^{(i)})\}_{i=1}^n$ denotes the given training dataset sampled from the domain $\mathbf{X} \times \mathbf{Y}$, $h(\theta, \cdot)$ is the hypothesis space, and $\theta$ is the trained parameters, $C : \mathcal{H} \times (\mathbf{X} \times \mathbf{Y}) \to \mathbb{R}_+$ is the designated loss function, and $\hat{\mathcal{R}}(\theta)$ (or $\mathcal{R}(\theta)$) represents the empirical (expected) risk [48].

The generalization error bound in (1) concerns when and how the trainability of QNNs can be enhanced by regularization techniques measured by (3).

Through systematic numerical simulations, we empirically exhibit the dilemma of QNNs such that it is hard to directly use current QNNs to gain quantum advantages on real-world datasets. Meanwhile, current QNNs suffer from both the poor trainability and generalization ability. The main contributions of our study are given as follows.

1) We compare the performance of QNNs and DNNs on large-scale datasets in terms of (1) under both the noiseless and NISQ scenarios. Furthermore, it is crucial to understand whether the trainability of QNNs can be enhanced by regularization techniques proposed in [10]. Since the effective model capacity determined by statistical learning theory is generally tight [38], [39], [40], [42], [43], [49]. In addition, QNNs do not gain obvious trainability enhancement assisted by regularization techniques, which sharply differs from DNNs. These observations partially explain why current QNNs fail to surpass DNNs on real-world tasks.

2) We indicate the negative role of noise with respect to the generalization ability and trainability of QNNs. Specifically, quantum noise degenerates the model capacity and exacerbates the difficulty of optimization. To this end, we discuss possible solutions such as advanced error mitigation techniques to enhance the capability of QNNs on real-world datasets.

3) We build a benchmark to evaluate the performance of QNNs and DNNs on both quantum synthetic data and classical data, supporting a variety of predefined models of QNNs and providing flexible interface for
researchers to define customizable architectures. The released benchmark will facilitate the standardization of assessment of various QNNs in the QML community and provide a comparable reference in the design of QNNs. The related code is released to the Github repository: https://github.com/QQQYang/QNN-Baseline.

The remaining part of this article is organized as follows. In Section II, we briefly introduce the theory of quantum computing and three typical QNNs that will be discussed in this article: quantum naive neural network (QNNN), quantum embedding neural network (QENN), and quantum CNN (QCNN). In Sections III and IV, we thoroughly discuss the generalization ability of QNNs and trainability of QNNs, respectively. Finally, we give our conclusions in Section V.

II. PRELIMINARY

A. Quantum Computing

Analogous to the fundamental role of bit in classical computing, the fundamental unit in quantum computation is quantum bit (qubit), which refers to a 2-D vector. Under the Dirac notation, a qubit state is defined as \( |\alpha\rangle = a_0|0\rangle + a_1|1\rangle \in \mathbb{C}^2 \), where \( |0\rangle = [1, 0]^T \) and \( |1\rangle = [0, 1]^T \) specify two unit bases and the coefficients \( a_0, a_1 \in \mathbb{C} \) satisfy \( |a_0|^2 + |a_1|^2 = 1 \). Similarly, an \( N \)-qubit state is denoted by \( |\Psi\rangle = \sum_{i=1}^{2^N} a_i |e_i\rangle \in \mathbb{C}^{2^N} \), where \( |e_i\rangle \in \mathbb{R}^{2^N} \) is the unit vector whose \( i \)th entry is 1 and other entries are 0, and \( \sum_{i=0}^{2^N-1} |a_i|^2 = 1 \) with \( a_i \in \mathbb{C} \). Apart from the Dirac notation, the density matrix can be used to describe more general qubit states. For example, the density matrix corresponding to the state \( |\Psi\rangle \) is \( \rho = |\Psi\rangle \langle \Psi| \in \mathbb{C}^{2^N \times 2^N} \). For a set of qubit states \( \{p_i, |\psi_i\rangle\}_{i=1}^{m} \) with \( p_i > 0 \), \( \sum_{i=1}^{m} p_i = 1 \), and \( |\psi_i\rangle \in \mathbb{C}^{2^N} \) for \( \forall i \in m \), its density matrix is \( \rho = \sum_{i=1}^{m} p_i |\psi_i\rangle \langle \psi_i| \) and \( \text{Tr}(\rho) = 1 \).

There are three types of quantum operations used to manipulate qubit states, which are quantum (logic) gates, quantum channels, and quantum measurements. Specifically, quantum gates, as unitary transformations, can be treated as the computational toolkit for quantum circuit models, i.e., an \( N \)-qubit gate \( U \in \mathcal{U}(2^N) \) obeys \( UU^\dagger = I_{2^N} \), where \( \mathcal{U}(\cdot) \) stands for the unitary group. Throughout the whole study, we focus on the single- and two-qubit quantum gate set \{H, X, Y, Z, RX, RY, RZ, CNOT, CZ\}, as summarized in Fig. 2. Note that the investigated set is universal such that these quantum gates can be used to reproduce the functions of all the other quantum gates [50]. Different from quantum gates that evolve qubit states in the closed system, quantum channels are applied to formalize the evolution of qubits states in the open system. Mathematically, every quantum channel \( \mathcal{E}(\cdot) \) is a linear, completely positive, and trace-preserving map [50]. A special quantum channel is called depolarization channel, which is defined as

\[
\mathcal{E}_p(\rho) = (1 - p)\rho + p \frac{I_{2^N}}{2^N}.
\]

