ROQNN: NOISE-AWARE TRAINING FOR ROBUST QUANTUM NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Quantum Neural Network (QNN) is a promising application towards quantum advantage on near-term quantum hardware. However, due to the large quantum noises (errors), the performance of QNN models has a severe degradation on real quantum devices. For example, the accuracy gap between noise-free simulation and noisy results on IBMQ-Yorktown for MNIST-4 classification is over 60%. Existing noise mitigation methods are *general* ones without leveraging unique characteristics of QNN and are only applicable to inference; on the other hand, existing QNN work does not consider noise effect. To this end, we present RoQNN, a QNN-specific framework to perform noise-aware optimizations in both training and inference stages to improve robustness. We analytically deduct and experimentally observe that the effect of quantum noise to QNN measurement outcome is a linear map from noise-free outcome with a scaling and a shift factor. Motivated by that, we propose *post-measurement normalization* to mitigate the feature distribution differences between noise-free and noisy scenarios. Furthermore, to improve the robustness against noise, we propose *noise injection* to the training process by inserting quantum error gates to QNN according to realistic noise models of quantum hardware. Finally, *post-measurement quantization* is introduced to quantize the measurement outcomes to discrete values, achieving the denoising effect. Extensive experiments on 8 classification tasks using 6 quantum devices demonstrate that RoQNN improves accuracy by up to 43% and 22% on average, and achieves over 94% 2-class, 80% 4-class, and 34% 10-class classification accuracy on real quantum computers. We also open-source our PyTorch library for construction and noise-aware training of QNN at this link.

1 INTRODUCTION

Quantum Computing (QC) is a new computational paradigm that can be exponentially faster than classical counterparts in various domains such as cryptography (Shor, 1999), database search (Grover, 1996), and chemistry (Kandala et al., 2017; Peruzzo et al., 2014; Cao et al., 2019). Quantum Machine Learning (QML) aims to leverage QC techniques to solve machine learning tasks and achieve much higher efficiency. Among various QML approaches, Quantum Neural Network (QNN) is a popular candidate in which a network of parameterized quantum gates are constructed and trained to embed data and perform certain ML tasks on a quantum computer, similar to the training and inference of classical neural networks.

Currently we are in the Noisy Intermediate Scale Quantum (NISQ) stage, in which quantum operations suffer from a high error rate of 10^{-2} to 10^{-3}, much higher than CPUs/GPUs (10^{-6} FIT). The quantum errors unfortunately introduces detrimental influence on QNN accuracy. Figure 1 shows the single-qubit gate error rates and the measured accuracy of classification tasks on different hardware. Three key observations are: (1) Quantum error rates (10^{-3}) are much larger than classical CMOS devices' error rates (10^{-6} failure per 10^9 device hours (Rao et al., 2007)). (2) Accuracy on real hardware is significantly degraded (up to 64%) compared with noise-free simulation. (3) The same QNN on different hardware has distinct accuracy due to different gate error rates. IBMQ-Yorktown has a five times larger error rate than IBMQ-Santiago, and higher error causes lower accuracy.

Researchers have proposed noise mitigation techniques (Endo et al., 2021; Li & Benjamin, 2017; Temme et al., 2017; Endo et al., 2018) to reduce the noise impact. However, they are *general*

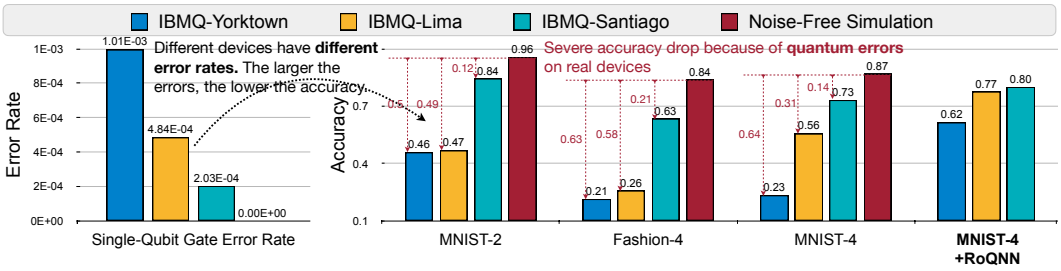

Figure 1: *Left:* Current quantum hardware has much larger error rates (around 10^{-3}) than classical CPUs/GPUs. *Right:* Due to the errors, QNN models suffer from severe accuracy drops. Different devices have various error magnitudes, leading to distinct accuracy. These motivate RoQNN, a *hardware-specific noise-aware* QNN training approach to improve robustness and accuracy.

