IOP Publishing New J. Phys. 26 (2024) 093037 https://doi.org/10.1088/1367-2630/ad6f3d

New Journal of Physics
The open access journal at the forefront of physics

Deutsche Physikalische Gesellschaft DPG
IOP Institute of Physics

Published in partnership with: Deutsche Physikalische Gesellschaft and the Institute of Physics

OPEN ACCESS

RECEIVED
29 June 2024

REVISED
4 August 2024

ACCEPTED FOR PUBLICATION
13 August 2024

PUBLISHED
25 September 2024

PAPER

Gate-based quantum neurons in hybrid neural networks

Changbin Lu[1,2], **Mengjun Hu**[2], **Fuyou Miao**[3,4] and **Junpeng Hou**[5,*]

[1] School of Computer Science and Technology, Anhui University of Technology, Maanshan 243002, People's Republic of China
[2] Beijing Academy of Quantum Information Sciences, Beijing 100193, People's Republic of China
[3] School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, People's Republic of China
[4] Hefei National Laboratory, University of Science and Technology of China, Hefei 230028, People's Republic of China
[5] Pinterest Inc., San Francisco, CA 94103, United States of America
[*] Author to whom any correspondence should be addressed.

E-mail: jhou@pinterest.com

Keywords: quantum neurons, hybrid neural networks, quantum deep neural networks, quantum expressibility, quantum ansatz

Abstract

Quantum computing is conceived as a promising and powerful next-generation platform for information processing and it has been shown that it could bring significant accelerations to certain tasks, compared to its classical counterparts. With recent advances in noisy intermediate-scale quantum (NISQ) devices, we can process classical data from real-world problems using hybrid quantum systems. In this work, we investigate the critical problem of designing a gate-based hybrid quantum neuron under NISQ constraints to enable the construction of scalable hybrid quantum deep neural networks (HQDNNs). We explore and characterize diverse quantum circuits for hybrid quantum neurons and discuss related critical components of HQDNNs. We also utilize a new schema to infer multiple predictions from a single hybrid neuron. We further compose a highly customizable platform for simulating HQDNNs via Qiskit and test them on diverse classification problems including the iris and the wheat seed datasets. The results show that even HQDNNs with the simplest neurons could lead to superior performance on these tasks. Finally, we show that the HQDNNs are robust to certain levels of noise, making them preferred on NISQ devices. Our work provides a comprehensive investigation of building scalable near-term gate-based HQDNNs and paves the way for future studies of quantum deep learning via both simulations on classical computers and experiments on accessible NISQ devices.

1. Introduction

Quantum computing is an emerging field that leverages quantum-mechanical properties to perform computations. Unlike classical computers, which operate on bits that can only assume binary states, quantum computers utilize qubits that can be in a superposition of multiple states simultaneously, which may also lead to bizarrely entangled states with 'spooky action at a distance'. These unique features allow quantum computers to solve certain computational problems significantly faster than classical computers [1], with potential applications in a wide range of areas such as cryptography [2, 3], chemistry [4], and optimization [5]. Ongoing research in this field aims to further develop quantum algorithms and hardware, as well as explore new applications that could benefit from quantum computing's unique capabilities. So far, noisy intermediate-scale quantum (NISQ) technology is the most promising candidate to bring near-term quantum impact [6] and has been demonstrated experimentally in vast scale [7–12].

On the other hand, machine learning is a rapidly evolving field within computer science that involves the development of algorithms and statistical models capable of analyzing data to uncover underlying patterns and relationships. Through these analyses, machine learning models can learn and make predictions or decisions about new, previously unseen data. Applications of machine learning are diverse and range from image recognition and natural language processing to recommender systems and fraud detection [13], which is supported by a plethora of intricate model architectures including deep neural network (DNN), convolutions neural network [14–16], transformers [17–19], diffusion models [20–22], etc. Most recently,

large pre-trained language models like ChatGPT [19] have shown surprising in-context learning ability [23, 24], bringing even more possibilities to the future of artificial intelligence. With the continued growth of large datasets, model capabilities, and computing resources, machine learning has the potential to revolutionize many fields in science, industry, and society [25].

Quantum machine learning combines the principles of quantum computing with machine learning algorithms, which is currently in its initial exploratory stages [26–30]. By leveraging the power from the micro-world on the scale of a single atom, quantum machine learning could bring advantages like solving problems that are intractable for classical computers, being able to learn from few data, etc [31–36]. Current approaches to building a quantum DNN (QDNN) include variational quantum circuits [37–40] and quantum simulation of classical neurons [41]. Among different proposals, it is common to use the continuous-variable architecture, which encodes quantum information in continuous degrees of freedom such as the amplitudes of the electromagnetic field [42]. To better utilize the current NISQ hardware, some hybrid computing frameworks were studied and showed promising results in different contexts including hybrid QDNN (HQDNN) and hybrid quantum algorithms [43–47]. However, there lacks a systematic study of HQDNN and it is unclear how to make the choices, among countless combinations of quantum circuits, to build a hybrid neuron effectively and efficiently. Not to mention how to characterize them.

In this work, we attempt to address the above critical problems by investigating a class of hybrid neurons composed of three components including a feature map, a neuron circuit, and a measurement process. Our goal is to provide comprehensive guidance for designing scalable HQDNNs rather than showcasing the quantum advantage of specific models on specific tasks. We distinguish the neurons' hybrid expressibility in the context of hybrid quantum computing, rather than the full quantum expressibility, and find those hybrid neurons with simple neuron circuits could have good hybrid expressibility even though their quantum expressibility is low. This leads to the observation that, unlike QDNNs, HQDNNs with simple and shallow neuron circuits can also exhibit superior performance. This suggests that HQDNNs are much easier to engineer and manipulate on NISQ platforms, making it possible to build large-scale DNNs with quantum features.

To demonstrate this, we further discuss other crucial components of an HQDNN including predictions, cost functions, forward/backward propagation, and optimizers. While those are inherently similar to their classical counterparts, we introduce a new schema to make multiple predictions out of a single hybrid neuron by taking advantage of its quantum characters. We then compose a platform for simulating HQDNNs via Qiskit [48], where we can easily customize the neuron circuit, network architecture, prediction schema, and optimizers to conduct experiments. We choose the iris dataset, which is a popular test set for studying QDNNs given the current limitation of classical simulation of general quantum dynamics, and a slightly harder wheat seed dataset. We carefully examine different setups on each dataset and the results suggest that HQDNNs with simple neuron circuits can achieve competitive performance while requiring much fewer quantum resources. Finally, we model the NISQ devices with Gaussian noises and show that the proposed HQDNNs are indeed robust to a certain level of noise, making them of particular interest in the NISQ era.

2. Hybrid quantum neurons

A neuron is the most critical building block of modern DNNs and thus, the first question to ask here is how to choose a proper hybrid quantum neuron and if there is useful guidance for us to explore possible configurations for hybrid neurons. So far, gated-based quantum circuits are the most popular and promising candidate for universal quantum computing and they have been realized via different types of NISQ devices [6]. Consequently, in this work, we will use gate-based quantum circuits as the primitive language for hybrid quantum neurons and HQDNNs. However, we would like to point out that there are also ongoing studies to design DNNs, being classical, quantum, or hybrid, via specialized computing devices from, e.g. optical and atomic systems [49–53].

Originated from perceptron, a neuron in modern classical DNNs can be described as a map that takes an input vector $\mathbf{x} \in \mathbb{R}^n$ and maps it into a single real number $y \in \mathbb{R}$

$$y = g_a\left(\mathbf{w}^T\mathbf{x} + b\right), \tag{1}$$

where $\mathbf{w}$ is the weight, b is the bias, and g_a represents a nonlinear activation function (see also figure 1).

