IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 21 December 2024, accepted 21 February 2025, date of publication 3 March 2025, date of current version 10 March 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3546956

RESEARCH ARTICLE

NR-QNN: Noise-Resilient Quantum Neural Network

SOHRAB SAJADIMANESH[1], HANIEH AGHAEE RAD[2], JEAN PAUL LATYR FAYE[2,3], AND EHSAN ATOOFIAN[1], (Senior Member, IEEE)

[1]Electrical and Computer Engineering Department, Lakehead University, Thunder Bay, ON P7B 5E1, Canada
[2]CMC Microsystems, Kingston, ON K7L 1H3, Canada
[3]Department of Physics, Faculty of Science and Technology, Cheikh Anta Diop University, Dakar 10700, Senegal

Corresponding author: Ehsan Atoofian (atoofian@lakeheadu.ca)

This work was supported by Mitacs through the Mitacs Accelerate Program.

ABSTRACT Quantum Neural Networks (QNNs) based on parameterized quantum circuits (PQCs) are gaining significant research attention due to their potential to achieve quantum advantages on Near-Term Noisy Intermediate-Scale Quantum (NISQ) computers. However, executing QNNs on NISQ devices is challenging due to quantum noise. To address this, we propose a noise-resilient QNN (NR-QNN) that leverages the unique characteristics of PQCs to perform noise-aware optimizations during the inference stage of QNNs. Specifically, NR-QNN employs two optimization techniques to mitigate the impact of noise on QNNs: quantum pruning and sensitivity-aware qubit mapping. The first technique is quantum pruning which identifies gates with small angle of rotations and removes them to simplify circuit of QNNs. The second optimization technique is sensitivity-aware qubit mapping which maps more important logical qubits to more reliable physical qubits. This technique is based on the observation that there can be variation in the sensitivity of an output to input qubits in a QNN. NR-QNN exploits this variability and guides qubit allocation to use reliable physical qubits for sensitive logical qubits. Our evaluation on a real quantum computer demonstrates that NR-QNN enhances the robustness of QNNs, enabling them to operate effectively on NISQ devices.

INDEX TERMS Quantum computing, NISQ device, quantum noise.

I. INTRODUCTION

Quantum computing (QC) is a new computing paradigm that can offer substantial computational speed and efficiency beyond the capability of classical computing. Quantum algorithms use quantum bits (qubits) to exploit properties of quantum mechanics such as superposition and entanglement and rely on quantum gates to change the states of the qubits. It has been shown that QC provides exponential or polynomial advantages in various domains such as cryptography [1], database search [2], quantum chemistry [3], [4], etc. In the realm of machine learning, QC can significantly enhance neural network models, leading to quicker and more precise predictions [5]. This advancement has the potential to enhance various consumer applications, including personal assistants [6], image [7] and speech recognition [8], and natural language processing [9]. The field of QC is currently undergoing a transition from purely theoretical research area into an industrial technology thanks to some companies such as IBM [10], Google [11], etc. offering real quantum computers with 10's to 100's of qubits. These prototype quantum computers enable researchers to understand the challenges in QC and use these insights to improve the design of future QC systems. Recently, Google claimed quantum supremacy on their 53-qubit Sycamore Quantum Processing Unit where a quantum sampling problem is completed in 200 seconds [12] while running the classical version of the same problem on the state-of-the-art supercomputers may take 10,000 years.

While many quantum algorithms can potentially offer substantial speed-up over their classical counterparts, quite often, execution of these algorithms on contemporary quantum computers is not feasible due to excessive hardware noises. A qubit is fickle which causes losing its state.

The associate editor coordinating the review of this manuscript and approving it for publication was Siddhartha Bhattacharyya.

© 2025 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

This type of error which is related to decay of qubit states is called coherence error. The other type of error is related to quantum gates. Quantum gates are not perfect and may generate outputs that differ from the correct ones. Gate errors are relatively small for single-qubit gates (e.g., 10^{-4} to 10^{-3} for IBM quantum computers), but can be significant for two-qubit gates (e.g., 10^{-3} to 10^{-2} for IBM quantum computers) [13]. Once quantum computation of a quantum circuit is finished, the outputs of the circuit are measured to convert the state of qubits to classical data. This step also causes error mainly due to measurement sensing. Additionally, non-coherent interference such as crosstalk [14], [15] can impose computational error. Qubits can be protected against errors using quantum error correction codes (QECs) [16]. However, QECs are costly as they incur significant overhead. A QEC requires 10-100 physical qubits to encode a fault-tolerant qubit. Existing quantum computers with 10's to 100's of qubits do not have the capacity to utilize QEC for even a moderate sized quantum circuit. Such quantum computers with noisy qubits are called Noisy Intermediate Scale Quantum computers (NISQ) [17]. Even though NISQ devices are unable to support QEC, they can still provide benefits for a class of applications [18].

Due to high error rate and lack of enough qubits in supporting QEC on NISQ devices, researchers have explored alternative avenues for quantum algorithms that can harvest the quantum advantages of NISQ devices. The variational quantum algorithm (VQA) [19], [20] stands out as one of the most prominent NISQ algorithms which employs a classical optimizer to train circuit parameters and generate desired outputs. VQA has been applied to several application domains. One of the increasingly popular domains is machine learning. Machine learning is a data-driven decision-making method in which a computer fits a mathematical model to data during the training phase and uses the model to derive decisions during the inference phase. A common method for implementation of a quantum machine learning algorithm is a quantum neural network (QNN) that has gained significant attention over the past few years [21], [22], [23]. Figure 1 shows the structure of a QNN. A QNN consists of three parts: data encoding, computation, and measurement. Data encoding converts classical data into quantum states. There are a variety of methods for data encoding. One way of classical to quantum conversion is to encode data into amplitude of a quantum state [22]. An n-qubit register can encode 2^n classical data. While amplitude encoding benefits from the exponentially large Hilbert space, it requires non-trivial quantum circuits to realize amplitude encoding. This leads to resource-intensive circuits which cannot be implemented on existing NISQ devices. Access to the states that encode data can be done efficiently by using a quantum random access memory (QRAM) [24]. However, implementation of a QRAM is a challenging task and currently, no practical circuit exists for QRAMs. An alternative approach is phase encoding where classical

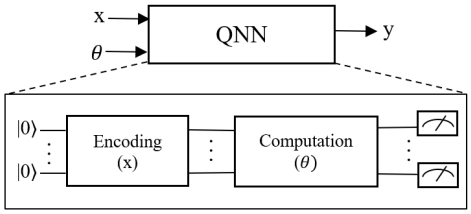

FIGURE 1. Structure of QNN (x is input data, θ is trainable parameter).

data are encoded through angle of quantum rotation gates. Phase encoding requires input data to first be mapped to angles between 0 and 2π. We use the second approach for data encoding in this paper as it requires a simple quantum circuit to realize on a NISQ device. The next step in the QNN is computation. The computation part of the QNN manipulates the encoded data to generate desired output states. One approach to realize computation part in the context of a quantum circuit is using multiply-and-accumulate operation, similar to classical neural networks (NNs). However, implementation of a quantum multiplier is costly as it requires a large number of quantum gates with high quantum depth [25]. A more practical approach is using a parametrized quantum circuit (PQC) whose parameters are used to construct a cost function that should be minimized using a classical algorithm such as back-propagation. The PQC approach is more convenient for NISQ devices since it requires a short-depth circuit and its parametric nature makes it more resilient to quantum noises. The last step in the QNN is measurement where the states of outputs in the computation part collapse to classical data. Prior studies have demonstrated that QNN models are more expressive than classical NNs [26], [27], [28]. In other words, a QNN has a higher capability in approximating a desired task (such as classification of data) in comparison with classical NNs with similar size (the same number of network parameters/weights). Recent works showed application of QNNs in medical image analysis [29], drug discovery [30], finance [31], and many other industrial problems [32].

Despite the success of QNNs in machine learning algorithms, their deployment into contemporary quantum computers is challenging as NISQ devices are noisy. Increasing the number of parameters in PQCs boosts accuracy of QNNs the same way that a larger set of neurons in classical NNs increases accuracy of predictions. However, more parameters mean a higher number of quantum gates which potentially increases depth of QNNs and inevitably error rate. Thus, reducing the impact of noise on QNNs is of pressing demand which helps to close the gap between the requirement of quantum machine learning algorithms and capacity of NISQ devices. Researchers have proposed noise mitigation techniques to reduce the impact of noise on quantum circuits. Noise adaptive qubit mapping [33], [34], [35] optimizes logical to physical qubit mapping to minimize

circuit errors. Noise adaptive instruction scheduling and crosstalk mitigation techniques [14], [36] focus on inter-qubit interference and circuit depth to mitigate quantum noises. However, these are general techniques and do not consider unique characteristics of QNNs.