methods without considering the unique characteristics of QNN, and can only be applied to QNN inference stage. On the other hand, existing QNN work (Biamonte et al., 2017; Harrow et al., 2009; Farhi et al., 2014; Lloyd et al., 2013; Rebentrost et al., 2014; Bausch, 2020; Jiang et al., 2021) does not consider the noise impact. This paper proposes a QNN-specific noise mitigation framework called RoQNN that optimizes QNN robustness in *both training and inference* stages, boosts the *intrinsic robustness* of QNN parameters, and improves accuracy on *real quantum machines*.

RoQNN comprises a three-stage pipeline. First, *post-measurement normalization* normalizes the measurement outcomes on each quantum bit (qubit) across data samples, thus removing the quantum error-induced distribution shift. Furthermore, we inject noise to the QNN training process by performing *error gate insertion*. The error gate types and probabilities are obtained from hardware-specific realistic quantum noise models provided by QC vendors. During training, we iteratively sample error gates, insert them to QNN, and updates weights. Finally, *post-measurement quantization* is further proposed to reduce the precision of measurement outcomes from each qubit and achieve a denoising effect.

Extensive experiments on 8 ML tasks with 5 different design spaces on 6 quantum devices show that RoQNN can improve accuracy by up to 42%, 43%, 23% for 2-class, 4-class and 10-class classification tasks and successfully demonstrates over 94%, 80% and 34% accuracy for 2-, 4-, and 10-classifications with *pure quantum parameters* on real quantum hardware. The PyTorch library we developed for construction and noise-aware training of QNN is open-sourced at this link. It is an easy-to-use infrastructure to query noise models from QC providers such as IBMQ, extract noise information, perform training on CPU/GPU and finally deploy on real QC (Appendix A.3).

2 Background and Related Work

QML and QNN. The quantum basics and quantum noise are introduced in Appendix A.1. Quantum machine learning (Biamonte et al., 2017) explores performing ML tasks on quantum devices. The path to *quantum advantage* on QML is typically provided by the quantum circuit's ability to generate and estimate highly complex kernels (Havlíček et al., 2019), which would otherwise be intractable to compute with conventional computers. They have been shown to have potential speed-up over classical counterparts in various tasks, including metric learning (Lloyd et al., 2020), data analysis (Lloyd et al., 2016), and principal component analysis (Lloyd et al., 2014). Quantum Neural Networks is one type of QML models using variational quantum circuits with trainable parameters to accomplish feature encoding of input data and perform complex-valued linear transformations thereafter. Various theoretical formulations for QNN have been proposed, e.g., quantum classifier (Farhi & Neven, 2018), quantum convolution (Henderson et al., 2020), and quantum Boltzmann machine (Amin et al., 2018), etc. Most are exploratory and rely on classical simulation of small quantum systems (Farhi & Neven, 2018).

Quantum Error Mitigation As the error forms the bottleneck of the quantum area. Researchers have developed various error mitigation techniques. Extrapolation methods (Temme et al., 2017; Li & Benjamin, 2017) perform multiple measurements of a quantum circuit under different error rates and then extrapolate the ideal measurement outcomes when there is no noise. Quasi-probability (Temme et al., 2017; Huo & Li, 2021) probabilistically inserts X, Y, Z gates to a quan-

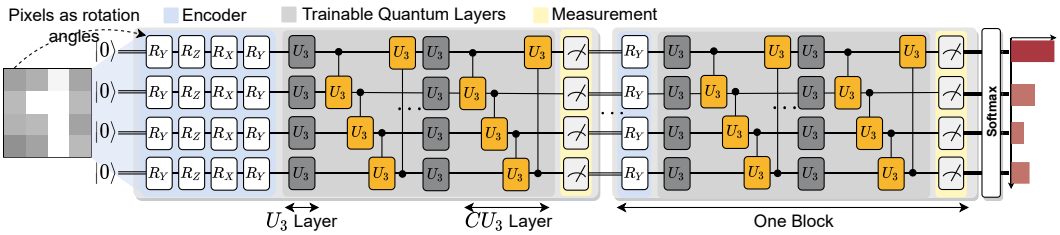

Figure 2: Quantum Neural Networks Architecture. QNN has multiple blocks, each contains an encoder to encode classical values to quantum domain; quantum layers with trainable weights; and a measurement layer that obtains a classical value from each qubit.

tum circuit and then sum them together to cancel out the noise effects. Other methods such as quantum subspace expansion (McClean et al., 2017) and learning-based mitigation (Strikis et al., 2020; Czarnik et al., 2020) are also proposed.