Intuitively, the depolarizing channel considers the worst case scenario such that the information of the input state can be entirely lost with some probability. The aim of quantum measurements is extracting quantum information of the evolved state, which contains the computation result, into the classical form. In this study, we concentrate on the positive operator-valued measures (POVMs), which is described by a collection of positive operators \( 0 \leq P_i \leq I \) satisfying \( \sum_i P_i = I \). Specifically, applying the measurement \( \{P_i\} \) to the state \( \rho \), the probability of outcome \( i \) is given by

\[
\text{Pr}(i) = \text{Tr}(P_i \rho).
\]

B. Quantum Neural Networks

They can be treated as the quantum generalization of DNNs, as shown in Fig. 3. Both of them leverage an optimizer to iteratively update parameters \( \theta \) of a trainable model \( h(\theta, \cdot) \) to minimize a predefined loss function \( C(\cdot, \cdot) \).

The key difference between QNNs and DNNs is the strategy to implement the trainable model \( h(\theta, \cdot) \), where the former employs the parameter quantum circuits (PQCs) or, equivalently, ansätze [51], [52], [53], while the latter utilizes neural networks [1]. In particular, PQCs are constituted by the encoding part \( U_E(\cdot) \), the trainable part \( U(\theta) \), and the readout part. The purpose of \( U_E(\cdot) \) is loading classical information into the quantum form, as the precondition to proceed further.

![Fig. 2. Quantum logic gates. The table contains the abbreviation, the mathematical form, and the graph representation of a set of universal quantum gates explored in this study.](https://github.com/QQQYang/QNN-Baseline)
quantum operators. Although there are many data encoding methods [54], here, we mainly focus on the qubit-encoding method and its variants, because of their resource-efficient property. Once the quantum example is prepared, the trainable unitary $U(\theta)$ is applied to this state, followed by the quantum measurement $\{\Pi_l\}$ to extract quantum information into the classical form (see the following for details). The collected classical information can either be used as the predicted label or the hidden feature, depending on the detailed QNN-based protocols.

In the subsequent three paragraphs, we elaborate on the implementation of three representative protocols, i.e., QNNN [24], QENN [55], and QCNN [56].

1) Quantum Naive Neural Network: We first follow (2) to elaborate on the implementation of PQCs or, equivalently, the hypothesis $h(\theta, x^{(i)})$, in QNNN. As shown in Fig. 3, the encoding circuit $U_E(\cdot)$ loads the classical example into the quantum state by specifying data features as rotational angles of single-qubit gates. Note that the topology of quantum in $U_E(\cdot)$ can be varied, e.g., a possible implementation is $U_E(x^{(i)}) = \bigotimes_{j=1}^{d_i} R_Y(x_j^{(i)})$ [55]. The trainable part $U(\theta)$ consists of trainable single-qubit quantum gates and fixed two quantum gates. Analogous to $U_E(\cdot)$, the topology of $U(\theta)$ is versatile, where involving more gates promises a higher expressivity but a more challenged trainability [49], [57]. Here, we mainly focus on the hardware-efficient structure such that the construction of $U(\theta)$ obeys a layerwise structure and the gate arrangement in each layer is identical. The explicit form satisfies $U(\theta) = \prod_{l=1}^{L} U_l(\theta_l)$, where $L$ is the layer number and $\theta_l$ denotes the trainable parameters at the $l$th layer. To extract the quantum information into the classical form, QNNN applies POVMs $\{\Pi_l\}_{l=1}^{d_i}$ to the state $|\psi(x^{(i)}, \theta)\rangle = U(\theta) U_E(x^{(i)}) |0\rangle^{\otimes d_i}$, i.e.,

$$h(\theta, x^{(i)}) = \left[ \operatorname{Tr} (\Pi_1 |\psi(x^{(i)}, \theta)\rangle \langle \psi(x^{(i)}, \theta)|), \ldots, \operatorname{Tr} (\Pi_{d_i} |\psi(x^{(i)}, \theta)\rangle \langle \psi(x^{(i)}, \theta)|) \right]^\top$$  

where $d_i$ is the dimension of the label space. In the training process, we adopt the first-order optimizer to update the parameters $\theta$ to minimize the loss function in (2), where the gradients $\frac{\partial}{\partial \theta} C(h(\theta, x^{(i)}), y^{(i)})/\delta \theta$ can be analytically evaluated by the parameter shift rule [25].

2) Quantum Embedding Neural Network: Instead of separating the encoding part from the training part, QENN integrates them together into the embedding circuit where the encoding circuit and the training circuit are carried out alternately. Specifically, in QENN, the employed PQCs are composed of multiple embedding layers equipped with trainable parameters, i.e., in each layer, an encoding circuit $U_E^{(l)}(x^{(i)})$ is followed by a trainable circuit $U_l(\theta_l)$, as shown in Fig. 3. Throughout the whole study, we consider an identical topology of $U_E^{(l)}(\cdot)$ for different layers, as the repetition of embedding layer is demonstrated to implement the classically intractable feature maps [58] and increases the expressive power of QNN [59]. The explicit form of such PQCs can be written as $U(x, \theta) = \prod_{l=1}^{L} U_E(x^{(i)}) U_l(\theta_l)$, where the meanings of $L$ and $\theta_l$ are the same with those in QNNN. As for the training of QENN, the strategy of measurement and optimization is identical to QNNN in Section II-B1.