This paper proposes noise-resilient quantum neural network (NR-QNN) which is a set of QNN-specific noise mitigation techniques to increase the intrinsic robustness of PQCs and improve accuracy on real quantum computers. NR-QNN comprises two optimization techniques. The first technique is quantum pruning where the circuit of a QNN is simplified to reduce the number of quantum gates and circuit depth. Pruning is a popular technique in classical machine learning to reduce computing requirements as well as memory footprints of an NN. We prune PQC gates with small parameters to simplify the corresponding QNN circuit. To assure that the accuracy of a pruned QNN is not affected, we finetune the QNN after each round of pruning to recover the original performance of the network. At the end of pruning, we end up with a slim circuit with similar noise-free performance but fewer gates and shallower circuit depth which in turn increases the robustness of the network to hardware noises. The second technique is sensitivity-aware qubit mapping where qubits of a QNN are mapped to physical qubits based on characteristics of logical and physical qubits. The error rate in physical qubits is not the same. Instead, the error rate varies across different physical qubits by a large margin [13], [26], [35]. For example, our detailed analysis of IBM-Brisbane quantum computer shows that [13] the coherence time across all qubits can vary by as much as 8x. Such variation can have a significant impact on the overall system reliability. On the software side, not all logical qubits are equally important. In the context of QNNs, the sensitivity of outputs to input qubits varies. While small changes in some input qubits may cause a QNN to misclassify, the QNN may still predict correctly in the presence of moderate or even large noises on other qubits. We exploit this sensitivity variability in QNNs and map more sensitive logical qubits to more robust physical qubits. It is important to note that while prior works used a variety of techniques for qubit mapping to reduce noise in quantum circuits [33], [34], [37], to the best of our knowledge, this is the first work that exploits sensitivity characteristics of QNNs to steer logical to physical mapping on NISQ devices.

Overall, NR-QNN exploits the combination of pruning and sensitivity-aware qubit mapping to mitigate the impact of quantum noise on QNNs. In summary, the contributions of NR-QNN are four-fold:

1) We exploit pruning to remove redundant gates and simplify QNNs. To mitigate the impact of pruning on accuracy, we finetune PQC parameters after each round of pruning. We show that pruning reduces the number of gates up to 63%.
2) We analyze sensitivity of QNNs to input qubits and show that not all qubits are equally important for outputs. We exploit this variability and propose mapping more sensitive logical qubits to less noisy physical qubits.
3) We extensively evaluate NR-QNN with popular datasets in quantum machine learning on real quantum computers and show that the combination of pruning and sensitivity-aware qubit mapping overcomes noise in NISQ devices and enables QNNs to generate meaningful results on contemporary quantum computers.

The rest of the paper is organized as follows. We discuss related works in Section II. Section III presents details of NR-QNN. Section IV describes evaluation methodology and reports the results. Finally, Section V draws our conclusions.

II. RELATED WORKS

Quantum computing promises computational advantages for some applications [38], [39]. In the absence of quantum error correction on contemporary quantum computers, optimization techniques play a vital role in closing the gap between the device and the application. Thus, mitigation of hardware errors through design of highly optimized quantum circuits is an active area of research.

A large body of prior work exists on quantum circuits optimization to reduce the total number of gates or number of layers [40], [41]. Maslov et al. [40] optimize a quantum circuit without considering hardware constraints. A quantum circuit is broken into subcircuits and then each subcircuit is processed to be replaced by an equivalent circuit that has lower cost. In the next step, a greedy algorithm is employed to compact quantum layers. A layer is defined as a set of gates that can operate in parallel. Layers in a circuit are formed one-by-one, starting from primary inputs. Each time that a new layer is created, it is analyzed to determine whether it can be merged with existing layers. The shortcoming of this technique is that it assumes that the error across qubits and gates is the same. Qubit placement [41] is an NP-complete problem and requires an effective heuristic-based algorithm to map a quantum circuit to hardware. The algorithm proposed in [41] starts with a non-optimized basic mapping where a quantum circuit can operate on hardware. In the next step, a hill climbing algorithm tries to match a qubit with the optimum physical gate. Each time, a new placement assignment is created by mapping a qubit to a new physical gate and is compared with the existing mapping. If it is less costly, then the new placement replaces the old one; otherwise, it moves on to the next qubit. Our optimization techniques are different as we focus on parametric circuits where operation of the circuits can be adjusted by training parameters of the circuits. On the contrary, the above optimization techniques are designed primarily for circuits with fixed and non-trainable gates.

Murali et al. [14] mitigate the impact of crosstalk noises in quantum circuits through software techniques. Crosstalk occurs when multiple quantum gates operate simultaneously, corrupting the state of qubits. Crosstalk arises from fundamental challenges in quantum hardware such as leakage of control signals (needed for gate operations) onto qubits

which are not part of the intended hardware operation. An approach to prevent interference between two high crosstalk components is to schedule them serially by using control instructions such as barrier. However, due to fragile nature of qubits, serialization of quantum operations causes loss of quantum information. Murali et al. [14] develop a software module that serializes some of crosstalk operations but also balances the need to avoid exponential decoherence error due to serialization. As an example, only those gate pairs that are separated by one hob are serialized as crosstalk talk noise for gates with distance of more than one hob is negligible. As a result, when two pairs are separated by two or more hobs, they can be executed in parallel to reduce the impact of serialization on coherence time.

Ding et al. [36] propose a frequency-aware software technique to reduce crosstalk noise via dynamic frequency tuning. In superconducting quantum computers, two qubits interact with each other through resonance of qubit frequencies. There are two main approaches to prevent accidental interference of resonance frequencies. The first one exploits tunable qubits where frequencies of qubits are set to be apart so that the probability of frequency interference reduces. The second method uses tunable couplers and temporarily disables connections that cause interference. IBM Q system uses fixed frequencies for all qubits and fixed couplers across all links and leaves it to a scheduler to avoid crosstalk noises [14]. On the contrary, Google uses tunable qubits with either fixed or tunable couplers [42]. Higher tunability offers more flexible hardware and provides more control over devices. However, it induces higher hardware overhead and causes sensitivity to control noise. Ding et al. [36] introduce a balanced design, i.e., qubit frequencies are tunable, but couplers are fixed. This approach offers a high program success rate via software which maps frequency decision to coloring of crosstalk graph. A vertex in the crosstalk graph represents a qubit and an edge represents a link, i.e. a capacitor in the frequency-tunable architecture. When the qubits are idle, we would like to have different frequencies for every pair of connected qubits. In the context of graph theory, it is equivalent to coloring the connectivity graph where end points of an edge have different colors. If the graph can be colored with C colors, then the qubits require C different frequencies to avoid interference. Crosstalk mitigation techniques can be combined with our optimization techniques to reduce noise in QNNs, further.

There have been numerous efforts to build tools for implementation of quantum circuits on hardware so that the noise level is contained. As an example, variation-aware qubit allocation (VQA) [35] takes into account error rate of links to map logical to physical qubits. This is in contrast to some other mapping techniques which are oblivious to the variation in the link reliability [33]. VQA starts with an initial mapping and then tries to converge to a configuration with minimum inter-qubit errors. To do so, it estimates the most frequently used qubits by analyzing the first n quantum gates and calculating frequency of accesses to qubits. Then, it maps the most frequently used qubits to the physical qubits with the most reliable links to reduce noise in quantum circuits. The downside of VQA is that it assumes that qubits are equally important for outputs. However, in QNNs, the sensitivity of an output varies from one input qubit to the other. We exploit this variability and propose a mapping scheme that uses more reliable hardware resources for qubits with higher sensitivities. As such, we were able to remove some of hardware noises that could not be eliminated by VQA.

QunatumNAS [43] is an evolutionary-based technique to reduce noise in variational quantum circuits. QunatumNAS first constructs SuperCircuit by grouping a sufficient number of quantum layers to cover a large fraction of a quantum circuit. Then, the SuperCircuit is broken into small sections called SubCircuits. QunatumNAS uses hardware noise information and relies on a genetic algorithm to find the most robust quantum circuits. A gene vector encodes a SubCircuit. Each element in the sub-gene represents the circuit width in a layer. The evolution engine searches for reliable circuits using a combination of mutation and crossover. Mutation randomly changes a subset of genes with a pre-determined probability. Crossover first picks two parents and then generates a new sample with genes which are sampled randomly from the parents. Thus, the new population is generated based on parent population, mutation, and crossover. Then, the most reliable circuit is selected from the new population. QunatumNAS can be integrated with our optimization techniques to reduce noise in QNNs, further.

Quantum on-chip training (QOC) [44] uses real quantum computers to accelerate training of PQCs. QOC uses parameter shift to obtain PQC gradients. Parameter shift calculates gradient of a parameter by simply shifting the parameter and calculating the difference between the corresponding outputs. Due to hardware noises, gradients obtained through parameter shift have low fidelity and thus reduce accuracy of training. QOC exploits pruning to mitigate the impact of noise on training. When the gradient magnitude is small, noises have more detrimental impact on signals. The unreliable gradients have harmful impact on training convergence as well as accuracy of the trained network. Skipping those unreliable gradients can mitigate the impact of noise on training of PQCs. Our approach is different as we focus on inference of QNNs and not training.