While it is tempting to keep an exact form for QDNNs, it is not very efficient to translate operations like simple addition/multiplication and nonlinear functions like ReLu and Softmax into a gate-based quantum circuit [41]. Considering the hybrid nature of the HQDNN studied in this work, we focus on a class of quantum neurons that can be described by the following expression

$$y = \langle \phi(x) | U_N^\dagger A U_N | \phi(x) \rangle, \tag{2}$$

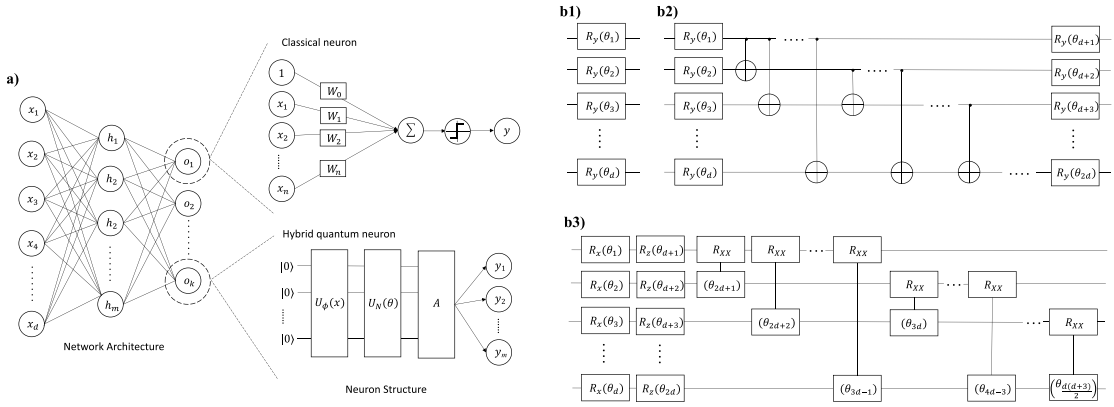

Figure 1. Illustration of hybrid quantum neuron network and examples of neuron circuits. (a) Hybrid and conventional neural networks share a similar network architecture, which is presented as a multi-layer perceptron in this work. A classical neuron takes a vector of real numbers as input and passes their linear combination into a nonlinear activation function to get the final results. A hybrid quantum neuron applies a series of gate operations to initial states and extracts the results via measurement. (b) Some examples of neuron circuits include (b1) single-layer R_y, (b2) Real Amplitude, and (b3) Quantum Circuit Born Machine.

which is also depicted in figure 1. Here $|\phi(x)\rangle$ is the input quantum state with encoded features, U_N represents the operation of the neuron circuits and A is what we called an evaluation operator. The unitary operator U_N dictates how we transform the input qubits and A is used to map the output qubits into a single real number. Note that, although U_N, A and $|\phi(x)\rangle$ always have matrix representations in non-interacting closed quantum systems, the quantum neuron described above is not linear as the representations usually involve trigonometry functions and the measurement processes inevitably leads to quadratic forms.

2.1. Structure of hybrid neuron

To start with, we describe the three components inside a hybrid quantum neuron. Firstly, the neuron takes classical input or a vector of real numbers x. To encode this classical information into qubits, we apply a feature map $U_\phi(x)$ to transform an initial state of the circuit (usually $|0\rangle$) into the input state $|\phi(x)\rangle = U_\phi(x)|0\rangle$. It's common to encode x as the amplitude of quantum states [54], which can be done via a simple circuit

$$U_\phi(x) = \otimes_i R_d(x), d = x, y. \tag{3}$$

We will refer to this as SLRd for later convenience. The choice of d is arbitrary, but we use R_y in this work since it only involves real amplitudes and such a circuit has been plotted in figure 1(b1). The reason R_z is not applicable here is that it only introduces phase differences and does not alter the density distribution.

Once having the input state $|\phi(x)\rangle$, we apply the neuron circuit $U_N(\theta)$, which is parameterized by the learnable parameters θ, to get the output state. This is usually the most important and complicated component and previous works on either QDNNs or HQDNNs tend to apply well-known variational ansatz like *RealAmplitudes* (RA) [46, 55] and quantum circuit Born machines (QCBMs) [37, 56], which are demonstrated in figures 1(b1) and (b2) respectively. We will carefully examine these choices in the next subsection.

In the last step, we perform a measurement on the output state, which gives its expectation value on a given operator A. However, we notice that given a different evaluation operator A', it is possible to define a unitary matrix so that $A' = U_A^\dagger A U_A$ and the measurement result on A' becomes

$$y(A') = \langle \phi(x)| U_N(\theta)^\dagger U_A^\dagger A U_A U_N(\theta) |\phi(x)\rangle. \tag{4}$$

Note that the above equation can also be interpreted as a measurement result of A on the same neuron but with different parameters $U_N(\theta') \approx U_A U_N(\theta)$. That is, in the case where the neuron can capture arbitrary unitary transformations, it makes no difference to choose any evaluation operator since the neuron can always properly adjust its parameters during backpropagation. In practice, this constraint is less strict since we do not need to maintain the exact unitary transformation but only the final measurement results. This suggests that even for simple neurons with limited expressibility, the effects of measurement operators might not be significant.

We also remark that we should avoid $[U_\phi(x)U_N(\theta), A] = 0$ as this would always lead to a trivial measurement result and the network cannot learn from data. Since we already use operators like R_x and R_y to

Table 1. Hybrid expressibility characterized by KL-divergence for different hybrid quantum neurons.

	$A = \sigma_x \otimes \sigma_x$	$\sigma_y \otimes \sigma_y$	$\sigma_z \otimes \sigma_z$
SLRy	0.060	0.693	0.054
RA	0.006	0.693	0.010
QCBM	0.394	0.018	0.018

encode classical features, it is convenient to choose $A = \otimes_i \sigma_z^i$ since $[\sigma_d, \sigma_e] = 2i\epsilon_{def}\sigma_f$, where ε_{def} is the Levi–Civita symbol and Einstein summation notation is implied. In this case, the corresponding measurement result is also known as polarization.

One drawback of the above single measurement schema is the cost of bountiful information inside a quantum neuron. There are different ways to extract more information from a single neuron to make HQDNNs more efficient. For example, we can use different measurement operators on individual qubits or we can conduct multiple measurements on a single neuron. While this topic deserves separate studies, we will demonstrate a case where one can make multiple inferences for classical predictions via only one single neuron.

2.2. Expressibility of hybrid neuron

Characterizing the expressibility of a quantum neuron is an interesting yet challenging problem [55]. In the context of fully quantum information processing, it is defined as the capability of a circuit to generate pure quantum states that are well representative of the Hilbert space [57]. Formally, it can be expressed as

$$\left\| \int_{\text{Haar}} (|\psi\rangle\langle\psi|)^{\otimes t} \, \mathrm{d}\psi - \int (|\psi_\theta\rangle\langle\psi_\theta|)^{\otimes t} \, \mathrm{d}\theta \right\|_{\text{HS}}, \tag{5}$$

where the Haar measure corresponds to maximally expressive uniform distribution of state, $|\psi_\theta\rangle = U_N(\theta)|0\rangle$ and HS represents Hilbert–Schmidt norm. However, this definition makes it hard to derive a single meaningful score and thus, the authors proposed to use the Kullback–Leibler (KL) divergence between the estimated fidelity distribution and that of the Haar-distributed ensemble to estimate the expressibility of the circuits [57].

The above characterization is invalid in the context of hybrid quantum neurons since the quantum state collapses to bounded real numbers after measurements. But following a similar argument, we can compute the KL divergence between the estimated measurement distribution and that of the Haar-distributed ensemble, which reduces to a uniform distribution on $[-1, 1]$. To distinguish this from quantum expressibility, we refer to this as hybrid expressibility in this work. Intuitively, higher quantum expressibility usually leads to higher hybrid expressibility. To demonstrate this, we take three different neurons SLRy, RA, and QCBM, and consider only two qubits for convenience. The three circuits are plotted in figure 1(b) and the corresponding results are summarized in table 1. We also include three different evaluation operators and 1 million samples are randomly drawn to compute the KL-divergence in each case. Regarding quantum expressibility, it is clear that RA is much better than both SLRy and QCBM, while SLRy has the lowest quantum expressibility. Such a trend remains for hybrid expressibility as suggested by the above table.

However, expressibility is not the only metric and we collect some other properties of the neuron circuit in table 2 (the linear RA is a reduced version RA with only linear numbers of CNOT gates and N is the number of qubits within a neuron). We can see that while SLRy has relatively lower expressibility, it only requires linear numbers of gates and has a constant circuit depth, which means it uses way fewer computing resources and might be more robust to noises. Now, the question comes if we can still build useful HQDNNs using SLRy, despite its low expressibility.