3) Quantum Convolutional Neural Network: CNN has demonstrated the superiority of performance in image processing tasks, due to two special local operators, i.e., convolution...
and pooling. The function of convolutional and pooling operations is extracting local features from the whole image and aggregating information from adjacent patches, respectively. Unlike CNN, QCNN [56] completes the convolutional operation by the quantum convolutional layer to pursue better learning performance. In particular, in the quantum convolutional layer, a fraction of the input image is embedded into the quantum circuit $U_E()$, interacted with PQCs $U(\theta)$, followed by the quantum measurements to extract the corresponding semantic features. As shown in Fig. 3, unlike QNNN and QENN, where the collected classical information is directly utilized as the predictable label, the output of the quantum convolutional layer is treated as a hidden feature map, which is taken as the input for the next quantum convolutional layer. After several quantum convolutional operations, a classical fully connected layer with activation function [1] acts on the extracted features to make predictions.

C. Optimization of QNNs

There are many methods to optimize (2) when the hypothesis is represented by (6), which includes the zero-order, the first-order, and the second-order optimizers [51]. For clearness, here, we concentrate on the first-order optimizer, the first-order, and the second-order optimizers [51].

Unlike SGD which chooses vanilla gradient as its optimization guidance, SQNGD employs the quantum natural gradient, a quantum analogy of natural gradient, to perform the parameter updating. While vanilla GD chooses the steepest descent direction in the $l_2$ geometry of parameter space, which has been shown to be suboptimal for the optimization of quantum variational algorithms [61], quantum natural GD works on the space of quantum states equipped with a Riemannian metric tensor (called Fubini–Study metric tensor) that measures the sensitivity of the quantum state to variations in the parameters. This method always updates each parameter with optimal step-size independent of the parameter, achieving a faster convergence than SGD. Formally, we denote the quantum state produced by the PQC as $|\psi_\theta\rangle$. The optimization rule of SQNGD involving the pseudoinverse $g^+(\theta_l)$ of the metric tensor yields

$$
\theta_{l+1} = \theta_l - \eta g^+(\theta_l) \nabla C(\theta)
$$

(9)

where $g_{ij}(\theta) = \text{Re}(G_{ij}(\theta))$ is the Fubini–Study metric tensor and $G_{ij}(\theta)$ is the quantum geometric tensor, which can be written as

$$
G_{ij}(\theta) = \left\langle \frac{\partial \psi_\theta}{\partial \theta_i}, \frac{\partial \psi_\theta}{\partial \theta_j} \right\rangle - \left\langle \frac{\partial \psi_\theta}{\partial \theta_i}, \psi_\theta \right\rangle \left\langle \psi_\theta, \frac{\partial \psi_\theta}{\partial \theta_j} \right\rangle
$$

(10)

where $\theta_l$ being the $l$th entry of parameter vector $\theta$.

III. GENERALIZATION OF QNNs

In this section, we explore the generalization ability of representative QNNs introduced in Section II. To be more concrete, we first apply these QNNs to learn real-world datasets and compare their performance with classical DNNs with varied parameter settings, i.e., the multilayer perceptron (MLP) and CNN whose number of trainable parameters is similar to QNNs and the overparameterized multilayer perceptron (MLP++) whose number of trainable parameters can be extremely large [5], [62]. We further conduct systematical simulations to benchmark the effective model capacity of QNNs and DNNs since this metric determines the generalization ability of learning models [2], [10].

A. Datasets

The selection of experimental datasets obeys the following rules. First, the candidate datasets are expected to be sufficiently diverse to cover both quantum data and classical data, which can characterize the overall performance of QNNs in manipulating quantum and classical tasks. Second, the datasets should involve multiple applications, such as image classification. Third, the selected datasets must be compatible with available computational resources. Specifically, current quantum simulators can only perform tasks within tens of qubits.

We benchmark the generalization ability of QNNs to process quantum data by accomplishing the classification task for quantum synthetic dataset proposed in [24]. Concretely, we randomly sample classical data from uniform distribution and then embed them into the quantum circuit by assigning each classical sample to the rotation angle of each quantum rotation gate. After converting classical data $x$ to quantum state...
\( \rho(x) \), we run the quantum circuit and measure the expectation of observable \( O \). The whole process is expressed as

\[
f_\theta(x) = \langle 0 | U^\dagger(x, \theta) O U(x, \theta) | 0 \rangle
\]

where \( U \) denotes the quantum circuit, which depends on classical sample \( x \) and trainable parameter \( \theta \). For binary classification task, the label \( y \) of input \( x \) is calculated by \( \text{sign}(f_\theta(x)) \). We totally collect 200 positive samples and 200 negative samples and split them into the training set and test set equally.

With the aim of removing data-dependent bias as much as possible, we also experiment on two real-world datasets, i.e., the Wine dataset [63] and MNIST handwritten digit dataset [64], to examine the generalization ability of QNNs and DNNs. Larger datasets, such as ImageNet, are excluded as possible, we also experiment on two real-world datasets, the Wine dataset and the MNIST dataset, respectively.

Table I shows the statistical characteristics of three datasets. Specifically, we train QNNN, QCNN, and MLP on the Wine dataset and quantum synthetic dataset that represent 1-D features and apply QCNN, CNN, and MLP on the MNIST dataset for the case of 2-D features. Note that QNNN and QCNN are excluded when processing image data because it requires an unaffordable number of qubits when embedding the high-dimensional image into a quantum circuit. For suppressing the effects of randomness, the statistical results are collected by repeating each setting ten times.