III. NOISE-RESILIENT QUANTUM NEURAL NETWORK (NR-QNN)

In this section, first, we explain the architecture of the QNN used in this work. Then, we discuss the details of NR-QNN.

A. ARCHITECTURE OF QNNS

A QNN consists of parametric quantum gates that should be trained to generate a desired input-output relationship. The

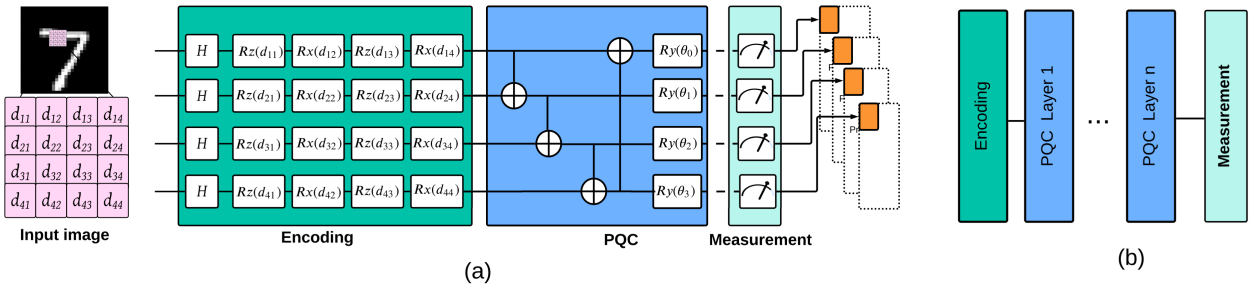

FIGURE 2. (a) Architecture of a QNN. (b) Multi-layer QNN.

QNN is modeled by $\Phi(x, \theta)$ where x is the input data and θ is a set of parameters for adaptive optimization. Since on IBM quantum computers, the default state for a qubit is $|0\rangle$, a QNN that is run on an IBM quantum computer maps $|0 \ldots 0\rangle$ to $\Psi(x, \theta)$ where $\Psi(x, \theta) = \Phi(x, \theta)\,|0 \ldots 0\rangle$.

Figure 2.a shows the structure of a single-layer QNN and figure 2.b shows a multi-layer QNN. The first step in the QNN is encoding where classical data are converted to quantum states. For a continuous variable, a popular technique is angle encoding where the continuous variable is encoded as an angle of rotation along the desired axis (X/Y/Z). As a state produced by a quantum rotation gate around any axis will repeat itself in 2π intervals, classical data are scaled to 0 to 2π range in a data preprocessing step. At the beginning of the encoding circuit, a Hadamard gate changes the state of an input qubit to a superposition of $|0\rangle$ and $|1\rangle$. This allows the encoding circuit to cover a larger subset of the Hilbert space.

One approach for angle encoding is using a separate qubit with one rotation gate for each classical data. This approach reduces the depth of the encoding circuit. However, for a classical input with n features, n qubits are needed. The other approach is encoding multiple continuous variables in a single qubit using sequential rotation [45]. While this approach reduces the number of qubits in the encoding circuit, it increases the depth of the circuit drastically. We use a combination of the two techniques to encode classical data. A classical input with n features (e.g., an input image with n pixels) is encoded with n_1 qubits where each qubit encodes n_2 input features using n_2 rotation gates ($n = n_1 \times n_2$).

The second step in a QNN is computing which is implemented using a PQC. A PQC contains a set of parametrized gates whose parameters are used to construct a cost function that should be minimized using a classical algorithm such as back-propagation. A QNN should be able to classify inputs accurately while having a low-depth circuit so that it can be realized on NISQ devices. The depth of a QNN with n qubits should be constrained to allow us to do inference with a number of elementary quantum operations that grow polylogarithmically with the width of the circuit. If a QNN circuit only relies on single-qubit gates, then it can exhaust only a small subset of the Hilbert space of n qubits. In other words, the set of output states that the circuit can reach is limited. In the context of machine learning, this limits the flexibility of classifiers. Similar to classical machine learning, the challenge of finding a generic QNN architecture is therefore to design a circuit of small depth that still creates powerful classifiers for different datasets. In an n-layer QNN, the PQC section is replicated n times.

An effective approach to designing a low-depth and accurate QNN is to consider circuits that prepare strongly entangled quantum states. An entangled circuit is capable of reaching wide corners in the Hilbert space. In other words, they have a better chance to project classical input x to state $\Psi(x, \theta)$ which generates the correct label y when it is measured. From a theoretical point of view, a classifier should be able to capture both short- and long-term correlations. There is clear evidence that a shallow circuit is suitable for this purpose if it is strongly entangled [46]. Figure 2 shows an example of an entangled QNN implemented using a PQC. The entanglement part is realized using a set of multi-qubit operations between the qubits such as CNOT to generate correlated states. The rotation gates in the PQC search through the solution space.

Entanglement in the PQC is considered as a workaround for the limitation established by the no-cloning theorem. Quantum computers cannot copy data and this restricts the design of quantum circuits as it limits the movement of quantum data. However, a classical computer does not have such a restriction. A similar quantum neuron can use quantum data only once. Entanglement makes it feasible to generate two qubits with the same state. As an example, a CNOT with target qubit in the state of $|0\rangle$ copies the state of the control to the target.

Once the outputs of a PQC are measured, they are fed to a dense layer [47]. The number of neurons in the dense layer is equal to the number of classes in the dataset. We consider a stochastic descent method for training the QNN. We choose a Multi-Class Cross-Entropy Loss function to evaluate the error of the QNN:

$$L(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}.log(\hat{y}_{ij}). \tag{1}$$

where n is the number of samples in the input dataset, C is the number of classes, y_{ij} is the correct label, and $\hat{y}_{ij}$ is the output of the QNN.

The training is an iterative process and searches for the best parameters in $\Phi(x, \theta)$. We train the QNN using the Adagrad gradient-based optimization algorithm. To compute optimum values for PQC parameters, the derivative of the QNN outputs with regard to inputs and parameters should be computed. There are a variety of techniques to compute the gradients. However, not all of them are suitable for hardware. As an example, the adjoint method [48] requires circuit intermediate state values to compute gradients that are not accessible when the circuit runs on hardware. A popular method to compute quantum gradients is parameter-shift [49]. It evaluates the target function at two distinct points to compute gradients. The parameter-shift method is implemented in the PennyLane [50] framework. In this work, we use PyTorch [51] and PennyLane [50] frameworks to train QNNs.

B. QUANTUM PRUNING

NISQ devices are vulnerable to quantum noise which may compromise the fidelity of the computation results. One of the sources of errors in quantum computers is coherence error. If the time that it takes for a quantum circuit to compute its output exceeds the coherence time of a qubit, then the output of the circuit is skewed with noise and is not reliable. The computation time of a quantum circuit is correlated with the depth of the circuits [13]. The depth of a circuit is determined by the critical path which is the longest path composed of serially connected quantum gates in a quantum circuit. Thus, reducing the depth of a quantum circuit is an effective way to decrease computation time which in turn reduces noise level. The other source of error in quantum circuits is quantum gates. Each quantum gate adds noise to its qubit, reducing the fidelity of the qubit. Thus, a quantum circuit with a smaller number of gates is more reliable.

An effective way to reduce the depth and the number of gates in QNNs is quantum pruning. We observe that for most rotation gates in the PQC of a QNN if the angle of rotation is far from zero for several iterations during the training phase of the QNN, it will likely stay far from zero in the next several iterations. Similarly, if the angle remains small for several iterations, it will likely stay small in the next several iterations. Thus, the angle of rotation reliably is predictable to some extent. We propose angle pruning to remove gates with a small angle of rotation. This method potentially reduces the depth of PQCs and reduces the probability of quantum computation time exceeding coherence time. In addition, quantum gate pruning reduces the number of gates which in turn decreases the impact of gate errors on fidelity of PQCs. It also decreases the number of parameters to be updated during the training phase and thus accelerates the training of QNNs.

A QNN is trained first to learn the parameters of the corresponding PQC. Next, we prune gates with small angles: all rotation gates with angles less than a threshold are removed from the network. Finally, we retrain the network to adjust the remaining angles so that they can compensate for pruned gates and maintain the accuracy of the original network. This process is repeated several times to remove gates as much as possible. During the retraining, it is better to retrain the angles with initial values left from the previous round of training rather than reinitializing the surviving gates from scratch. NNs contain fragile co-adapted features [52]: gradient descent is able to find a decent solution when a network is initially trained, but not after reinitializing network parameters from scratch and retraining them.