The answer is affirmative. To show this, we consider the simplest architecture where we apply SLRy to both feature map and neuron circuit and use polarization as measurement results. With such a setup, it is easy to show that the final measurement results of the neuron become $y = \prod_{j=1}^{N} \cos\chi_j, \chi_j = x_j + \theta_j$, which can be rewritten via Taylor expansion as

$$y \approx -\sum_j x_j\theta_j - \frac{1}{2}\sum_j \theta_j^2 + \sum_j \left(x_j \sum_{k\neq j} x_k^2 \right) \theta_j + \ldots, \tag{6}$$

where we drop all constant terms including those containing only x_j. The very first term reproduces the classical neuron with $g_a = \tanh$ when x_j and θ_j are small. We also notice that there are higher-order terms of x_j, which are explicit feature interactions that can help the model learn better. These observations suggest that a HQDNN with even the simplest neurons like SLRy can also tackle challenging problems when it is

Table 2. Comparison of different quantum neurons. N is the number of qubits inside a single neuron.

	# of gates	# of parameters	Circuit depth	Quantum expressibility	Hybrid expressibility
SLRz	N	N	1	None	None
SLRx/SLRy	N	N	1	Low	Low
Linear RA	$3N$	$2N$	$O(N)$	Medium	High
Full RA	$\frac{1}{2}\left(N^2+3N\right)$	$2N$	$O(N^2)$	High	High
QCBM	$\frac{1}{2}\left(N^2+3N\right)$	$\frac{1}{2}\left(N^2+3N\right)$	$O(N^2)$	High	Medium

properly scaled up. So an HQDNN can be regarded as a nonlinear generalization of classical NNs [58–60] and such a generalization can be highly nontrivial as the number of meaningful polynomials in Taylor expansion can grow exponentially, which brings quantum-specific behaviors to the network.

While SLRy is already a strong neuron when compared to its classical counterpart, entanglement via controlled gates can further improve their capability as we already see in previous discussions that RA and QCBM have higher quantum and hybrid expressibility. For demonstration purposes, we will use a minimal nontrivial example with only two qubits. For SLRy with two qubits, we have $p_1=\cos^2\frac{\chi_1}{2}$, $p_2=\cos^2\frac{\chi_2}{2}$ and $p_3=\frac{1}{4}\left(2+\cos(\chi_1-\chi_2)\cos(\chi_1+\chi_2)\right)$, where $p_1=\rho_0+\rho_1, p_2=\rho_0+\rho_2, p_3=\rho_0+\rho_3$ and $\rho_j=|\langle j|U_N|\phi(x)\rangle|^2$. The physical meaning of the three quantities is clear $p_1=P(qb_1=0)$, $p_2=P(qb_0=0)$ and $p_3=P(qb_0=qb_1)$ and $qb_i, i=0,1$ represents the qubit inside the neuron circuit. In this case, features and learnable parameters are linearly combined within some trigonometry functions.

If we add a CNOT gate after SLRy, we will have

$$
\begin{aligned}
p_1 &= \frac{1}{2}\left(1+\cos x_1\cos\theta_1-\sin x_1\sin x_2\sin\theta_1\right),\\
p_2 &= \frac{1}{2}\left(1+\cos x_1\cos x_2\cos\theta_2-\sin x_2\sin\theta_2\right),\\
p_3 &= \frac{1}{2}\left(1+\sin x_1\sin\theta_1\sin\theta_2+\cos x_2\cos\theta_1\cos\theta_2-\cos x_1\cos\theta_1\sin x_2\sin\theta_2\right).
\end{aligned}
$$

The power of entanglement comes from crossing terms like $\cos x_1\cos x_2$, $\sin x_1\sin x_2$, and $\cos x_1\sin x_2$, which leads to an infinite order of interactions between different features under Taylor expansion. Those higher-order interactions among features are very powerful in large-scale classical machine learning models [61, 62] while they are naturally introduced in hybrid neurons. Note that even though we use a simple two-qubit case for the sake of analytical results, the conclusions here can be generalized to arbitrary numbers of qubits within a single neuron.

Although we have shown that entanglement could boost the expressibility and learning capability of hybrid neurons, SLRy already provides a competitive candidate. In the following, we will conduct simulations of HQDNNs and show how simple neurons could bring superior performance, which paves the way for studying large-scale hybrid NNs in the NISQ era.

3. Hybrid quantum neural networks

With only the building blocks or hybrid neurons, we are not yet ready to construct a useful HQDNN and still need to address several important questions including network architecture and forward/backward prorogation. For demonstration, we assume a classification task with multiple classes and discuss these topics based on the hybrid neurons studied in the previous section.

We will mostly consider a hybrid MLP and its structure is largely the same as its classical counterpart, except we are using a hybrid quantum neuron. The final measurement result of a single neuron is confined to $y\in[-1,1]$ when unitary evaluation operators are assumed. This can be easily mapped to a rotation angle and serves as the input to neurons in the subsequent layers.

The hybrid MLP also has three parts, namely, the input layer, the hidden layer, and the output layer. We will first discuss how to make inferences/predictions with hybrid neurons and then give more details about both forward and backward propagations.

3.1. Prediction & cost function

Assuming we are tackling a classification problem with m classes and we keep the cost function to be cross-entropy, which is widely used in classical neuron networks

$$\mathcal{L} = -\frac{1}{N_d}\sum_{\alpha=1}^{N_d}\sum_{\beta=0}^{m-1} y_\alpha^\beta \log P(\beta|x_\alpha)\,, \tag{7}$$

where N_d is the number of data points in the training set, x_α (y_α^β) is the feature set (ground-truth label) of the αth data point. To infer a reasonable result for m-class classification, we require at least $m-1$ output for the cost function to process.

A straightforward way is to follow the scheme in classical neuron networks and use multiple neurons in the output layer, which is referred to as multi-neuron single-measurement or MNSM in this work. However, this method does not utilize the power of quantum computing. Here, we propose a different approach to make multi-class prediction via single-neuron multi-measurement (SNMM). For such a schema, we only have a single neuron in the output layer, but we can perform a set of measurements on the single neuron. For the case of qubits, we can use three Pauli matrices to make three measurement results, which works for a classification problem with up to three classes. In general, we can also extend this to qudit or use different measurements on different qubits to address problems with arbitrary classes.

We use the softmax function, similar to Boltzmann distribution, to map the measurement results into probability before we feed it into the cost function

$$P(\beta|x_\alpha) = e^{-y(\sigma_\beta)/\mathcal{T}}/q, \tag{8}$$

where $q = \sum_\beta e^{-y(\sigma_\alpha)/\mathcal{T}}$ and $\mathcal{T}$ is, in general, a tuneable (or learnable) temperature hyperparameter, which is necessary here as the measurement results are strictly bounded.

Note that since computing the cost function requires the measurement results from the output layer, this is typically done using the classical computing resource in the hybrid architecture.

3.2. Forward propogation

We only consider MLP with one hidden layer and we can denote it as a function $f_h : [0,\pi]^{n_f} \to [-1,1]^{n_h}$ and the output of the hidden layer reads $h' = f_h(x,\theta)$, where n_f is the number of neurons in previous layer (or input layer in this case) and θ contains all the learnable parameters of the n_h neurons. Then h' is linearly transformed into the interval $[0,\pi]$ via $h = \frac{\pi}{2}(h'+1)$, which becomes the features being fed into next layer. The same step can be repeated so that one can build more complicated HQDNNs with multiple hidden layers.

So the forward propagation starts from the input layer, which consumes the normalized and transformed features. Then the results are passed to the hidden layer. Once hitting the output layer, we process the outputs using the softmax function and compute the cost as discussed in the previous subsection.

3.3. Backpropagation & optimizer

For HQDNNs, we use classical algorithms to optimize the cost function during the backpropagation phase. Similar to classical DNNs, this can be either gradient-based or gradient-free algorithms.

Regarding gradient-free methods, we consider the so-called particle swarm optimization (PSO) [63]. The algorithm iterates for a given number of steps or stops automatically when reaching a local minimum and it is useful in classical optimization problems including optical inverse design of structural color [64]. In our implementation, we use the Pyswarms package [65] and set the number of particles to twice the number of qubits in each neuron. We also fix the hyperparameters $\{c1, c2, w\}$ controlling the swarm dynamics to be $\{0.5, 0.5, 0.5\}$ and the number of iterations to be 50 in each epoch.

Also, we test the gradient-based optimization algorithms offered by Qiskit, such as the vanilla gradient descent (GD) and ADAM [66, 67]. The parameters of both algorithms used here are default except the iteration is 10 and the learning rate is 0.1 in each epoch. The vanilla GD is an iterative scheme to find the minimum of a function f with an initial point $\vec{\theta}_0$, by updating the parameters in the direction of the negative gradient of f for a small learning rate $\eta_n > 0$:

$$\vec{\theta}_{n+1} = \vec{\theta}_n - \eta_n \vec{\nabla} f\left(\vec{\theta}_n\right). \tag{9}$$

The ADAM [66] algorithm only requires first-order gradients with little memory requirement and is invariant to the diagonal rescaling of the gradients. Moreover, it can cope with non-stationary objective

functions and noisy and/or sparse gradients. AMSGRAD [67] (a variant of ADAM) uses a 'long-term memory' of past gradients and, thereby, improves convergence properties.