### B. Model Architecture

To make a fair comparison between QNNs and DNNs, we employ QNNN, QCNN, and their classical counterpart MLP and CNN to handle different tasks. In addition, we consider QENN to cover different data encoding styles in QNNs. In the following, we will give a detailed description about the architecture and parameter counts of adopted models.

1) **Quantum Naive Neural Network:** It is roughly divided into two blocks, including the feature embedding block and trainable measurement block. In this article, the feature embedding block converts the classical data into quantum state by setting the parameter of quantum gates with classical data.

The measurement block consists of a variational quantum circuit that linearly transforms the prepared quantum state and the measurement operation that calculates the expectation of Pauli Z basis. The number of qubits in experiments depends on the feature dimension of datasets. Concretely, we employ quantum circuits with 16, 13, and four qubits for quantum synthetic data, the Wine dataset, and the MNIST dataset, respectively.

2) **Quantum Embedding Neural Network:** The basic architecture of QENN model is the same as that of QNNN. What is different is that the embedding layer also contains trainable parameters. By making the feature embedding learnable, the quantum circuit forms a more flexible and complicated transform function, which is more powerful than the vanilla QNNN.

3) **Quantum Convolutional Neural Network:** The QCNN employed in this article can be regarded as a quantum version of classical CNN, where the classical convolutional kernel is replaced with quantum circuit. To be concrete, we implement a 2 \( \times \) 2 kernel with a variational quantum circuit of four qubits and six trainable parameters. In addition, multiple duplicates of the quantum circuit are introduced to simulate the multichannel mechanism. The feature maps returned by quantum kernels are further processed by two fully connected layers of 32 and ten nodes.

4) **Convolutional Neural Network:** The structure of CNN adopted in the experiments is the same as that of QCNN, with the quantum convolutional kernels substituted by classical convolutional kernels and other components unchanged.

5) **Multilayer Perceptron:** It is constructed by sequentially connecting multiple fully connected layers one by one. When processing quantum synthetic data and the Wine data, we adopt a three-layer MLP, with the dimension of hidden layer depending on the limitation of total number of trainable parameters. For the MNIST dataset, the original 2-D image is first flattened into a 1-D vector, which is then input into a three-layer MLP with 32 hidden nodes. Therefore, the MLP designed for MNIST replaces the convolution layer with the fully connected layer. In this way, we can observe how quantum convolution affects the training process.

For a fair comparison, the number of trainable parameters of different models for the same task is kept close and the structure-independent hyperparameters during training are not fine-tuned specifically for each model but always keep the same setting for one task. Meantime, the experiment with a specific configuration is repeated ten times and calculates the average as final experiment result. The detailed parameter count for each model and the rigorous experiment setup are listed in Tables II and III, respectively.
C. Results

The QNNs referred in this article are implemented based on PennyLane [65] and PyTorch [66]. Due to the limited accessibility of physical quantum computers, all experiments are simulated on classical computers with Intel® Xeon® Gold 6267C CPU @ 2.60 GHz and 128-GB memory. To simulate the quantum noise, we adopt the noise model from ibmq_16_melbourne, which is one of the IBM Quantum Canary Processors [67].

Before moving on to present experiment results, let us address the generalization error measure defined in (1). In particular, there are two components that together completely characterize the generalization error, i.e., the empirical risk $\hat{R}_S(\hat{\theta})$ and the expected risk $R(\hat{\theta})$. In our experiments, we employ the accuracy of the training set to quantify the empirical risk in which high train accuracy reflects low $\hat{R}_S(\hat{\theta})$. Meanwhile, following the explanation in (1), the accuracy on the test set is adopted to estimate the expected risk such that high test accuracy implies low $R(\hat{\theta})$. Under the above insights, when a learning model possesses a good generalization ability, it should achieve high train accuracy and test accuracy, as well as a small gap between them.

The learning performance of QNNs and DNNs for the quantum synthetic dataset and real-world datasets is shown in Fig. 4. Toward the quantum synthetic dataset, both the train and test accuracy of QNNN and QENN fast converge to 92% after 20 epochs. Conversely, although the train accuracy of MLP reaches 90% after 30 epochs, its test accuracy is no better than the random guess. Therefore, the generalization error of classical DNNs, i.e., the discrepancy between train accuracy and test accuracy, is much higher than that of quantum models (0.4 for MLP versus 0.01 for QNNN and QENN), as shown in the bar chart in Fig. 4(d). This advantage of QNNs on quantum synthetic data is explained in [42], which claims that large geometric difference between a classical ML model and a quantum ML model means that there exists a quantum dataset with obvious quantum advantages. The quantum synthetic dataset constructed in our experiments follows this
Fig. 5. Learning performance on quantum data and classical data with random labels. (a) How various models fit quantum data with random labels. MLP++, representing MLP with larger scale, achieves zero training error. (b) Changes of accuracy when fitting classical data with random labels (Wine). MLP can still completely fit the random labels. (c) Ability of fitting MNIST with random labels. No model performs better than random guess.

idea and hence leads to a better generalization performance of QNNs than that of DNNs. In addition, as envisioned by the statistical learning theory, such a quantum advantage can be understood through the lens of PAC learning. As shown in Fig. 1, the quantum synthetic dataset produced by quantum circuits that are hard to simulate classically corresponds to the target concepts $H_Q \backslash H_D$, leading to the quantum advantage on generalization error. However, the learning performance behaves quite different when the above models are applied to learn real-world datasets. As shown in Fig. 4(b), there exists an evident step-by-step accuracy dropping on the Wine dataset along the sequence of MLP, QENN, and QNNN. In particular, QNNN and QENN fall behind MLP by 10%–20%. Meantime, there is a more serious performance degradation for quantum models evaluated by test accuracy, especially for QNNN, which holds almost 15% generalization error that is three times higher than that of MLP. The learning performance of QCNN and CNN on the MNIST dataset obeys the same manner. As shown in Fig. 4(c), QCNN achieves 93% accuracy on both training and test sets, which is slightly worse than CNN by approximately 3%. It is worth noting that the relatively small gap among three models is most attributed to the subtle differences in the network structure, where QCNN, CNN, and MLP only differ in the first layer.