To put it in a formal way, let's assume $\Phi(x, \theta)$ represents a QNN where θ is the set of angles in rotation gates. Our objective is to prune the model $\Phi(x, \theta)$ so that rotation gates with small angles are removed. However, the accuracy of the original network is maintained:

$$\Phi(x, \theta) = \Phi(x, \theta_1) \tag{2}$$

where $\theta_1 \subset \theta$. The pruning strategy selects gates based on the magnitude of their angles. Mathematically:

$$f_{prune}(\theta_j; \lambda) = \begin{cases} 0 & \text{if } \theta_j < \lambda \\ \theta_j & \text{otherwise} \end{cases} \tag{3}$$

where λ is a predefined threshold. Let's define the sparsity/pruning ratio S of the pruned network as the fraction of angles that are zero:

$$S = \frac{\text{Number of pruned gates}}{\text{Total number of gates}} = \frac{n_{\text{pruned}}}{n_{\text{total}}} \tag{4}$$

where n_{pruned} is the number of pruned gates and n_{total} is the total number of rotation gates in the original network. The goal is to increase S while maintaining the accuracy of the original network. After pruning a set of weights, the model often undergoes fine-tuning to recover from any potential loss in accuracy. This involves retraining the network with the pruned angles frozen to allow the remaining angles to compensate for the missing gates and connections. Mathematically, after pruning and fine-tuning, the loss function may converge to a new local minimum:

$$\mathcal{L}_{\text{fine-tuned}} = \mathcal{L}(\theta_{\text{pruned}}) + \epsilon \tag{5}$$

where ϵ represents a small adjustment due to fine-tuning. The pruning process involves iterative rounds of pruning and fine-tuning, progressively increasing the sparsity S until no additional gates can be pruned.

Algorithm 1 describes quantum gate pruning for QNNs. We divide the training phase into n epochs and perform pruning periodically at the end of each epoch. First, the gradient of the cost function with regard to the parameters of the PQC is computed using the parameter shift technique [49] (line 11). Then, parameters are updated using the gradient descent method (line 12). Once a round of training with

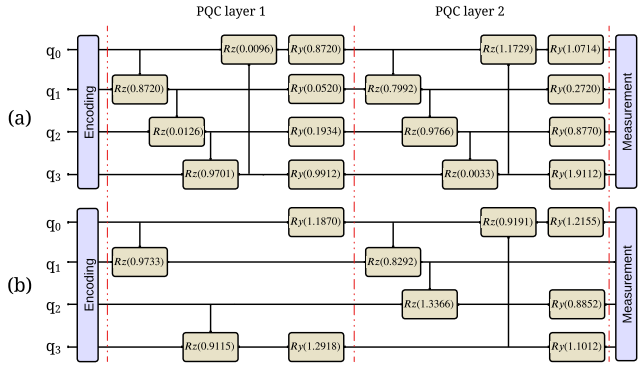

FIGURE 3. An example of QNN pruning for a circuit with two PQC layers where threshold is $1\pi/4 = 0.7854$. (a) Base QNN circuit. (b) QNN circuit after pruning and fine-tuning.

all inputs from the training dataset (S_{train}) is finished (lines 10-13), the pruning phase starts. The angle of each rotation gate in the PQC is compared with a predetermined threshold value (line 16). If the angle is less than the threshold, then the corresponding gate is removed from the circuit. Since pruning a subset of rotation gates from the quantum circuit may degrade accuracy, the network is retrained in the following epoch so that parameters of the remaining gates are updated to compensate for potential accuracy drop. Retraining the pruned network starting with parameters generated in the previous epoch requires less computation because it is not needed to backpropagate through the entire network. The pruned network is smaller which accelerates the training process.

Pruning rotation gates with small angle values is an iterative procedure. One approach for pruning would be training a QNN for n epochs and then pruning the network only once. This method reduces the overhead of pruning but is not able to reach the full potential of pruning. It may generate a network that is smaller than the original network but it may still contain a large number of unnecessary gates. Pruning a QNN after each epoch removes redundant gates gradually, enables finding a QNN with a smaller number of gates, and increases the success rate of the pruned network when it is deployed into a NISQ device. The iterative procedure described in algorithm 1 is very similar to the mammalian brain [53] where synapses are created in the first few months after a child is born. However, during the postnatal development of the child, synapses change based on their efficacy and activity. Synapses with no or little usage experience a postsynaptic gating mechanism which is similar to the pruning of gates with small angles of rotations. On the contrary, those synapses that are used frequently remain in the nervous system and their corresponding axon will carry larger electrical signals.

Figure 3 shows an example of QNN pruning for a circuit with two PQC layers using Algorithm 1. Figure 3.a is the original circuit and Figure 3.b is the pruned circuit. Pruning removes the rotation gates and controlled rotation gates with

Algorithm 1 Quantum Gate Pruning

1: INPUTS:
2: S_{train}: training set
3: C: cost function
4: η: learning rate
5: n: number of epochs
6: θ: angle of rotation
7: $f(\theta)$: output of PQC
8: thld: Threshold for pruning
9: **for** epoch $= 1, 2, \ldots, n$ **do**
10: **for** a batch B from S_{train} **do**
11: $\nabla_\theta C_B(\theta) = 0.5(\frac{\partial f(\theta)}{\partial \theta})^T(\frac{\partial C(\theta)}{f(\theta)})$
12: $\theta = \theta - \eta \nabla_\theta C_B(\theta)$
13: **end for**
14: //PRUNING PHASE
15: **for** each rotation gate $g(\theta)$ in PQC layers **do**
16: **if** $\theta <$ thld **then**
17: remove $g(\theta)$ from PQC layers
18: **end If**
19: **end for**
20: **end for**
21: OUTPUT: Pruned QNN

angles below a predefined threshold. In this example, the threshold is $1\pi/4 = 0.7854$. After each round of pruning, we retrain and fine-tune the circuit to recover accuracy. Pruning only affects the rotation gates in computation part (PQC layers) of the circuit and does not impact the rest of the circuit such as encoding part.

C. SENSITIVITY-AWARE QUBIT MAPPING

It is well-known that the error of qubits in NISQ devices changes by a large margin across physical qubits. Due to process variation, temperature drifts, and environmental impacts, the error of a qubit can vary by as much as $7x$ in IBM quantum computers [13]. Using a NISQ device effectively requires very efficient and near-optimal mapping of quantum algorithms onto the hardware. In this section, we propose a new method for mapping of QNNs onto NISQ devices.

A functional mapping of a quantum circuit onto NISQ hardware requires first an initial placement of the circuit qubits onto the hardware qubits in order to reduce error. Then, an effective strategy is needed to reduce the likelihood of decoherence and operational errors when the circuit runs on real quantum hardware. Our work performs mapping based on daily calibration data released by IBM in order to avoid using unrealistic qubit errors and to prioritize qubit positioning to reduce the likelihood of quantum errors. IBM calibrates quantum computers regularly to mitigate the impact of quantum errors on quantum circuits [13]. Daily calibrations include single- and two-qubit calibrations. Hourly calibrations deal with readout errors and stability checks. Figure 4 shows T_1, T_2, and Readout errors for all 127 qubits in IBM-Brisbane. A qubit in a high energy state

($|1\rangle$) has a natural tendency to decay to a low energy state ($|0\rangle$). The time that it takes for this transition is called T_1 coherence time. This type of error is similar to retention errors in classical computers. However, classical computers are only subject to bit-flip errors whereas quantum computers are also susceptible to phase change errors. The time constant associated with phase change is called T_2 coherence time. Phase error is the result of interaction between a qubit and the environment. Over the past decade, coherence time has improved drastically [54].

From the daily calibration log, we observe that T_1 coherence time varies up to $8x$. The average and standard deviation for T_1 coherence time are $245\mu s$ and $76.5\mu s$, respectively. The T_2 coherence time varies up to $26.2x$ and the average and standard deviation for T_2 coherence time are $158\mu s$ and $88.83\mu s$, respectively.

Readout error on IBM-Brisbane quantum device varies from 0.41×10^{-2} to 0.316. These fluctuations stem from material defects caused by the lithographic process used to manufacture qubits and are expected to be present in future generations of NISQ devices [55].

We use Qiskit 0.45.0 [56] as the baseline compiler for qubit mapping. The compiler takes into account the variability in link errors and assigns logical qubits to physical qubits so that the number of SWAPs is minimized. The main shortcoming of the Qiskit compiler as well as other qubit allocation techniques [33], [34], [37] is that none of them consider the variability of logical qubits. In other words, these mapping techniques are oblivious to the sensitivity of the output of a quantum circuit to input qubits. In the context of a QNN, not all input qubits are equally important. The sensitivity of the output of the QNN varies from one logical qubit to the other.