To confine the learnable parameters to a proper range, we set a bound for each learnable parameter in PSO. For gradient-based algorithms like GD and ADAM, we add a modular operation after each training epoch.

We also remark that the current schema for backward propagation is hard to implement on quantum computing devices so this part must be done on classical computers. In a hybrid computing architecture, after the optimization is done, the update parameters will be passed into the quantum devices to perform the next round of forward propagation.

4. Numerical simulations

As we focus on classification tasks in this work, we use the popular iris and wheat seed data sets from the UCI machine learning repository [68] to test out hybrid neural networks. Currently, we only simulate our models on a gated-based universal quantum computer using Qiskit and left for future work training and testing on state-of-the-art quantum devices. Because simulations of quantum computers on classical computing platforms require exponential resources, we consider an HQDNN containing only one hidden layer with two hybrid neurons.

For each dataset, five trials of quantum simulation are performed and each simulation contains 15 epochs. Then we take the average value of five simulation as the result, and use standard error(SE) to calculate errorbar, which is $\mathrm{SE} = \sigma/\sqrt{n}$. We use SLRy feature mapping and networks are initialized using a normal distribution with unit deviation and scaled to $[-\pi, \pi]$. We explore a set of neurons including SLRy, RA, and QCBM with varying prediction layers and optimizers. Overall, we observe superior performance compared to previous work and the results are consistent with our discussions so far.

4.1. Iris dataset

The iris dataset consists of 150 samples, which can be classified into three classes. For each iris sample, it has four real-valued features, namely, sepal length, sepal width, pedal length, and petal width. The three classes of setosa, versicolor, and virginica are labeled by integers from 0–2. In each experiment, eighty percent of the data points are randomly selected for training and the remaining are for testing.

4.1.1. Tests of optimizers

We first test the performance of HQDNNs using different optimization algorithms, including PSO, GD, and ADAM, as discussed in the previous section. For other network setups, we use SLRy neurons with polarization measurement, MNSM prediction, and $\mathcal{T} = 10$. In figure 2, we plot the training process with cost and accuracy evaluated on training and test datasets respectively. The results suggest that the cost function can be optimized with all algorithms and achieve accuracy better than 80%. The two gradient methods achieve similar cost and accuracy at the end of the training process while ADAM is slighter better than GD with faster convergence. The PSO algorithm converges faster than GD at the beginning but it settles down on a worse local minimum. It's convenient to have classical optimizers work on HQDNNs as we do not need to develop new quantum algorithms for this so that they can be readily implemented with NISQ devices. In the following, we will see that those algorithms also work when we have more complicated and entangled neurons.

Accuracy alone is not enough to fully characterize the performance of a classifier, thus, we also compute other critical metrics including precision, recall, F1-score, and AUC (here use micro averaging). Specifically, we use the classification_report() API from scikit-learn to compute all relevant metrics and the results are organized in table 3. The actual occurrence of each class, or support, is 10, 8, and 12 for classes 0, 1, and 2 respectively.

Overall, we have found consistent results from the various metrics presented above, compared to figure 2. ADAM is better in terms of all three metrics while PSO has slightly lower readings than the others. The reported performance here is comparable to SOTA quantum machine learning models studied in the literature (see appendix A for more details).

In conclusion, the proposed HQDNN shows promising results with different optimizers and ADAM is in general preferred as it converges faster while achieving better results on average. This observation is consistent with classical neural networks and provides the possibility to scale up HQDNNs easily under accessible hardware platforms.

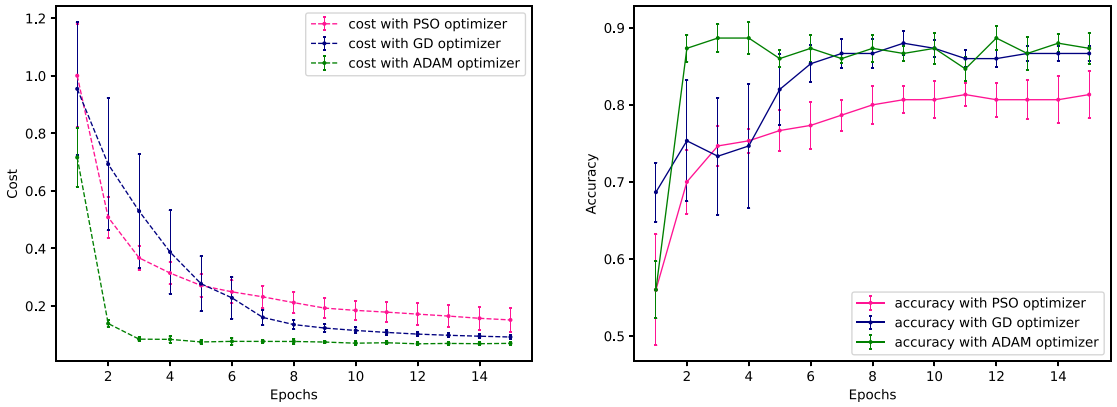

Figure 2. Respective average cost (left) and accuracy (right) of training and test data achieved by PSO, GD, and ADAM optimizers. From the results, we see that ADAM achieves the fastest convergence and generally better performance.

Table 3. Average performance on each class with PSO, GD, and ADAM optimizer.

	Precision	Recall	F1-score	AUC	Optimizer
Class 0	0.94	0.88	0.91		
Class 1	0.75	0.90	0.81	0.95	PSO
Class 2	0.88	0.80	0.84		
Class 0	1.00	0.92	0.96		
Class 1	0.72	0.98	0.83	0.97	GD
Class 2	0.96	0.80	0.87		
Class 0	1.00	0.98	0.99		
Class 1	0.78	0.98	0.87	0.98	ADAM
Class 2	0.98	0.83	0.90		

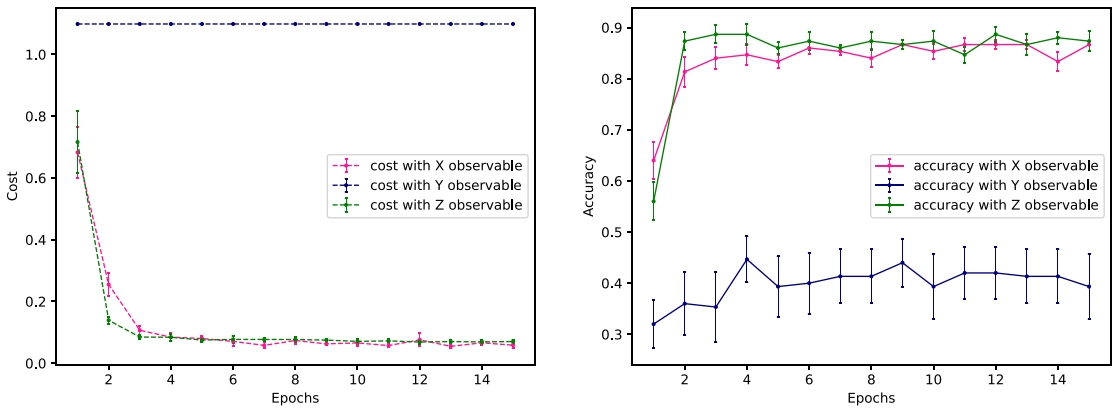

Figure 3. Average cost of training data and accuracy of test data achieved by X, Y, and Z observable. The X or Z observable can achieve outstanding performance while the Y observable cannot due to vanishing gradients.

4.1.2. Effects of evaluation operator

In the previous section, we have shown that the evaluation operator makes no difference to the HQDNNs when the neuron can learn arbitrary unitary transformations and argued that this might still be true even if the neuron has slightly weak expressibility. While it is challenging to prove it analytically, we use numerical results to support this.

We keep the same setting as in previous experiments and fix the ADAM optimizer. We then test the HQDNNs with different evaluation operators from the generators of SU(2) group and the results are summarized in figure 3 and table 4.