The generalization ability of a learning model is dominated by its effective model capacity, which concerns the model’s ability to fit random labels [10]. Namely, a learning model possesses a high effective model capacity when it reaches a high train accuracy on a dataset with random labels, as ensured by the randomization test in nonparametric statistics. Empirical studies have validated that DNNs possess sufficient effective model capacity and speculated that this property contributes to the great success of DNNs. With this regard, we conduct randomization experiments on both QNNs and DNNs to compare their effective model capacity.

The results related to the effective model capacity are shown in Fig. 5, which reveal a large gap between QNNs and DNNs, regardless of whether the training data are quantum or classical. In particular, QNNs achieve relatively low train accuracy (0.562 for QNNN and 0.630 for QENN), which is only slightly better than random guess. By contrast, for DNNs, MLP reaches 90% and the perfect 100% train accuracy on the quantum synthetic dataset and Wine dataset, respectively, after 40 epochs. If we further increase the number of trainable parameters of MLP (MLP++), it will also completely fit the quantum synthetic data with random labels, as shown by the purple line in Fig. 5(a). Note that the same strategy is inappropriate to be applied to QNNs because increasing trainable parameters will incur both the barren plateau issues and the accumulated quantum noise, which hinder the optimization [41]. As for MNIST, all these models fail to fit the random labels, whose behavior imitates the random guess. Similarly, we can enlarge the scale of MLP to obtain a leap of accuracy on training set with random labels at the expense of little growth of training time, as shown by the purple line in Fig. 5(c).

In addition to the noiseless scenario, we explore the generalization ability of QNNs under the NISQ scenario. The achieved results are shown in Fig. 6. In particular, the solid and dashed lines describe the training process on noiseless and NISQ devices, respectively. An immediate observation is that quantum system noise largely weakens the effective model capacity and generalization of QNNs, leading to a severe accuracy drop (from about 94% to 80%) on quantum synthetic data. Furthermore, the noisy QNNs seem to completely lose the ability of matching the random data.

D. Implications

The achieved results indicate the following three substantial implications with respect to the dilemma of existing QNNs.

1) The learning performance of current QNNs is no better than DNNs on real-world datasets. This observation questions the necessity to employ QNNs to tackle real-world learning tasks since it remains elusive how QNNs can benefit these tasks.

2) The effective model complexity of current QNNs is poor, which is stark contrast with DNNs. The low effective
model complexity enables us to leverage statistical learning theory to analyze the generalization ability of QNNs with a tight bound [39], [42]. Nevertheless, as shown in Fig. 1, a severely restricted model capacity fails to cover complicated target concepts in real-world tasks, which prohibits the applicability of QNNs.

3) The limited model capacity is further reduced by imperfection of NISQ machines. The narrowed hypothesis space deteriorates the performance of QNNs.

There are two possible directions to seek potential advantages of QNNs over DNNs. The first way is designing clever overparameterization quantum learning models as with DNNs. Partial evidence to support this solution is the improved performance of QENN compared with QNNN. A critical issue in such a model design is how to avoid barren plateau phenomenons [46]. The second way is to develop a new paradigm of quantum models to further introduce nonlinearity into quantum learning pipeline. For instance, theoretical results have proven potential advantages of quantum kernels [24], [68], [69].

IV. TRAINABILITY OF QUANTUM MODELS

Here, we investigate the trainability of QNNs, which serves as another dominant factor manipulating the learnability of quantum models. In particular, we first examine the performance of QNNN and QENN on the Wine dataset with consideration of various implicit and explicit regularization techniques under the noiseless scenario. Subsequently, for the purpose of understanding how quantum noise affects the performance of QNNs, we conduct the same experiments under the noisy scenario in which the noisy model is extracted from ibmq_16_melbourne, which is one of the IBM Quantum Canary Processors [67].

A. Implicit Regularization

Recall that a learning model, e.g., QNNs or DNNs, is warranted to possessing a good trainability if it requires a small number of epochs to surpass a threshold accuracy and fast converges to a stationary point, i.e., the term $\mathcal{J}(\theta) \to 0$ in (3).

Fig. 6. Quantum noise reduces quantum model capacity. FT denotes the fault-tolerant devices, TL is the abbreviation of “true label,” and RL is the abbreviation of “random label.” When running the same quantum circuit on noisy devices, there is a serious performance decline in terms of the ability of learning the original data and fitting random label.

Fig. 7. Effects of regularizations on the performance of quantum model on the Wine dataset. The labels “GD,” “SGD,” “SQNGD,” “WD,” and “N” refer to the gradient descent optimizer, the SGD optimizer, the stochastic quantum natural gradient descent optimizer, the weight decay, and the execution of experiments on NISQ chips, respectively. (a) Effects of regularizations on optimization. SGD plays a significant role in accelerating convergence and achieving higher accuracy, while others play a significant role in the optimization process instead of boosting performance. (b) Learning curve of GD with more training epochs.