To be able to guide the mapping procedure based on the importance of logical qubits, we need a mechanism to calculate the sensitivity of output to individual qubits. Formally, the sensitivity of the output of a QNN to input qubit q_i can be defined as $\frac{\partial C}{\partial \theta_i}$ where C is the cost function of the QNN and θ_i is the angle of rotation for q_i. We can use the chain rule to compute the derivative of the cost function with regard to input parameters. Let's assume $\{C \rightarrow g_1^p \rightarrow \ldots \rightarrow g_n^p\}$ is the path that emerges from output towards q_i and g_i^p represents a quantum gate. Since there may be more than one such path, we use superscript p to distinguish different paths. The sensitivity is given by:

$$\frac{\partial c}{\partial \theta_i} = \sum_{p=1}^{N_p} \frac{\partial c}{\partial g_1^p} \frac{\partial g_1^p}{\partial g_2^p} \cdots \frac{\partial g_n^p}{\partial \theta} \tag{6}$$

Instead of going through the burden of computing derivatives for individual gates, we use a profiling approach to extract the sensitivity of logical qubits. For qubit mapping, the absolute value of the sensitivities is not important. We only need a mechanism that sorts logical qubits based on their relative sensitivity. Once a QNN is trained, we evaluate the sensitivity of output to individual inputs by increasing the angle of input rotation gates by a small value: Δ. It is important to note that the angles of all rotation gates used for the encoding of the qubit are increased by Δ. If the prediction made by the QNN changes, then we record $1/\Delta$ as sensitivity for input qubit q_i; otherwise, we increase the input angle by another Δ. We continue this procedure (each time, the angle is increased by Δ) until the prediction made by the QNN before and after the angle increase changes. Intuitively, Δ is a measure of the vulnerability of qubits to noises. An input qubit that requires a large increase in its angle to flip prediction has lower sensitivity. On the contrary, if the QNN predicts a different class by a small increase in an input angle, then the sensitivity of the qubit is high.

Algorithm 2 Logical to Physical Qubit Mapping

Steps:

1. Use IBM Qiskit compiler to generate initial mapping.
2. Use profiling to find sensitivity of individual qubits.
   - 2.1 For each input qubit, increase the angle of the corresponding rotation gates by Δ.
   - 2.2 if QNN predicts a different class, set sensitivity to $\frac{1}{n\Delta}$ (n is the number of times the angle is increased); otherwise go to step 2.1.
3. Sort logical qubits in descending order of sensitivity.
4. Sort physical qubits in descending order of reliability extracted from IBM calibration data.
5. Map the sorted logical qubits to the sorted physical qubits one by one.

Once we compute the sensitivity of all input qubits, we are ready to map logical qubits to physical qubits. Since the Qiskit compiler optimizes the number of SWAP gates, we use the same physical qubits selected by the compiler for the QNN. We only change the mapping of logical qubits to the selected physical qubits. For example, figure 5 shows the topology of qubits in IBM-Brisbane. The four qubits contained in a red box are selected by the compiler for implementation of a QNN with 4-qubit. We use the same qubits but change the logical to physical mapping based on the sensitivities of the logical qubits. The reliability of the physical qubits is determined based on calibration data released by IBM and includes T_1 coherence, T_2 coherence, and measurement errors. A more sensitive logical qubit is mapped to a more reliable physical qubit. Algorithm 2 shows steps in sensitivity-aware qubit mapping.

IV. EVALUATION

In this section, we present results for NR-QNN. We compare the accuracy of QNNs in noiseless simulation, hardware emulation, and NISQ hardware. The goal of this work is to optimize QNN circuits so that they overcome noise when they are run on NISQ devices and generate meaningful results. We do not focus on QNN designs with the highest accuracies as the accuracy of QNNs is not the focus of this work.

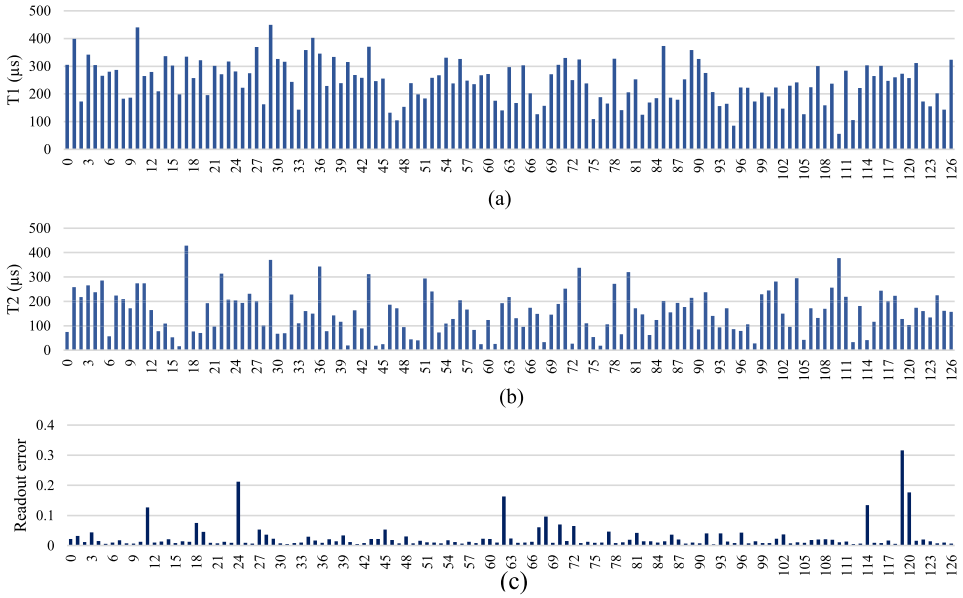

FIGURE 4. Distribution of a) T_1, b) T_2, and c) read-out errors over 127 qubits for IBM-Brisbane [13].

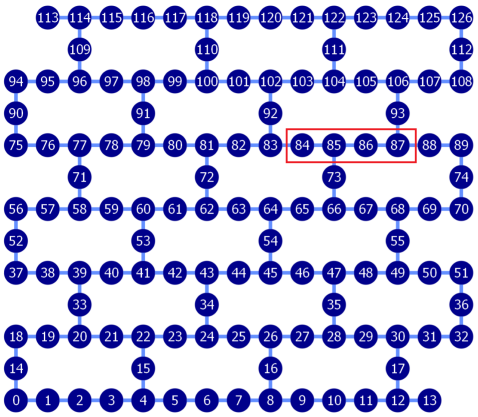

FIGURE 5. IBMQ-Brisbane layout [13].

A. EVALUATION METHODOLOGY

We conduct experiments using MNIST, Fashion-MNIST, and CIFAR-10 datasets. MNIST and Fashion-MNIST datasets contain 28 × 28 single channel images and both share the same number of classes for classification. MNIST dataset contains images of handwritten digits from zero to nine. Fashion-MNIST dataset includes images of different pieces of clothing. Each of the two datasets is divided into 50,000 and 10,000 images for training and testing, respectively. The CIFAR-10 dataset contains RGB images with size 3 × 32 × 32 pixels. CIFAR-10 has 50,000 images for training and 10,000 images for testing. CIFAR-10 consists of ten different classes: airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. We use average accuracy as a metric to measure the performance of QNN models. The average accuracy is calculated by performing a forward pass through the network for each sample in the test dataset, measuring the fraction of the samples that are classified correctly.

For MNIST and Fashion-MNIST, we use QNNs with 4 qubits [47]. The encoding section of the QNNs uses 4 rotation gates per qubit (Figure 2.a). As a result, for a 28 × 28 image, $7 \times 7 = 49$ quantum circuit executions are required to generate $4 \times 49 = 196$ features for the next fully connected layer. For CIFAR-10, we use 6 qubits where each qubit uses two rotation gates for the encoding section [47]. Thus, each $3 \times 32 \times 32$ image requires $16 \times 16 = 256$ circuit executions to generate $6 \times 256 = 1536$ output features. After measurements, the output features are concatenated and flattened for the next fully connected layer.

We use PyTorch [51], PennyLane [50], and Qiskit [56] packages to model and train QNNs used in this work. All networks are trained with Adagard optimizer [57] with a learning rate of 0.5. We use the default PennyLane configuration to run QNNs in noiseless simulations. For hardware emulation, we use fake backends provided in Qiskit V0.45.0 for IBM quantum computers. The fakebackend is used to model noise in quantum hardware. The noise is based on IBM calibration data and involves T_1 and T_2 coherence errors, gate errors, and read-out (measurement) errors. Several techniques are used to model coherence errors such as Inversion Recovery, Ramsey Experiments, and Hahn Echoes [58]. Errors in single-qubit gates are simulated using a single-qubit depolarization error followed by a single-qubit relaxation error. Similarly, errors in a two-qubit gate are simulated by applying depolarization error and thermal relaxation error on both qubits of the gate. Each measured qubit is flipped with a probability to model the impact of

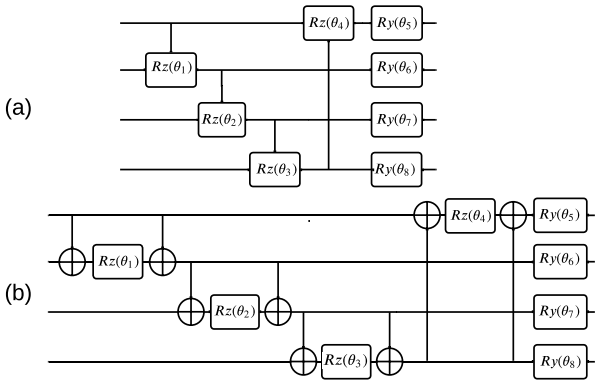

FIGURE 6. Two flavors of PQCs for MNIST and Fashion-MNIST.

measurement error. IBM provides access to calibrated data through Qiskit APIs. During the simulation, the hardware emulator loads these noises, compiles the circuit with native gates, and generates outputs under the aforementioned noise model. We use the hardware emulator to compare the performance of various QNNs under noise. We also run QNNs on a 127-qubit real IBM quantum computer: IBM-Brisbane. The IBM quantum computers use the Pauli-Z expectation values for the measurement of the outputs. Each QNN is run for 20,000 shots on the IBM-Brisbane.