It's clear that the cost function remains a constant and all the metrics fluctuate around 0.5, which corresponds to the baseline of a random classifier when $A = \otimes_i \sigma_y^i$. This suggests that the network barely learns from the data when $A = \otimes_i \sigma_y^i$ and the neuron has a SLRy structure due to vanishing gradients. On the other hand, when $A = \otimes_i \sigma_x^i$ or $A = \otimes_i \sigma_z^i$, we have seen comparable results as they converge in the same peace

Table 4. Average performance of HQDNNs with different evaluation operators σ_d.

	Precision	Recall	F1-score	AUC	A
Class 0	1.00	0.94	0.97		
Class 1	0.79	0.95	0.86	0.96	$\otimes_i \sigma_x^i$
Class 2	0.91	0.83	0.87		
Class 0	0.47	0.60	0.52		
Class 1	0.43	0.48	0.44	0.62	$\otimes_i \sigma_y^i$
Class 2	0.57	0.37	0.45		
Class 0	1.00	0.98	0.99		
Class 1	0.78	0.98	0.87	0.98	$\otimes_i \sigma_z^i$
Class 2	0.98	0.83	0.90		

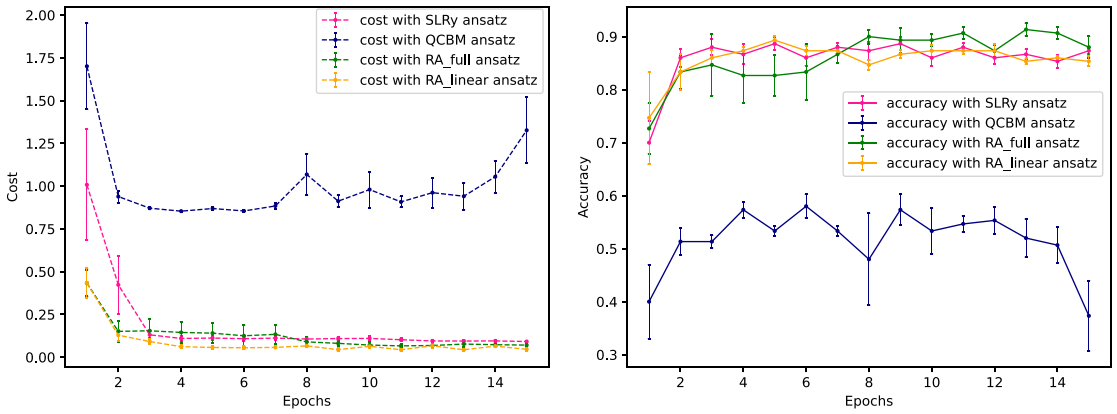

Figure 4. Average cost of training data and accuracy of test data achieved by SLRy, RA, and QCBM. We find that SLRy and RA achieve the same level of performance while full RA is slightly better since it has more learnable parameters and higher expressibility. However, QCBM shows lower performance despite its high expressibility, which is caused by the mismatch between the feature map and the neuron circuit.

and arrive at final accuracy that is close to 0.9. Their performance is also similar in terms of precision, recall, and F1 score. Those results suggest that even the most simple hybrid neuron can adjust itself to different evaluation operators to make useful inferences from the data. Overall, the numerical simulations are highly consistent with our previous claims.

4.1.3. Experiments on neuron circuits

So far, we have tested the HQDNNs on different optimizers and evaluation operators and observed superior performance on classical classification tasks. However, we have focused on the special case of SLRy, which contains only product states and does not fully utilize quantum features like entanglement. In the following, we will study HQDNNs with 'real' quantum circuits that cannot be efficiently simulated on any classical computing devices. More specifically, we tested RA (with both linear and full CNOT gates) and QCBM and the results are presented in figure 4 and table 5. The setup of the network is the same as previous experiments and a SNMM prediction layer is applied.

From the results, we can see that RA is slightly better than SLRy. Its training cost drops faster at the beginning and then hits a plateau higher than SLRy, but eventually converges at better local minimums. All metrics of RA in each class are also better than SLRy. This is reasonable since RA has more parameters and higher expressibility. However, for this simple task, we only see marginal gains of a much more complicated and entangled quantum neuron. There are several assumptions here. First of all, our setup does not fully utilize the quantum features, and the dense information carried by a single strong neuron gets lost during measurement. On top of that, the task might be too easy to make a significant distinction between different neurons. This also leads to a trade-off between the capability of a single neuron and the complexity of the overall network structure.

On the other hand, QCBM seems to have a relatively lower performance. Although it is better than a random classifier, it has a hard time converging and the performance metrics show large gaps compared to even simple SLRy neurons. This is a result of a mismatch between the feature map and the neuron circuit as the density information encoded with R_y is hard to extract by subsequent operations like R_x and/or R_{xx}.

Table 5. Average performance on each class with SLRy, RA, and QCBM.

	Precision	Recall	F1-score	AUC	Ansatz
Class 0	1.00	0.92	0.96		
Class 1	0.77	0.98	0.86	0.97	SLRy
Class 2	0.95	0.83	0.89		
Class 0	0.79	0.64	0.70		
Class 1	0.66	0.85	0.67	0.78	QCBM
Class 2	0.71	0.57	0.60		
Class 0	1.00	1.00	1.00		
Class 1	0.84	0.98	0.90	0.99	RA_full
Class 2	0.98	0.87	0.92		
Class 0	1.00	1.00	1.00		
Class 1	0.79	0.95	0.86	0.99	RA_linear
Class 2	0.96	0.83	0.89		

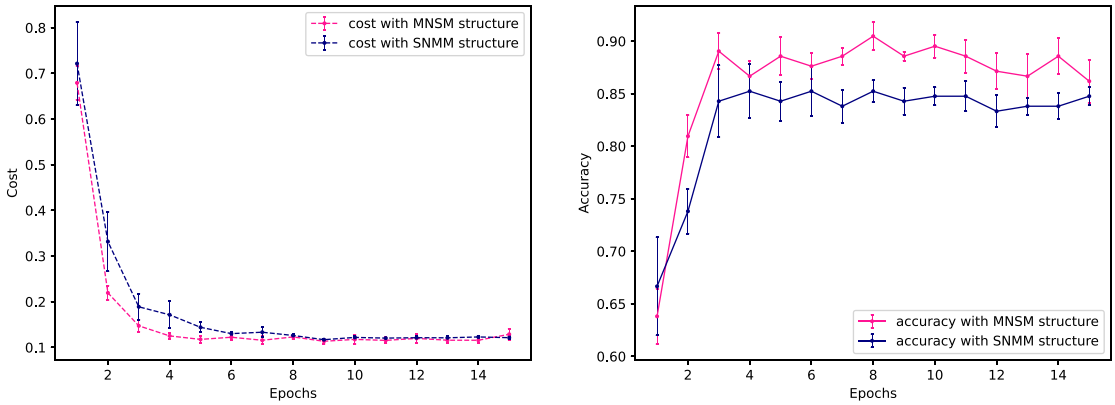

Figure 5. Average cost of training data and accuracy of test data achieved by MNSM and SNMM structure. The better performance of MNSM can be ascribed to more learnable parameters.

Table 6. Average performance on each class with MNSM and SNMM structure.

	Precision	Recall	F1-score	AUC	structure
Class 0	0.93	0.90	0.90		
Class 1	0.89	0.97	0.92	0.98	MNSM
Class 2	1.00	0.95	0.97		
Class 0	0.94	0.81	0.87		
Class 1	0.79	0.95	0.86	0.98	SNMM
Class 2	0.99	0.96	0.97		

4.2. Wheat seed dataset

As we observe the superior performance of different HQDNNs on the iris dataset, we would like to conduct some experiments on a more complex problem. Here, we turn to look at the wheat seed dataset, which has been less tested on quantum machine learning models so far.

The wheat seed dataset has three classes, Kama, Rosa, and Canadian (labeled as class 0,1,2), with each class containing 70 samples. It also has 7 real-valued features, making it much more complex than the iris dataset. Here we use this dataset to show that our model is capable of handling more complex scenarios and study how noisy gates affect the performance of the HQDNNs. Similarly, eighty percent of all data is chosen randomly for training and the remaining for testing.

4.2.1. Effects of prediction schemes

We keep the network architecture as before and use SLRy as the neuron circuit. We then tested two different prediction schemes MNSM and SNMM with ADAM optimizer. The results are summarized in figure 5 and table 6. The actual occurrence of each class is 15, 12, and 15 for each class respectively.