Many theoretical studies in the regime of deep learning have proven that SGD can help learning models escape saddle points efficiently [70], [71]. Driven by the success of SGD in deep learning, an immediate interest is exploring whether this method can enhance the trainability of QNNs, especially for alleviating barren plateaus issues.

The experiment results are summarized in Fig. 7. With the aim of investigating whether SGD facilitates the quantum model optimization, we also apply the GD optimizer to learn the same tasks as a reference. Specifically, Fig. 7(a) shows that the train accuracy of QNNN and QENN optimized by SGD rapidly rises to 70% after 20 epochs, while its train accuracy remains at around 50% with the GD optimizer. Meanwhile, QENN with SGD optimizer achieves higher test accuracy than that in the setting of GD (at least 20%). Notably, the performance of QENN with GD optimizer presents a monotone increasing trend, which suggests that the model may potentially reach higher performance with more training epochs.

Therefore, we extend the total training epochs from 100 to 500 and train the QENN with GD. Fig. 7(b) shows that the accuracy of QENN trained by GD has been increasing smoothly and reaches 80% after 500 epochs, narrowing the gap between GD and SGD from 30% to 10%. Nevertheless,
the SGD training accuracy is around 90% after the 100th epoch, and the test accuracy of SGD reaches 88% at the 150th epoch and fluctuates slightly in the later training, which reveals that SGD can achieve fast and stable convergence for QNNs. We also explore the performance of QNNs under the NISQ scenario. As shown by the brown line in Fig. 7, quantum noise leads to the degraded trainability of QNNs, i.e., around 10% accuracy decline compared with noiseless settings.

Motivated by the large gain from SGD, we conduct additional experiments to exploit how the batch size affects the learning performance of QNNs. Specifically, we train QNNs many times on the Wine data by SGD with batch size growing exponentially. As shown in Fig. 8, for both QNNN and QENN, the increased batch size suppresses the convergence rate. For example, QNNN achieves 80% accuracy on the Wine dataset when the batch size is equal to 4 after 40 epochs, which is 10% higher than that of batch size with 8. These experimental results accord with the conclusions developed in empirical risk minimization, where in training discriminative learning models (including both DNNs and QNNs), the larger batch size of SGD implies the worse convergence [72], [73]. One possible reason is pointed out in [73], which states that large-batch SGD tends to converge to sharp minimizers and hence leads to poor performance.

B. Explicit Regularization

We then study how the explicit regularization technique, i.e., the weight decay, affects the trainability of QNNs. Mathematically, the weight decay strategy refers to adding a penalty term for the trainable parameters, i.e., \( \arg\min_\theta L(\theta) = (1/n) \sum_{i=1}^n \ell(y^{(i)}, \hat{y}^{(i)}) + \lambda \|\theta\| \). The effect of weight decay on the trainability of QNNs is shown in Fig. 7. An immediate observation is that this strategy fails to enhance the performance of QNNN with a GD optimizer. With respect to the SGD optimizer, the weight decay method improves the test accuracy of QNNN by 2% at the expense of a slightly low convergence rate. For QENN, weight decay together with SGD assists models to obtain the fastest convergence in the first 20 epochs. However, it fails to efficiently narrow the gap between train accuracy and test accuracy after the 40th epoch when the phenomenon of over fitting begins to appear.

C. Optimizers

Another potential factor that influences the trainability of QNNs is the choice of optimizers. In this work, we consider four typical optimizers that perform well in classical models, e.g., SGD, adaptive moment (Adam) [74], root-mean-squared propagation (RMSprop) [75], and Nesterov momentum [76]. Moreover, a recent study demonstrated that a quantum variant of SGD, i.e., SQNGD [60], can well address the barren plateau’s issue. To figure out how these techniques contribute to the optimization of QNNs, we train QNNN and QENN for Wine classification with various optimizers. As shown in Fig. 9, both QNNN and QENN achieve a faster convergence when optimized with gradient momentum, such as Adam, RMSprop, and Nesterov momentum. After 20 epochs, Adam, RMSprop, and Nesterov momentum assist QNNN to reach 85% train accuracy, which is nearly 10% higher than that of QNNN optimized by SGD. Surprisingly, SQNGD, the quantum natural gradient version of SGD, further expands the accuracy gap by 5%, reaching the highest accuracy and fastest convergence of QNNN. Although the introduction of quantum geometry information significantly benefits the training of QNNs, the computational overhead of geometry metric tensor makes the overall training time grow quickly.

D. Implications

The achieved results indicate that the widely used regularization techniques in classical deep learning play a different
role in QML. Although SGD with the appropriate batch size slightly benefits the optimization of QNNs, other regularization strategies, such as weight decay, fail to enhance the trainability of QNNs. This differs QNNs with DNNs. Advanced techniques are highly desired to improve the trainability of QNNs, especially addressing the barren plateaus phenomena. In addition, empirical results exhibit that quantum system noise exacerbates the training difficulty of QNNs. A promising way to resolve this issue is introducing various error mitigation techniques into QNNs [13], [77], [78], [79], [80], [81].