B. EXPERIMENTAL RESULTS

Figure 2 is the skeleton of all QNNs used in this work. We change the configuration of the PQC section of the QNNs to evaluate the impact of different quantum circuits on accuracy. By no means, this is not an exhaustive exploration of design space for PQCs. We only evaluate a few configurations and select the best one for the rest of this paper.

The PQC in figure 2 consists of CNOT and R_y gates. Figure 6 shows two other configurations for PQCs for MNIST and Fashion-MNIST. The first one is built out of controlled R_z and R_y gates and the second one uses blocks of CNOT and R_z gates. Regardless of the configuration of PQCs, the circuits must include two-qubit gates to benefit from entanglement and thus cover larger regions of Hilbert space.

The images in MNIST and Fashion-MNIST are single channels as all images in the two datasets are black-and-white. However, the CIFAR-10 images are colored and each image has three channels: R, G and B. To exploit the correlation between the output of CIFAR-10 QNN and the three input channels, we use PQCs that differ from the configurations presented in figure 6. Figure 7 shows PQCs for CIFAR-10. While the intra-channel PQC (figure 7.a) uses entanglement between qubits within a channel, the inter-channel PQC (figure 7.b) exploits entanglement across channels. The inter-/intra-channel PQCs (figures 7.c and 7.d) exploit entanglements both within and across channels and thus it is expected that they offer higher accuracies. As we already mentioned, the goal of this work is to optimize QNNs to overcome noise in NISQ devices. Exploring the design

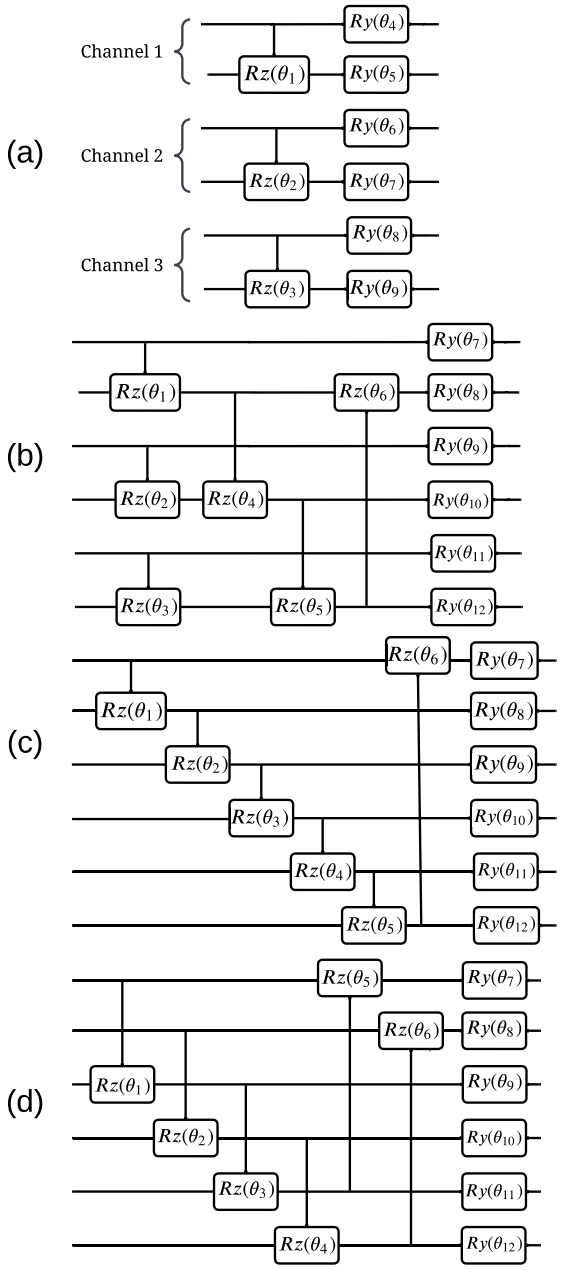

FIGURE 7. PQCs for CIFAR-10. (a) Intra-Channel. (b) Inter-Channel. (c) and (d) Intra/Inter-Channel.

space of PQCs and offering the optimum circuit is beyond the scope of this work.

Table 1 shows the average accuracy of QNNs for MNIST, Fashion-MNIST, and CIFAR-10. Accuracies of MNIST and Fashion-MNIST are higher than CIFAR-10. This is mainly due to the simplicity of the two datasets. Both datasets exploit black-and-white images. Based on Table 1, we use CR_Z gates for both MNIST and Fashion-MNIST. In CIFAR-10, intra/inter-1 and intra/inter-2 offer higher accuracies. This is expected as they exploit correlations both within and across channels and are able to differentiate input images better than

TABLE 1. Accuracy of Noiseless QNNs.

Method	MNIST	FashionMNIST
CRZ (Fig. 6.a)	97.50%	79.25%
CNOT (Fig. 2)	88.33%	71.66%
CNOT+RZ (Fig. 6.b)	94.17%	78.33%
	CIFAR10	
Intra	25.83%	
Inter	30.83%	
Intra/Inter-1	38.33%	
Intra/Inter-2	40.83%	

TABLE 2. Similarity between Pruned QNNs and Noiseless Simulations and Pruning Rate. The number of layers changes from one to four ($L_1 \sim L_4$).

Layers	Pruning Angle	Similarity (%) / Pruning rate (%) MNIST	FashionMNIST	CIFAR10
L_1	$\pi/4$	87%/12%	87%/13%	**87%**/16%
	$2\pi/4$	87%/13%	**90%**/13%	83%/17%
	$3\pi/4$	**87%**/37%	83%/50%	80%/42%
	$4\pi/4$	83%/38%	83%/50%	80%/58%
	$5\pi/4$	80%/74%	80%/63%	80%/67%
	$6\pi/4$	80%/75%	83%/88%	80%/92%
L_2	$\pi/4$	83%/13%	**83%**/13%	74%/13%
	$2\pi/4$	**87%**/31%	80%/31%	73%/21%
	$3\pi/4$	83%/44%	80%/56%	73%/46%
	$4\pi/4$	83%/69%	80%/63%	70%/54%
	$5\pi/4$	77%/93%	73%/75%	**77%**/63%
	$6\pi/4$	77%/94%	73%/88%	70%/79%
L_3	$\pi/4$	**83%**/8%	**73%**/17%	**77%**/11%
	$2\pi/4$	77%/25%	67%/21%	70%/33%
	$3\pi/4$	77%/42%	73%/29%	70%/53%
	$4\pi/4$	77%/46%	63%/46%	67%/69%
	$5\pi/4$	73%/54%	63%/75%	63%/72%
	$6\pi/4$	73%/79%	63%/88%	60%/83%
L_4	$\pi/4$	**83%**/9%	**67%**/9%	**70%**/17%
	$2\pi/4$	77%/22%	63%/34%	60%/31%
	$3\pi/4$	73%/44%	63%/53%	60%/44%
	$4\pi/4$	70%/56%	63%/56%	60%/63%
	$5\pi/4$	67%/72%	53%/75%	50%/67%
	$6\pi/4$	67%/78%	53%/84%	50%/79%

intra and inter-schemes. We use intra/inter-2 for CIFAR-10 as its accuracy is slightly higher than intra/inter-1 scheme.

Table 2 compares predictions made by fake-backend and noiseless simulations. The goal of NR-QNN is to mitigate the impact of quantum noise on the accuracies of QNNs. Thus, Table 2 reports similarity between pruned QNNs and baseline noiseless QNNs. For example, the similarity of 90% in Table 2 shows that 90% of outputs in a pruned QNN on fake-backend is the same as baseline noiseless QNN. We also report the fraction of gates that are removed due to pruning.

To evaluate how our proposed techniques work in deep circuits, we replicate the PQC section of QNNs. For example, L_3 in Table 2 indicates that the PQC has three layers. In other words, the PQCs in figures 6.a and 7.d are replicated three times. It is important to note that the fake-backend does not model the entire spectrum of quantum noise and as a result, the output of a quantum circuit on a real quantum computer is not necessarily the same as the fake-backend simulation. It is used to provide a first-order estimation of how a circuit behaves when it is deployed on a real quantum computer.