Both schemes achieve fairly good performance on the dataset, indicating the potential of the HQDNNs to handle complex real-world problems. Overall, we observe a comparable performance of the two schemas given a similar AUC of 0.98. However, MNSM shows marginally better performance than SNMM if we look

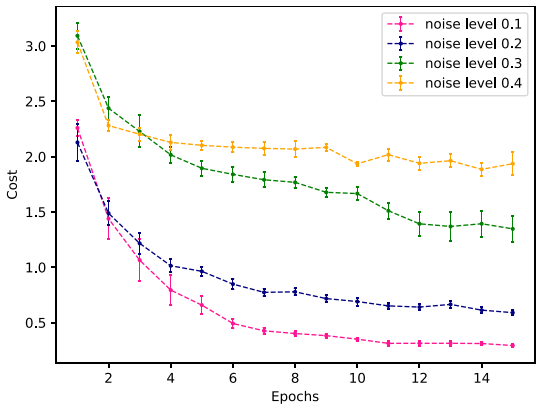

Figure 6. Average cost of training data under different noise levels. The HQDNN still learns under relatively small noises and there is a large performance gap when the noise level crosses 0.2.

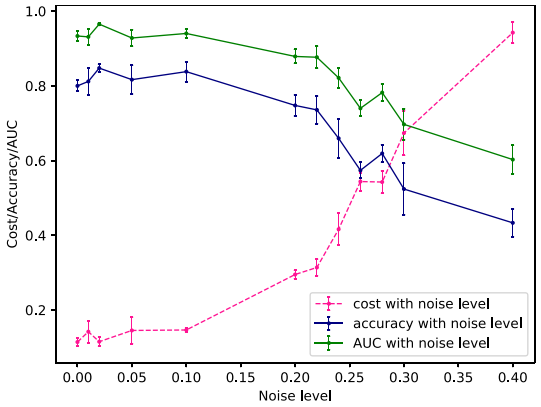

Figure 7. Average cost, accuracy, and AUC under different noise levels. The performance degrades linearly when the noise level is between 0.1 and 0.4. Even when the noise level comes to 0.4, the accuracy is still better than random guesses with a baseline accuracy of 0.33.

at micro-averaged metrics including precision, recall, and F1-score for both class 0 and class 1. Moreover, MNSM converges slightly faster than SNMM. There could be multiple reasons for this. First, the MNSM schema uses more neurons in the output layer and thus has more learnable parameters than SNMM, leading to higher overall model capability. On top of that, we use a simple SLRy neuron, and the prediction capability is limited. Introducing complicated neurons with entanglement could be very helpful to the SNMM schema.

4.2.2. Effects of gate noises

An important characteristic of NISQ devices is they exhibit some kind of noise in their gate operations and measurements. Here, we also investigate how the proposed HQDNN behaves under noises and show that it could perform reasonably using SOTA NISQ devices.

For simplicity, we assume there are only noises in the gate operations within the neuron circuit, which follow a standard Gaussian distribution, and its amplitude is controlled by an extra parameter, the noise level. We then simulate the HQDNN and plot its learning curves against different noise levels in figure 6. The structure of the network is similar to the previous subsection and we use SNMM prediction with PSO optimizer. Our numerical results suggest that gradient-based optimizers are more likely to be affected by noises since they are sensitive to small local changes. All data points are computed from 10 independent runs to reduce the effects of outliers.

Given relatively small noise levels like 0.1 and 0.2, the model can still learn from the data and the cost can be optimized to a certain value, which is usually much larger than the ideal case. When the noise level comes to 0.3 and 0.4, the network cannot effectively minimize the cost and has a harder time converging from runs to runs as we observe large errors in the final cost after even 15 epochs in different runs. Although a continued downward trend of the cost is seen when the noise level is 0.3, the cost settles quickly after just a few epochs if we increase it to 0.4, indicating that the learning process has been hindered by the strong noises.

To further quantify the effects of the noises, we plot different metrics with respect to varying noise levels in figure 7. It's obvious that the performance of the model shows an approximately linear degradation when the noise level goes beyond 0.1. This is related to the simple architecture of our neuron circuits, otherwise, we would observe much fast decay of AUC and accuracy (see appendix B for more details). Also, when the noise level hits 0.4, the accuracy drops close to 0.4, but it is still much better than a random guess, which corresponds to an accuracy of 0.33. Thus, even under such a strong noise, the model is still capable of extracting some meaningful information from the data.

Another interesting observation from the result is that the performance of the network actually goes up when the noise level is around 0.02. This is similar to what has been studied in a class of quantum-inspired algorithms including the Hopfield recurrent neural network for quadratic unconstrained binary optimization, where a certain level of noise could boost the performance [69]. Current SOTA NISQ devices like IBM Prague can achieve very high fidelity and low noise for most common gates [70]. Taking quantum computer 'ibm_wasington' with the newest Eagle quantum processor for an example, the median error for σ_x is $3.01 \times 10^{(-4)}$ while CNOT is $1.75 \times 10^{(-2)}$ [71]. This actually suggests that the QDNNs might show better performance on current NISQ devices than noise-free clean simulations. It would also be interesting to study how the effects of noise scale with the network size, but we will leave it to future studies.

5. Conclusion and outlook

In this work, we investigate a class of HQDNNs with a focus on gate-based hybrid neurons. We propose to characterize the neuron via hybrid expressibility, as a generalization to quantum expressibility and show that even the simplest hybrid neuron can bring advantages to classical neurons as they contain higher-order feature interactions, which can be further enhanced by entanglement. We also discuss related components of HQDNNs including network structure, inference, cost functions, and optimizers. Most importantly, we show it is possible to make multiple predictions through a single hybrid neuron. We then develop a highly customizable platform to simulate the proposed HQDNNs via Qiskit and show that they can achieve superior performance on different classification problems. Last but not least, we show that HQDNNs with simple neuron circuits are more robust to noises, making it possible to build useful large-scale neural networks with quantum features in the NISQ era.

There are still many fundamental questions to be answered about HQDNNs. For example, we use SLRy as the feature map, which significantly limits the capability of the hybrid neuron. It would be interesting to search for a better feature map to encode denser classical information into a hybrid neuron so different types of feature interactions are more involved. Moreover, as we reveal the important role of entanglement in the feature interaction, is it possible to make qubits entangled across neurons or even layers to enable more effective and efficient learning?

From a pure machine learning perspective, we have not utilized any modern deep learning techniques like dropout [72], layer/batch normalization [73, 74], residual connections [75], etc. It would also be fascinating to understand how those tricks could help HQDNNs perform better on different problems including classifications, regressions, and even generative tasks.

Finally, when it comes to experimental realizations. We can try to accelerate the training/prediction process via fewer measurements, which is not covered in this work. We may also build hardware-specific neurons, instead of gate-based ones, to better fit the hybrid schema.

In conclusion, our work provides a comprehensive study of a class of gate-based quantum hybrid neurons and HQDNNs built upon it. Through both analyses and numerical simulations, we show that HQDNNs are able to resolve challenging real-world problems and can be scaled up in the NISQ era. Our work provides necessary guidance for future studies of hybrid machine-learning algorithms and may help reveal more quantum advantages in the long run.

Data availability statement

All data that support the findings of this study are included within the article (and any supplementary files).

Acknowledgments

J Hou acknowledges X Wang for inspiring discussions about classical deep learning theory. This work is supported by Anhui Province University Natural Science Research Project (2023AH051102). F Miao is supported by Innovation Program for Quantum Science and Technology (2021ZD0302902).

Appendix A. Comparing with previous works

This appendix seeks to evaluate the performance of our HQDNNs in comparison to other QNNs using the iris dataset. The dataset consists of three classes: the setosa class, denoted as class 0 in our paper, is linearly separable from the other two classes. The remaining two classes (versicolor, labeled as class 1, and virginica, labeled as class 2) are not linearly separable from each other.

In previous work, a novel machine learning technique called the deep quantum-inspired neural network (DQINN) was introduced by integrating classical deep belief networks (DBNs) and quantum-inspired neural networks (QINNs) [76]. The DQINN is designed by combining the linear superposition of multiple DBNs with quantum intervals in the last hidden layer. In the study, the performance of the DQINN was evaluated on the iris dataset for multi-classification tasks, and a mean accuracy of approximately 97.3% was achieved.

In a recent study, Li and Wang [77] present a novel quantum neuron network composed of neurons that utilize swap test technology and phase estimation method to complete the mapping of input qubits to output qubits. The proposed QNN was then simulated on the iris dataset for multi-classification tasks, yielding recognition rates of 93.33% and 96.67% for the training and testing sets, respectively.