### V. Conclusion

In this study, we proceed with systematic numerical experiments to understand the generalization ability and trainability of typical QNNs in view of statistical learning theory. The achieved results exhibited that current QNNs struggle with a poor effective model capacity. As shown in Fig. 1, this observation well explains why current QNNs attain computational advantages on quantum synthetic data classification tasks and discrete logarithm problems, while they fail to compete with DNNs in tackling real-world learning tasks. Moreover, our study illustrates that the regularization techniques, which greatly contribute to the success of DNNs, have limited effects toward the trainability of QNNs. In addition, our study exhibits that quantum system noise suppresses the learnability of QNNs, which echoes with the theoretical study [49]. Finally, to alleviate the dilemma of current QNNs, we discuss several prospective directions such as designing overparameterized QNNs without barren plateaus and developing effective error mitigation techniques.

Besides the contributions toward understanding the power of QNNs, we build an open-source benchmark to fairly and comprehensively assess the learnability of various QNNs in a standard process and consequently benefit the design of new paradigms of QNNs. Specifically, this benchmark provides several ready-to-use datasets, quantum and classical models, as well as evaluation scripts. Furthermore, we adopt the factory method in the software design to help users easily register their self-defined models into the whole framework. More models and tasks will be supported in the future. We believe that this benchmark will facilitate the development of whole QML community.

**Note Added:** During the preparation of this article, we notice that a very recent theoretical study [82] indicated that to deeply understand the power of QNNs, it is necessary to demonstrate whether QNNs possess the ability to achieve zero risk for a randomly relabeled real-world classification task. Their motivation highly echoes with our purpose such that statistical learning theory can be harnessed as a powerful tool to study the capability and limitations of QNNs. From this perspective, the achieved results in this study provide a negative response to their question. Combining the analysis in [82] and our results, a promising research direction is analyzing the nonuniform generalization bounds of QNNs to understand their power.

### References

[1] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.

[2] M. Mohri, A. Rostamizadeh, and A. Talwalkar, Foundations of Machine Learning (Adaptive Computation and Machine Learning), 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[3] V. Vapnik, The Nature of Statistical Learning Theory. Cham, Switzerland: Springer, 2013.

[4] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 4700–4708.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Proc. Adv. Neural Inf. Process. Syst. (NIPS), vol. 25. Stateline, NV, USA, Dec. 2012, pp. 1097–1105.

[6] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997.

[7] M. Schuster and K. K. Paliwal, “Bidirectional recurrent neural networks,” IEEE Trans. Signal Process., vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[8] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srivastava, “Exploring generalization in deep learning,” in Proc. 31st Int. Conf. Neural Inf. Process. Syst. (NIPS). Red Hook, NY, USA: Curran Associates, 2017, pp. 5949–5958.

[9] R. Sun, “Optimization for deep learning: Theory and algorithms,” 2019, arXiv:1912.08957.

[10] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, “Understanding deep learning requires rethinking generalization,” in Proc. 5th Int. Conf. Learn. Represent. (ICLR). Toulon, France, Apr. 2017, pp. 1–15. [Online]. Available: https://openreview.net/forum?id=Sy8gdB9xx

[11] F. Arute et al., “Quantum supremacy using a programmable superconducting processor,” Nature, vol. 574, no. 7779, pp. 505–510, 2019.

[12] J. Preskill, “Quantum computing in the NISQ era and beyond,” Quantum, vol. 2, p. 79, Aug. 2019.

[13] K. Bharti et al., “Noisy intermediate-scale quantum algorithms,” Rev. Modern Phys., vol. 94, no. 1, 2022, Art. no. 015004.

[14] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, “Quantum machine learning,” Nature, vol. 549, no. 7671, p. 195, 2017. [Online]. Available: https://www.nature.com/articles/nature23474

[15] M. Ceruzo, A. Poremba, L. Cincio, and J. P. Coles, “Variational quantum fidelity estimation,” Quantum, vol. 4, p. 248, Mar. 2020.

[16] A. W. Harrow and A. Montanaro, “Quantum computational supremacy,” Nature, vol. 549, no. 7671, p. 203, 2017.

[17] M. Schuld, I. Sinayskiy, and F. Petruccione, “The quest for a quantum neural network,” Quantum Inf. Process., vol. 13, no. 11, pp. 2567–2586, Nov. 2014, doi: 10.1007/s11128-014-0809-8.

[18] A. Galindo and M. A. Martin-Delgado, “Information and computation: Classical and quantum aspects,” Rev. Modern Phys., vol. 74, no. 2, p. 347, 2002.

[19] P. A. M. Casares and M. A. Martin-Delgado, “A quantum active learning algorithm for sampling against adversarial attacks,” New J. Phys., vol. 22, no. 7, Jul. 2020, Art. no. 073026.

[20] G. D. Paparo, V. Dunjko, A. Makmal, M. A. Martin-Delgado, and H. J. Briegel, “Quantum speedup for active learning agents,” Phys. Rev. X, vol. 4, no. 3. Jul. 2014, Art. no. 031002.

[21] K. Beer et al., “Training deep quantum neural networks,” Nature Commun., vol. 11, no. 1, pp. 1–6, 2020.

[22] I. Cong, S. Choi, and M. D. Lukin, “Quantum convolutional neural networks,” Nature, vol. 15, no. 12, pp. 1273–1278, 2019.

[23] E. Farhi and H. Neven, “Classification with quantum neural networks on near term processors,” 2018, arXiv:1802.06002.

[24] V. Havlíček et al., “Supervised learning with quantum-enhanced feature spaces,” Nature, vol. 567, no. 7747, p. 209, 2019.