We prune QNNs with thresholds of $\pi/4 \sim 6\pi/4$ with strides of $\pi/4$ in Table 2. While increasing the threshold reduces the cost of the pruned circuit as a greater number of gates are eliminated, a higher threshold does not necessarily offer a higher level of accuracy. As a PQC shrinks due to pruning, a smaller subset of Hilbert space is covered by the circuit which makes it more difficult for the steepest descent algorithm to optimize the network. For each dataset, we select the highest accuracy per PQC layer and use it for the rest of the paper (entries with bold fonts in Table 2).

TABLE 3. Running MNIST, Fashion-MNIST, and CIFAR-10 on IBM-Brisbane.

	Layers	Pruning Angle	Baseline QNN	Pruning QNN	NR-QNN
MNIST	L_1	$3\pi/4$	93%	100%	100%
	L_2	$2\pi/4$	77%	87%	93%
	L_3	$1\pi/4$	63%	80%	90%
	L_4	$1\pi/4$	57%	73%	80%
Fashion-MNIST	L_1	$2\pi/4$	67%	80%	90%
	L_2	$1\pi/4$	63%	77%	83%
	L_3	$1\pi/4$	57%	63%	80%
	L_4	$1\pi/4$	43%	57%	77%
CIFAR-10	L_1	$1\pi/4$	67%	80%	87%
	L_2	$5\pi/4$	57%	67%	77%
	L_3	$1\pi/4$	53%	60%	73%
	L_4	$1\pi/4$	30%	53%	67%

Table 3 shows the result of running QNNs on IBM-Brisbane quantum computer. The third column of Table 3 shows the angles determined by pruning (Table 2). As the number of layers increases the similarity rate drops for the baseline QNN rapidly (fourth column). This is expected as a deeper PQC results in more quantum noise on IBM-Brisbane, causing a rapid increase in dissimilarity between the output of baseline QNN and noiseless simulations. As an example, in MNIST, the similarity rate drops from 93% in L_1 to 57% in L_4. The fifth column in Table 3 shows the similarity between noiseless circuits and pruned QNNs whereas the last column shows the results for NR-QNN. Pruning reduces quantum noise and enhances accuracy over the baseline scheme. NR-QNN improves accuracy even further and reduces the gap between NR-QNN and noiseless simulations. On average, NR-QNN increases the similarity between QNNs run on IBM-Brisbane and noiseless simulations by 17%, 19%, 23%, and 31% for L_1, L_2, L_3, and L_4, respectively.

V. CONCLUSION

We propose NR-QNN, a method to mitigate the impact of quantum noise on QNNs. NR-QNN leverages two optimization techniques to enhance the robustness of PQCs in QNNs. The first technique involves pruning unnecessary rotation gates, which reduces the number of gates and potentially lowers the depth of QNNs. This pruning process includes retraining PQCs to compensate for any potential accuracy loss from removing gates. The second technique is sensitivity-aware qubit mapping, based on the observation that output sensitivity varies across input qubits. Prior studies aimed at minimizing noise in QNNs have not fully exploited this aspect. By mapping sensitive qubits to more stable ones, the fidelity of QNNs on NISQ devices can be improved.

Using a combination of pruning and sensitivity-aware qubit mapping, we successfully ran a set of QNNs on real quantum hardware using popular datasets.

ACKNOWLEDGMENT

The authors would like to acknowledge CMC Microsystems, the Manager of the FABrIC project funded by the Government of Canada, for the provision of products and services that facilitated this research, including their member access to the IBM Quantum Hub at PINQ^2.

REFERENCES

[1] K. K. Singamaneni, G. Muhammad, and Z. Ali, "A novel multi-qubit quantum key distribution ciphertext-policy attribute-based encryption model to improve cloud security for consumers," *IEEE Trans. Consum. Electron.*, vol. 70, no. 1, pp. 1092–1101, Feb. 2024.

[2] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. 28th Annu. ACM Symp. Theory Comput. (STOC)*. New York, NY, USA: Association for Computing Machinery, 1996, pp. 212–219, doi: 10.1145/237814.237866.

[3] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature*, vol. 549, no. 7671, pp. 242–246, Sep. 2017.

[4] Y. Cao, J. Romero, J. P. Olson, M. Degroote, P. D. Johnson, M. Kieferová, I. Kivlichan, T. Menke, B. Peropadre, N. P. D. Sawaya, S. Sim, L. Veis, and A. Aspuru-Guzik, "Quantum chemistry in the age of quantum computing," *Chem. Rev.*, vol. 119, no. 19, pp. 10856–10915, Aug. 2019.

[5] T. Humble, "Consumer applications of quantum computing: A promising approach for secure computation, trusted data storage, and efficient applications," *IEEE Consum. Electron. Mag.*, vol. 7, no. 6, pp. 8–14, Nov. 2018. [Online]. Available: https://www.osti.gov/biblio/1490615

[6] R. Chatterjee, S. Mazumdar, R. S. Sherratt, R. Halder, T. Maitra, and D. Giri, "Real-time speech emotion analysis for smart home assistants," *IEEE Trans. Consum. Electron.*, vol. 67, no. 1, pp. 68–76, Feb. 2021.

[7] N. Jia, X. Liu, and Y. Sun, "Salient object detection network with selection mechanism for consumer electronics imaging systems," *IEEE Trans. Consum. Electron.*, vol. 70, no. 1, pp. 3403–3413, Feb. 2024.

[8] P. D. Paikrao, A. Mukherjee, U. Ghosh, P. Goswami, M. Novak, D. K. Jain, M. S. Al-Numay, and P. Narwade, "Data driven neural speech enhancement for smart healthcare in consumer electronics applications," *IEEE Trans. Consum. Electron.*, vol. 70, no. 2, pp. 4828–4838, May 2024.

[9] Y.-W. Lai and M.-Y. Chen, "Using natural language processing with explainable AI approach to construct a human-centric consumer application for financial climate disclosures," *IEEE Trans. Consum. Electron.*, vol. 70, no. 1, pp. 1112–1121, Feb. 2024.

[10] IBM. *Quantum Computing IBM Q*. Accessed: Jul. 10, 2024. [Online]. Available: https://www.research.ibm.com/ibm-q/

[11] *Google Quantum AI*. Accessed: Jul. 10, 2024. [Online]. Available: https://quantumai.google/

[12] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, and D. A. Buell, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, Oct. 2019.

[13] I. Q. Computing. (2023). *IBM Quantum Computing*. Accessed: Jul. 10, 2024. [Online]. Available: https://quantum-computing.ibm.com/services/resources?tab=systems

[14] P. Murali, D. C. Mckay, M. Martonosi, and A. Javadi-Abhari, "Software mitigation of crosstalk on noisy intermediate-scale quantum computers," in *Proc. 25th Int. Conf. Architectural Support Program. Lang. Operating Syst.* New York, NY, USA: Association for Computing Machinery, Mar. 2020, pp. 1001–1016, doi: 10.1145/3373376.3378477.

[15] S. Sheldon, E. Magesan, J. M. Chow, and J. M. Gambetta, "Procedure for systematically tuning up cross-talk in the cross-resonance gate," *Phys. Rev. A, Gen. Phys.*, vol. 93, no. 6, Jun. 2016, Art. no. 060302.

[16] G. Guardia, *Quantum Error Correction: Symmetric, Asymmetric, Synchronizable, and Convolutional Codes* (Quantum Science and Technology). Cham, Switzerland: Springer, 2020. [Online]. Available: https://books.google.com/books?id=bnPtDwAAQBAJ

[17] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018.

[18] A. Morello, "What would you do with 1000 qubits?" *Quantum Sci. Technol.*, vol. 3, no. 3, Jul. 2018, Art. no. 030201.

[19] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, "Variational quantum algorithms," *Nature Rev. Phys.*, vol. 3, no. 9, pp. 625–644, Aug. 2021, doi: 10.1038/s42254-021-00348-9.

[20] N. Killoran, T. R. Bromley, J. M. Arrazola, M. Schuld, N. Quesada, and S. Lloyd, "Continuous-variable quantum neural networks," *Phys. Rev. Res.*, vol. 1, no. 3, Oct. 2019, Art. no. 033063.

[21] E. Farhi and H. Neven, "Classification with quantum neural networks on near term processors," 2018, *arXiv:1802.06002*.

[22] M. Schuld, A. Bocharov, K. M. Svore, and N. Wiebe, "Circuit-centric quantum classifiers," *Phys. Rev. A, Gen. Phys.*, vol. 101, no. 3, Mar. 2020, Art. no. 032308.

[23] I. Cong, S. Choi, and M. D. Lukin, "Quantum convolutional neural networks," *Nature Phys.*, vol. 15, no. 12, pp. 1273–1278, Aug. 2019.

[24] V. Giovannetti, S. Lloyd, and L. Maccone, "Quantum random access memory," *Phys. Rev. Lett.*, vol. 100, no. 16, Apr. 2008, Art. no. 160501.

[25] H. V. Jayashree, H. Thapliyal, H. R. Arabnia, and V. K. Agrawal, "Ancilla-input and garbage-output optimized design of a reversible quantum integer multiplier," *J. Supercomput.*, vol. 72, no. 4, pp. 1477–1493, Apr. 2016.