RoQNN is fundamentally different from existing methods: (i) Prior work focuses on low-level numerical correction in inference only; RoQNN embraces more optimization freedom in both *training and inference*. It improves the intrinsic robustness and statistical fidelity of *QNN parameters*. (ii) QNN has a good built-in error-tolerance which motivates RoQNN's post-measurement quantization to reduce the numerical precision of intermediate results while preserving accuracy. (iii) RoQNN has a very small overhead (less than 2%), while prior work introduces high measurements, circuit complexity cost, etc. We also show that existing methods such as extrapolation is orthogonal to RoQNN and can be combined together in Section 4.

Quantization and Noise Injection of Classical NN. To improve NN efficiency, extensive work has been explored to trim down redundant bit representation in NN weights and activations (Han et al., 2015; Zhu et al., 2016; Jacob et al., 2018; Wang et al., 2019; 2020; Lin et al., 2017). Though low-precision quantization limits the model capacity, it can improve the generalization and robustness (Lin et al., 2019). An intuitive explanation is that quantization corrects errors by value clamping, thus avoiding cascaded error accumulation. Moreover, by sparsifying the parameter space, quantization reduces the NN complexity as a regularization mechanism that mitigates potential overfitting issues. Similarly, injecting noises into neural network training is demonstrated to help obtain a smoothed loss landscape for better generalization (Matsuoka, 1992; He et al., 2019; Zur et al., 2009; Seltzer et al., 2013). By emulating the real noisy environment when deploying NNs, noise-injection-based training significantly boosts the noise-robustness, especially for emerging applications and computing platforms (Gu et al., 2020; Xu et al., 2014).

3 Noise-Aware QNN Training

Figure 2 shows the QNN architecture. The inputs are classical data such as image pixels, and the outputs are classification results. The QNN consists of multiple blocks. Each has three components: encoder encodes the classical values to quantum states with rotation gates such as `RY`; trainable quantum layers contain parameterized gates that can be trained to perform certain ML tasks; measurement part measures each qubit and obtains a classical value. The measurement outcomes of one block are passed to the next block. For the MNIST-4 example in Figure 2, the first encoder takes the pixels of the down-sampled 4×4 image as rotation angles θ of 16 rotation gates. The measurement results of the last block are passed through a Softmax to output classification probabilities.

The overview of RoQNN is shown in Figure 3. First, we normalize the measurement outcome distribution of each qubit across input samples during both training and inference to compensate for information loss. Then, we leverage a realistic quantum noise model of quantum devices to insert noise into the training procedure and boost the error resilience. Finally, we quantize the measurement outcomes to discrete values to correct quantum noise-induced errors.

3.1 Post-Measurement Normalization

Measurement outcome shift due to quantum noises. Before delving into the noise mitigation techniques, we first show analytically how quantum noises influence the QNN block output. The measurement outcomes of the QNN are sensitive to both the input parameters and any perturbations

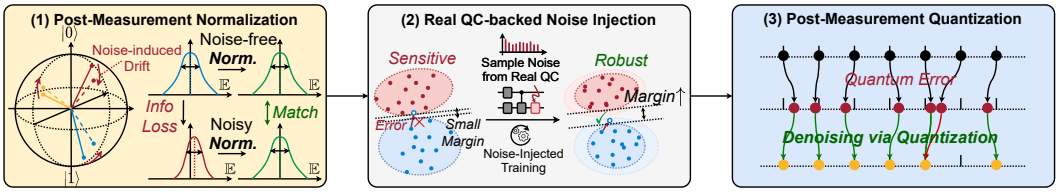

Figure 3: RoQNN Overview. (1) Post-measurement normalization matches the distribution of measurement results between noise-free simulation and real hardware deployment. (2) Based on realistic noise models, noise-injection inserts *quantum error gates* to the training process to increase the classification margin between classes. (3) Measurement outcomes are further quantized for denoising.