[25] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, “Quantum circuit learning,” Phys. Rev. A. Gen. Phys., vol. 98, no. 3, Sep. 2018, Art. no. 032309.

[26] M. Schuld and N. Killoran, “Quantum machine learning in feature Hilbert spaces,” Phys. Rev. Lett., vol. 122, no. 4, Feb. 2019, Art. no. 040504.

[27] N. H. Nguyen, E. C. Behrman, M. A. Moustafa, and J. E. Steck, “Benchmarking neural networks for quantum computations,” IEEE Trans. Neural Netw. Learn. Syst., vol. 31, no. 7, pp. 2522–2531, Sep. 2019.

[28] J. Bausch, “Recurrent quantum neural networks,” in Proc. Adv. Neural Inf. Process. Syst., vol. 33. Cambridge, MA, USA: MIT Press, 2020, pp. 1368–1379.

[29] T. Kasumoto, K. Mitarai, K. Fuji, M. Kitagawa, and M. Negoro, “Experimental quantum machine learning with nuclear spins in a solid,” 2019, arXiv:1911.12021.
A. Abbas, D. Sutter, C. Zoufal, A. Lucchi, A. Figalli, and S. Woerner, \"Optimizing quantum machine learning,\" in Proc. Adv. Neural Inf. Process. Syst., vol. 35, no. 4, Apr. 2024, Art. no. 01022.

A. R. Benning, G. A. Ford, and Z. A. Pan, \"Deriving quantum algorithms from quantum chemistry,\" in Proc. Adv. Neural Inf. Process. Syst., vol. 35, no. 4, Apr. 2024, Art. no. 01023.

A. Peruzzo et al., \"Quantum approximate optimization is computationally universal,\" 2018, arXiv:2001.02138.

A. S. S. Boyd and L. Vandenberghe, \"Convex Optimization,\" Cambridge U.K.: Cambridge Univ. Press, 2004.

A. B. Coyle, \"The power of data in quantum machine learning,\" in Proc. Conf. Learn. Theory, 2015, pp. 797–842.

A. Ge, F. Huang, C. Jin, and Y. Yuan, \"Scalable and robust algorithms for generative adversarial networks,\" Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2021, Art. no. 040321.

B. He, X. Zhang, S. Ren, and J. Sun, \"Deep residual learning for image classification,\" in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 770–778.

B. H. Li, J. M. Gambetta, and A. Perdomo-Ortiz, \"Generation of high-resolution handwritten digits with an ion-trap quantum computer,\" Phys. Rev. X, vol. 5, no. 10, Oct. 2019, Art. no. eaaw9918.

C. Hempel et al., \"Quantum chemistry calculations on a trapped-ion quantum simulator,\" Phys. Rev. X, vol. 8, no. 3, Jul. 2018, Art. no. 031022.

C. Hernandez, S. Shaya, S. Pradhan, and T. Cook, \"Quantum neural networks: Powering image recognition with quantum circuits,\" Quantum Mach. Intell., vol. 2, no. 1, pp. 1–9, Jun. 2020.

D. A. Hestenes and W. E. Diffenderfer, \"Lagrange multipliers in the calculus of variations,\" SIAM Rev., vol. 11, no. 3, Jul. 1969, pp. 164–168.

D. Dua and C. Graff, \"UCI Machine Learning Repository. [Online]. Available: http://archive.ics.uci.edu/ml\"
Yang Qian received the B.S. degree from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2016, and the M.S. degree from the Institute of Automation, Chinese Academy of Sciences (CASIA), Beijing, China, in 2019. He is currently pursuing the Ph.D. degree with the School of Computer science, The University of Sydney, Sydney, NSW, Australia, under the supervision of Prof. Dacheng Tao. His current research interests include deep learning, quantum machine learning, and distributed optimization.

Xinbiao Wang received the B.S. degree from Lanzhou University, Lanzhou, China, in 2018, and the M.S. degree from Wuhan University, Wuhan, China, in 2021, where he is currently pursuing the Ph.D. degree. His current research interests include quantum machine learning and quantum learning theory.

Yuxuan Du received the Ph.D. degree from the School of computer science, The University of Sydney, Sydney, NSW, Australia, in 2021. He is currently a Senior Researcher with the JD Explore Academy, Beijing, China, and a member of Doctor Management Trainee at JD.com, Beijing. He has published his research outcomes in many top-tier journals and conferences in physics and computer science, including Physical Review Letters, Physical Review X Quantum, IEEE TRANSACTIONS ON INFORMATION THEORY, and Conference on Computer Vision and Pattern Recognition. His research interests include fundamental algorithms for quantum machine learning, quantum learning theory, and quantum computing.

Dacheng Tao (Fellow, IEEE) is currently an Advisor and a Chief Scientist with the Digital Science Institute, The University of Sydney, Sydney, NSW, Australia; the Director of the JD Explore Academy, Beijing, China; and the Vice President of JD.com, Beijing. He mainly applies statistics and mathematics to artificial intelligence and data science, and his research is detailed in one monograph and over 200 publications in prestigious journals and proceedings at leading conferences.

Xingyao Wu received the B.S. degree from the University of Science and Technology of China, Hefei, China, in 2012, and the Ph.D. degree from the National University of Singapore, Singapore, in 2017. He is currently working as a Quantum Scientist at the JD Explore Academy, Beijing, China. His current research interests include quantum algorithms, quantum software, and noisy intermediate-scale quantum (NISQ) quantum applications.