[26] 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.

[27] M. Schuld, R. Sweke, and J. J. Meyer, "Effect of data encoding on the expressive power of variational quantum-machine-learning models," *Phys. Rev. A, Gen. Phys.*, vol. 103, no. 3, Mar. 2021, Art. no. 032430.

[28] S. Sim, P. D. Johnson, and A. Aspuru-Guzik, "Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms," *Adv. Quantum Technol.*, vol. 2, no. 12, Dec. 2019, Art. no. 1900070.

[29] A. K. K. Don, I. Khalil, and M. Atiquzzaman, "A fusion of supervised contrastive learning and variational quantum classifiers," *IEEE Trans. Consum. Electron.*, vol. 70, no. 1, pp. 770–779, Feb. 2024.

[30] K. Batra, K. M. Zorn, D. H. Foil, E. Minerali, V. O. Gawriljuk, T. R. Lane, and S. Ekins, "Quantum machine learning algorithms for drug discovery applications," *J. Chem. Inf. Model.*, vol. 61, no. 6, pp. 2641–2647, May 2021.

[31] M. Pistoia, S. F. Ahmad, A. Ajagekar, A. Buts, S. Chakrabarti, D. Herman, S. Hu, A. Jena, P. Minssen, P. Niroula, A. Rattew, Y. Sun, and R. Yalovetzky, "Quantum machine learning for finance ICCAD special session paper," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, Nov. 2021, pp. 1–9.

[32] A. Luckow, J. Klepsch, and J. Pichlmeier, "Quantum computing: Towards industry reference problems," *Digitale Welt*, vol. 5, no. 2, pp. 38–45, Apr. 2021.

[33] G. Li, Y. Ding, and Y. Xie, "Tackling the qubit mapping problem for NISQ-era quantum devices," in *Proc. 24th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Apr. 2019, pp. 1001–1014.

[34] P. Murali, J. M. Baker, A. Javadi-Abhari, F. T. Chong, and M. Martonosi, "Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers," in *Proc. 24th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Apr. 2019, pp. 1015–1029.

[35] S. S. Tannu and M. K. Qureshi, "Not all qubits are created equal: A case for variability-aware policies for NISQ-era quantum computers," in *Proc. Twenty-Fourth Int. Conf. Architectural Support Program. Lang. Operating Syst.* New York, NY, USA: Association for Computing Machinery, Apr. 2019, pp. 987–999, doi: 10.1145/3297858.3304007.

[36] Y. Ding, P. Gokhale, S. F. Lin, R. Rines, T. Propson, and F. T. Chong, "Systematic crosstalk mitigation for superconducting qubits via frequency-aware compilation," in *Proc. 53rd Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2020, pp. 201–214.

[37] S. S. Tannu and M. Qureshi, "Ensemble of diverse mappings: Improving reliability of quantum computers by orchestrating dissimilar mistakes," in *Proc. 52nd Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2019, pp. 253–265.

[38] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, "The theory of variational hybrid quantum-classical algorithms," *New J. Phys.*, vol. 18, no. 2, Feb. 2016, Art. no. 023023.

[39] R. Orús, S. Mugel, and E. Lizaso, "Quantum computing for finance: Overview and prospects," *Rev. Phys.*, vol. 4, Nov. 2019, Art. no. 100028.
[40] D. Maslov, G. W. Dueck, D. M. Miller, and C. Negrevergne, "Quantum circuit simplification and level compaction," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 3, pp. 436–444, Mar. 2008.
[41] D. Maslov, S. M. Falconer, and M. Mosca, "Quantum circuit placement," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 4, pp. 752–763, Apr. 2008.
[42] R. Barends, C. Quintana, A. Petukhov, Y. Chen, D. Kafri, K. Kechedzhi, R. Collins, O. Naaman, S. Boixo, and F. Arute, "Diabatic gates for frequency-tunable superconducting qubits," *Phys. Rev. Lett.*, vol. 123, no. 21, Nov. 2019, Art. no. 210501.
[43] H. Wang, Y. Ding, J. Gu, Y. Lin, D. Z. Pan, F. T. Chong, and S. Han, "QuantumNAS: Noise-adaptive search for robust quantum circuits," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Apr. 2022, pp. 692–708.
[44] H. Wang, Z. Li, J. Gu, Y. Ding, D. Z. Pan, and S. Han, "QOC: Quantum on-chip training with parameter shift and gradient pruning," in *Proc. 59th ACM/IEEE Design Autom. Conf.* New York, NY, USA: Association for Computing Machinery, Jul. 2022, pp. 655–660, doi: 10.1145/3489517.3530495.
[45] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, and J. I. Latorre, "Data re-uploading for a universal quantum classifier," *Quantum*, vol. 4, p. 226, Feb. 2020.
[46] Y. Levine, D. Yakira, N. Cohen, and A. Shashua, "Deep learning and quantum entanglement: Fundamental connections with implications to network design," 2017, *arXiv:1704.01552*.
[47] M. Alam and S. Ghosh, "QNet: A scalable and noise-resilient quantum neural network architecture for noisy intermediate-scale quantum computers," *Frontiers Phys.*, vol. 9, Jan. 2022, Art. no. 755139.
[48] X.-Z. Luo, J.-G. Liu, P. Zhang, and L. Wang, "Yao.Jl: Extensible, efficient framework for quantum algorithm design," *Quantum*, vol. 4, p. 341, Oct. 2020.
[49] L. Banchi and G. E. Crooks, "Measuring analytic gradients of general quantum evolution with the stochastic parameter shift rule," *Quantum*, vol. 5, p. 386, Jan. 2021.
[50] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, S. Ahmed, V. Ajith, M. S. Alam, G. Alonso-Linaje, B. AkashNarayanan, and A. Asadi, "PennyLane: Automatic differentiation of hybrid quantum-classical computations," 2018, *arXiv:1811.04968*.
[51] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, no. 721. Red Hook, NY, USA: Curran Associates, 2019, pp. 8026–8037.
[52] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, Dec. 2014, pp. 3320–3328.
[53] J. Rauschecker, "Neuronal mechanisms of developmental plasticity in the cat's visual system," *Hum. Neurobiol.*, vol. 3, no. 2, pp. 109–114, 1984.
[54] M. H. Devoret and R. J. Schoelkopf, "Superconducting circuits for quantum information: An outlook," *Science*, vol. 339, no. 6124, pp. 1169–1174, Mar. 2013.
[55] P. Klimov, J. Kelly, Z. Chen, M. Neeley, A. Megrant, B. Burkett, R. Barends, K. Arya, B. Chiaro, and Y. Chen, "Fluctuations of energy-relaxation times in superconducting qubits," *Phys. Rev. Lett.*, vol. 121, no. 9, Aug. 2018, Art. no. 090502.
[56] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon, S. Martiel, P. D. Nation, L. S. Bishop, A. W. Cross, B. R. Johnson, and J. M. Gambetta, "Quantum computing with Qiskit," 2024, *arXiv:2405.08810*.
[57] J. C. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, no. 61, pp. 2121–2159, Feb. 2011.
[58] J. A. Montañez-Barrera, M. R. von Spakovsky, C. E. D. Ascencio, and S. Cano-Andrade, "Decoherence predictions in a superconducting quantum processor using the steepest-entropy-ascent quantum thermodynamics framework," *Phys. Rev. A, Gen. Phys.*, vol. 106, no. 3, Sep. 2022, Art. no. 032426, doi: 10.1103/physreva.106.032426.

SOHRAB SAJADIMANESH received the M.Sc. degree in computer architecture from the University of Tehran, Iran, in 2015. He is currently pursuing the Ph.D. degree in electrical and computer engineering with Lakehead University. His research interests include quantum computing, quantum machine learning, GPGPUs, and energy-efficient computing.

HANIEH AGHAEE RAD received the B.Sc. degree in physics and materials science and engineering from the Sharif University of Technology, Iran, and the M.Sc. degree from The University of British Columbia, Canada. She is currently a Quantum Staff Scientist at CMC Microsystems. Her research interests include quantum computing.

JEAN PAUL LATYR FAYE received the master's degree in computer science from the University of Sherbrooke, Quebec City, Canada, and the Ph.D. degree in physics. In physics, his research interests include quantum mechanics phenomena, including spin magnetism, superconductivity, and quantum computing. In computer science, he specializes in machine learning.

EHSAN ATOOFIAN received the B.Sc. and M.Sc. degrees in computer engineering from the University of Tehran, Tehran, Iran, in 2000 and 2003, respectively, and the Ph.D. degree from the University of Victoria, Victoria, Canada, in 2008. He is currently an Associate Professor with the Electrical and Computer Engineering Department, Lakehead University, Thunder Bay, Canada. His research has been funded by the Natural Sciences and Engineering Research Council of Canada and MITACS. His research interests include computer architecture, quantum computing, and efficient deep-learning computing.