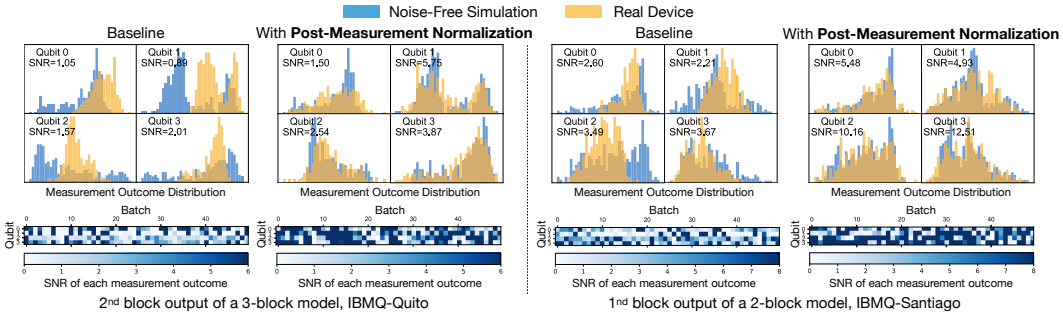

Figure 4: Post-measurement normalization reduces the distribution mismatch between noise-free simulation and noisy results on real hardware, thus improving the Signal-to-Noise Ratio (SNR).

by some noisy quantum process. This section provides insights on such noisy transformations and discusses their impacts on QNN inference.

Theorem 3.1. (informal version). *The measurement outcome y of a quantum neural network for the training input data x is transformed by the quantum noise that the system undergoes with a linear map $f(y_x) = \gamma y_x + \beta_x$, where the translation β_x depends on the input x and quantum noises, while scaling factor γ is input independent.*

We refer to Appendix Section A.2.2 for background and a complete proof. The main theoretical contribution of this theorem equips our proposed normalization methodology with robustness guarantees. Most importantly, we observe that the changes in measurement results can often be compensated by proper post-measurement normalization across input batches. For simplicity, we restrict our analysis on Z-basis single-qubit measurement outcome y. Similar analytical results for multi-qubit general-basis measurement will follow if we apply the same analysis qubit by qubit. Theorem 3.1 is most powerful when applied on a small batch of input data $\mathbf{x} = \{x_1, \ldots, x_m\}$ where each x_i is a set of classical input values for the encoder of the QNN and m is the size of the batch. In an ideal noiseless scenario, the QNN model outputs measurement result y_i for each input x_i. For a noisy QNN, the measurement result undergoes a composition of two transformations: (1) a constant scaling by γ; (2) a input-specific shift by β_i, i.e., $f(y_i) = \gamma y_i + \beta_i$. In the realistic noise regime, the scaling constant $\gamma \in [-1, 1]$. However, for small noises, γ is close to 1, and β_i is close to 0. Therefore, the distribution of noisy measurement results undergoes a constant scaling by $\gamma \leq 1$ and a small shift by each β_i. In the small-batch regime when $\boldsymbol{\beta} = \{\beta_1, \ldots, \beta_m\}$ has small variance, the distribution is shifted by its mean $\beta = \mathbb{E}[\boldsymbol{\beta}]$. Thus $f(y_i) \approx \gamma y_i + \beta$.

Post-measurement normalization. Based on the analysis above, we propose *post-measurement normalization* to offset the distribution scaling and shift. For each qubit, we collect its measurement results on a batch of input samples, compute mean and std, then make the distribution of each qubit across the batch *zero-centered* and of *unit variance*. This is performed during both training and inference. During training, for a batch of measurement results: $\boldsymbol{y} = \{y_1, \ldots, y_m\}$, the normalized results are $\hat{y}_i = (y_i - \mathbb{E}[\boldsymbol{y}])/\sqrt{\mathrm{Var}(\boldsymbol{y})}$. For noisy inference, $\widehat{f(y_i)} = (f(y_i) - \mathbb{E}[f(\boldsymbol{y})])/\sqrt{\mathrm{Var}(f(\boldsymbol{y}))} = ((\gamma y_i + \beta) - (\gamma \mathbb{E}[\boldsymbol{y}] + \beta))/\sqrt{\gamma^2 \mathrm{Var}(\boldsymbol{y})} = \hat{y}_i$. Thus the error can be corrected.

Empirical results confirm the analysis above. Figure 4 compares the noise-free measurement result distribution of 4 qubits (blue) with their noisy counterparts (yellow) for MNIST-4 on two devices.