In paper [78], the authors introduce a variational quantum tensor network (QTN) that utilizes truncated outputs which are subsequently fed into a classical neural network. The design of shallow quantum circuits is enabled by leveraging kernel encoding, circuit models, multiple readouts, and stochastic GD algorithms. The accuracy of the proposed QTN is evaluated on the iris dataset for binary classification tasks with a 100% accuracy achieved for class 1 or 2, class 1 or 3, and class 2 or 3.

In another study, Dong *et al* [79] propose a novel variational QNN model that incorporates a quantum activation circuit and employs quantum particle swarm optimization to optimize the cost function. The proposed model is then applied to any two categories of the iris dataset for binary classification tasks, culminating in an accuracy of approximately 93%.

In a recent development, Yu *et al* [80] introduce a novel QNN architecture based on parallel-entanglement models, involving the establishment of connections between qubits through quantum entanglement. The proposed approach is shown to be highly effective in solving practical problems, exemplified by achieving an average accuracy exceeding 99% in classification tasks for the iris dataset.

Appendix B. Noisy RA ansatz on wheat seed dataset

In this appendix, we investigate the performance of hybrid neurons with full RA (figure B1 left) and linear RA (figure B1 right) when tested on the wheat seed dataset under varying levels of noise.

The prediction schema utilized here is SNMM and the optimizer is PSO, similar to the setup in the main text. We assume there's no noise in the CNOT gate. Five independent runs were conducted to mitigate the impact of outliers on the results. Our findings indicate that even when the noise level reaches 0.05 or 0.1, a satisfactory optimal cost can still be attained by linear RA. While for full RA, there's a jump in the final cost when the noise level increases from 0.05 to 0.1.

In a noise-free setup, it is clear that both full RA and linear RA can achieve a lower final cost than SLRy under the assumption of equivalent network architecture. However, when the noise level comes to 0.1, SLRy

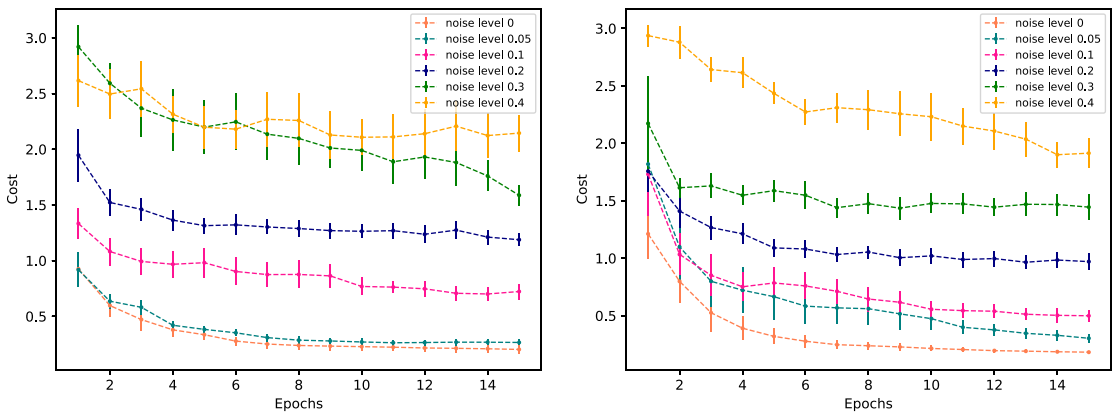

Figure B1. Average cost of Full RA ansatz (left) and Linear RA (right) with noise level. The network with Full and Linear RA ansatz can also achieve certain cost with 0.05 or 0.1 noise level while SLRy ansatz can achieve similar cost with a higher noise level of 0.1 or 0.2 and this lead to the observation that the simpler ansatz is more robust to the noise.

already shows much better results than RA. This indicates that in the NISQ era, it would be beneficial to utilize simple neurons so that the model is more robust to noises and we can scale it up to resolve meaningful real-world problems.

ORCID iDs

Changbin Lu https://orcid.org/0009-0001-0907-6461
Fuyou Miao https://orcid.org/0000-0001-8682-6003

References

[1] Arute F *et al* 2019 Quantum supremacy using a programmable superconducting processor *Nature* **574** 505–10
[2] Bennett C H and Brassard G 2020 Quantum cryptography: Public key distribution and coin tossing (arXiv:2003.06557)
[3] Shor P W 1994 Algorithms for quantum computation: discrete logarithms and factoring *Proc. 35th Annual Symp. on Foundations of Computer Science* (IEEE) pp 124–34
[4] McArdle S, Endo S, Aspuru-Guzik A, Benjamin S C and Yuan X 2020 Quantum computational chemistry *Rev. Mod. Phys.* **92** 015003
[5] Farhi E, Goldstone J, and Gutmann S 2014 A quantum approximate optimization algorithm (arXiv:1411.4028)
[6] Preskill J 2018 Quantum computing in the nisq era and beyond *Quantum* **2** 79
[7] Tannu S S and Qureshi M K 2019 Not all qubits are created equal: A case for variability-aware policies for nisq-era quantum computers *Proc. 24th Int. Conf. on Architectural Support for Programming Languages and Operating Systems* pp 987–99
[8] Kjaergaard M, Schwartz M E, Braumüller J, Krantz P, Wang J I-J, Gustavsson S and Oliver W D 2020 Superconducting qubits: Current state of play *Annu. Rev. Condens. Matter. Phys.* **11** 369–95
[9] Siddiqi I 2021 Engineering high-coherence superconducting qubits *Nat. Rev. Mater.* **6** 875–91
[10] Sadana S, Maccone L and Sinha U 2022 Testing quantum foundations with quantum computers *Phys. Rev. Res.* **4** L022001
[11] Niu S and Todri-Sanial A 2023 Enabling multi-programming mechanism for quantum computing in the Nisq era *Quantum* **7** 925
[12] Suppressing quantum errors by scaling a surface code logical qubit 2023 Suppressing quantum errors by scaling a surface code logical qubit *Nature* **614** 676–81
[13] Domingos P 2015 *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake our World* (Basic Books)
[14] Krizhevsky A, Sutskever I and Hinton G E 2017 Imagenet classification with deep convolutional neural networks *Commun. ACM* **60** 84–90
[15] Chollet F 2017 Xception: deep learning with depthwise separable convolutions *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* pp 1251–8
[16] Zewen Li, Liu F, Yang W, Peng S and Zhou J 2021 A survey of convolutional neural networks: analysis, applications and prospects *IEEE Transactions on Neural Networks and Learning Systems* (https://doi.org/10.1109/TNNLS.2021.3084827)
[17] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A N, Kaiser Łukasz and Polosukhin I 2017 Attention is all you need Advances in *Neural Information Processing Systems* 30 (available at: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf)
[18] Khan S, Naseer M, Hayat M, Waqas Zamir S, Shahbaz Khan F and Shah M 2022 Transformers in vision: a survey *ACM Computing Surveys (CSUR)* **54** 1–41
[19] van Dis E A M, Bollen J, Zuidema W, van Rooij R and Bockting C L 2023 Chatgpt: five priorities for research *Nature* **614** 224–6
[20] Sohl-Dickstein J, Weiss E, Maheswaranathan N and Ganguli S 2015 Deep unsupervised learning using nonequilibrium thermodynamics *Int. Conf. on Machine Learning* (PMLR) pp 2256–65 (available at: https://proceedings.mlr.press/v37/sohl-dickstein15.pdf)
[21] Jonathan H, Jain A and Abbeel P 2020 Denoising diffusion probabilistic models *Advances in Neural Information Processing Systems* vol33 pp 6840–51 (available at: https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf)
[22] Ramesh A, Dhariwal P, Nichol A, Chu C and Chen M 2022 Hierarchical text-conditional image generation with clip latents (arXiv:2204.06125)
[23] Dai D, Sun Y, Dong Li, Hao Y, Sui Z and Wei F 2022 Why can gpt learn in-context? language models secretly perform gradient descent as meta-optimizers (arXiv:2212.10559)
[24] Wies N, Levine Y and Shashua A 2023 The learnability of in-context learning (arXiv:2303.07895)
[25] Jordan M I and Mitchell T M 2015 Machine learning: trends, perspectives and prospects *Science* **349** 255–60
[26] Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N and Lloyd S 2017 Quantum machine learning *Nature* **549** 195–202
[27] Huang H-Y, Broughton M, Mohseni M, Babbush R, Boixo S, Neven H and McClean J R 2021 Power of data in quantum machine learning *Nat. Commun.* **12** 2631
[28] Cerezo M, Verdon G, Huang H-Y, Cincio L and Coles P J 2022 Challenges and opportunities in quantum machine learning *Nature Comput. Sci.* **2** 567–76
[29] Schuld M and Killoran N 2022 Is quantum advantage the right goal for quantum machine learning? *Prx Quantum* **3** 030101
[30] Martín-Guerrero J D and Lamata L 2022 Quantum machine learning: a tutorial *Neurocomputing* **470** 457–61
[31] Riste D, Da Silva M P, Ryan C A, Cross A W, Córcoles A D, Smolin J A, Gambetta J M, Chow J M and Johnson B R 2017 Demonstration of quantum advantage in machine learning *npj Quantum Inf.* **3** 16
[32] Sentís G, Monras A, Munoz-Tapia R, Calsamiglia J and Bagan E 2019 Unsupervised classification of quantum data *Phys. Rev.* X **9** 041029
[33] Peters E, Caldeira J ao, Ho A, Leichenauer S, Mohseni M, Neven H, Spentzouris P, Strain D and Perdue G N 2021 Machine learning of high dimensional data on a noisy quantum processor *npj Quantum Inf.* **7** 161
[34] Huang H-Y, Chen S and Preskill J 2022 Learning to predict arbitrary quantum processes (arXiv:2210.14894)
[35] Huang H-Y *et al* 2022 Quantum advantage in learning from experiments *Science* **376** 1182–6
[36] Jerbi S, Fiderer L J, Poulsen Nautrup H, Kübler J M, Briegel H J and Dunjko V 2023 Quantum machine learning beyond kernel methods *Nat. Commun.* **14** 517

[37] Benedetti M, Garcia-Pintos D, Perdomo O, Leyton-Ortega V, Nam Y and Perdomo-Ortiz A 2019 A generative modeling approach for benchmarking and training shallow quantum circuits *npj Quantum Inf.* **5** 45
[38] Beer K, Bondarenko D, Farrelly T, Osborne T J, Salzmann R, Scheiermann D and Wolf R 2020 Training deep quantum neural networks *Nat. Commun.* **11** 808
[39] Feulner V and Hartmann M J 2022 Variational quantum eigensolver ansatz for the j 1- j 2-model *Phys. Rev.* B **106** 144426
[40] Pan X *et al* 2023 Deep quantum neural networks on a superconducting processor *Nat. Commun.* **14** 4006
[41] Cao Y, Giacomo Guerreschi G and Aspuru-Guzik A 2017 Quantum neuron: an elementary building block for machine learning on quantum computers (arXiv:1711.11240)
[42] Killoran N, Bromley T R, Miguel Arrazola J, Schuld M, Quesada N and Lloyd S 2019 Continuous-variable quantum neural networks *Phys. Rev. Res.* **1** 033063
[43] van Loock P *et al* 2008 Hybrid quantum computation in quantum optics *Phys. Rev.* A **78** 022303
[44] Zhu D *et al* 2019 Training of quantum circuits on a hybrid quantum computer *Sci. Adv.* **5** eaaw9918
[45] Endo S, Cai Z, Benjamin S C and Yuan X 2021 Hybrid quantum-classical algorithms and quantum error mitigation *J. Phys. Soc. Japan.* **90** 032001
[46] Arthur D *et al* 2022 A hybrid quantum-classical neural network architecture for binary classification (arXiv:2201.01820)
[47] Rosmanis A 2024 Hybrid quantum-classical search algorithms *ACM Trans. Quantum Comput.* **5** 1–18
[48] Qiskit Contributors 2023 *Qiskit: An Open-Source Framework for Quantum Computing* (Zenodo)
[49] Zuo Y *et al* 2019 All-optical neural network with nonlinear activation functions *Optica* **6** 1132–7
[50] Shao Y, Hellstrom M, Mitev P D, Knijff L and Zhang C 2020 Pinn: A python library for building atomic neural networks of molecules and materials *J. Chem. Inf. Model.* **60** 1184–93
[51] Zhang H *et al* 2021 An optical neural chip for implementing complex-valued neural network *Nat. Commun.* **12** 457
[52] Wang T, Shi-Yuan M, Wright L G, Onodera T, Richard B C and McMahon P L 2022 An optical neural network using less than 1 photon per multiplication *Nat. Commun.* **13** 123
[53] Spall J, Guo X and Lvovsky A I 2022 Hybrid training of optical neural networks *Optica* **9** 803–11
[54] Schuld M and Killoran N 2019 Quantum machine learning in feature Hilbert spaces *Phys. Rev. Lett.* **122** 040504
[55] Cerezo M *et al* 2021 Variational quantum algorithms *Nat. Rev. Phys.* **3** 625–44
[56] Liu J-G and Wang L 2018 Differentiable learning of quantum circuit born machines *Phys. Rev.* A **89** 062324
[57] Sim S, Johnson P D and Aspuru-Guzik A 2019 Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms *Adv. Quantum Technol.* **2** 1900070
[58] van Hemmen J L and Kühn R 1986 Nonlinear neural networks *Phys. Rev. Lett.* **57** 913
[59] Grossberg S 1988 Nonlinear neural networks: principles, mechanisms and architectures *Neural Netw.* **1** 17–61
[60] Huang D-S and Ma S-D 1999 Linear and nonlinear feedforward neural network classifiers: a comprehensive understanding *J. Intell. Syst.* **9** 1–38
[61] Wang R, Shivanna R, Cheng D, Jain S, Lin D, Hong L and Chi E 2021 Dcn v2: improved deep & cross network and practical lessons for web-scale learning to rank systems *Proc. web Conf. 2021* pp 1785–97
[62] Rendle S 2010 Factorization machines *2010 IEEE Int. Conf. on Data Mining* (IEEE) pp 995–1000
[63] Eberhart R and Kennedy J 1995 Particle swarm optimization *Proc. IEEE Int. Conf. on Neural Networks* vol 4 pp 1942–8
[64] Wang H and Jay Guo L 2022 Neutron: neural particle swarm optimization for material-aware inverse design of structural color *iScience* **25** 104339
[65] Lester James V M 2018 PySwarms, a research-toolkit for Particle swarm optimization in python *J. Open Source Softw.* **3** 433
[66] Kingma D *et al* 2014 A method for stochastic optimization *1412* (arXiv:1412.6980)
[67] Reddi S J, Kale S and Kumar S 2019 On the convergence of adam and beyond (arXiv:1904.09237)
[68] Blake C 1998 Uci repository of machine learning databases (available at: www. ics. uci. edu/~ mlearn/MLRepository. html)
[69] Cai F *et al* 2020 Power-efficient combinatorial optimization using intrinsic noise in memristor hopfield neural networks *Nature Electron.* **3** 409–18
[70] Temme K, van den Berg E, Kandala A and Gambetta J 2022 With fault tolerance the ultimate goal, error mitigation is the path that gets quantum computing to usefulness (available at: https://quantum-computing.ibm.com)
[71] IBM Quantum (available at: https://quantum-computing.ibm.com)
[72] Srivastava N, Hinton G, Krizhevsky A, Sutskever I and Salakhutdinov R 2014 Dropout: a simple way to prevent neural networks from overfitting *J. Mach. Learn. Res.* **15** 1929–58
[73] Ioffe S and Szegedy C 2015 Batch normalization: accelerating deep network training by reducing internal covariate shift *Int. Conf. on Machine Learning* pmlr, pp 448–56
[74] Jimmy Lei B, Ryan Kiros J and Hinton G E 2016 Layer normalization (arXiv:1607.06450)
[75] Szegedy C, Ioffe S, Vanhoucke V and Alemi A 2017 Inception-v4, inception-resnet and the impact of residual connections on learning *Proc. of the AAAI Conf. on Artificial Intelligence* vol 31
[76] Gao Z, Cunbao M, Song D and Liu Y 2017 Deep quantum inspired neural network with application to aircraft fuel system fault diagnosis *Neurocomputing* **238** 13–23
[77] Li P and Wang B 2020 Quantum neural networks model based on swap test and phase estimation *Neural Netw.* **130** 152–64
[78] Huang R, Tan X and Qingshan X 2021 Variational quantum tensor networks classifiers *Neurocomputing* **452** 89–98
[79] Dong Y, Xie J, Wanbin H, Liu C and Luo Y 2022 Variational algorithm of quantum neural network based on quantum particle swarm *J. Appl. Phys.* **132** 104401
[80] Yu Z, Yao H, Li M, and Wang X 2022 Power and limitations of single-qubit native quantum neural networks (arXiv:2205.07848)