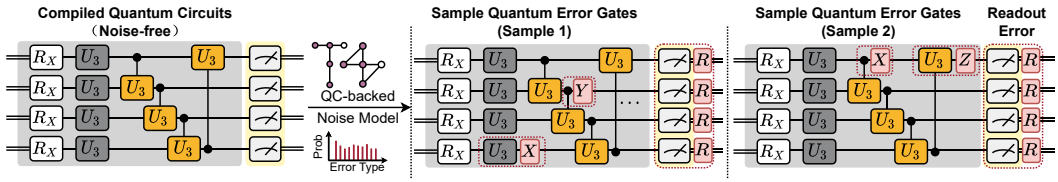

Figure 5: Noise injection via error gate insertion. X, Y, Z are sampled Pauli error gates. R is the injected readout error. Probabilities for gate insertion are obtained from real device noise models.

Qualitatively, we can clearly observe that the post-measurement normalization reduces the mismatch between two distributions. Quantitatively, we adopt signal-to-noise ratio, $SNR = \|\boldsymbol{A}\|_2^2 / \|\boldsymbol{A} - \widetilde{\boldsymbol{A}}\|_2^2$, the inverse of relative matrix distance (RMD), as the metric. The SNR on each qubit and each individual measurement outcome is clearly improved. Though similar, it is different from Batch Normalization (Ioffe & Szegedy, 2015) as the testing batch uses its own statistics instead of that from training, and there is no trainable affine parameter.

3.2 Quantum Noise Injection

Although the normalization above mitigates error impacts, we can still observe small discrepancies on each individual measurement outcome, which degrade the accuracy. Therefore, to make the QNN model robust to those errors, we propose noise injection to the training process.

Quantum error gate insertion. As introduced in Section 2, different quantum errors can be approximated by Pauli errors via Pauli Twirling. The effect of Pauli errors is the random insertion of Pauli X, Y, and Z gates to the model with a probability distribution $\mathcal{E}$. How to compute $\mathcal{E}$ is out of the scope of this work. But fortunately, we can directly obtain it from the realistic device noise model provided by quantum hardware manufacturers such as IBMQ. The noise model specifies the probability $\mathcal{E}$ for different gates on each qubit. For single-qubit gates, the error gates are inserted *after* the original gate. For two-qubit gates, error gates are inserted after the gate on *one or both* qubits. For example, the SX gate on qubit 1 on IBMQ-Yorktown device has $\mathcal{E}$ as {X: 0.00096, Y: 0.00096, Z: 0.00096, None: 0.99712}. When 'None' is sampled, we will not insert any gate. The same gate on different qubits or different hardware will have up $10\times$ probability difference. As in Figure 5, during training, for each QNN gate, we sample error gates based on $\mathcal{E}$ and insert it after the original gate. A new set of error gates is sampled for each training step. In reality, the QNN is compiled to the basis gate set of the quantum hardware (e.g., X, CNOT, RZ, CNOT, and ID) before performing gate insertion and training. We will also scale the probability distribution by a constant *noise factor* T and scale the X, Y, Z probability by T during sampling. T factor explores the trade-off between adequate noise injection and training stability. Typical T values are in the range of [0.5, 1.5]. The gate insertion overhead is typically less than 2%.

Readout noise injection. Obtaining classical values from qubits is referred as readout/measurement, which is also error-prone. The realistic noise model provides the statistical readout error in the form of a $2\times$ 2 matrix for each qubit. For example, the qubit 0 of IBMQ-Santiago has readout error matrix [[0.984, 0.016], [0.022, 0.978]] which means the probability of measuring a $|0\rangle$ as 0 is 0.984 and as 1 is 0.016. We emulate the readout error effect during training by changing the measurement outcome. For instance, originally $P(0) = 0.3, P(1) = 0.7$, the noise injected version will be $P'(0) = 0.3 \times 0.984 + 0.7 \times 0.022 = 0.31, P'(1) = 0.7 \times 0.978 + 0.3 \times 0.016 = 0.69$.

Direct perturbation. Besides gate insertion, we also experimented with directly perturbing measurement outcomes or rotation angles as noise sources. For outcome perturbation, with benchmarking samples from the validation set, we obtain the error Err distribution between the noise-free and noisy measurement results and compute the mean μ_{Err} and std σ_{Err}. During training, we directly add noise with Gaussian distribution $\mathcal{N}(\mu_{Err}, \sigma_{Err}^2)$ to the normalized measurement outcomes. Similarly, for rotation angle perturbation, we add Gaussian noise to the angles of all rotation gates in QNN and make the effect of rotation angle Gaussian noise on measurement outcomes similar to real QC noise. We show in Section 4 that the gate insertion method is better than direct perturbations.

3.3 Post-Measurement Quantization

Finally, we propose post-measurement quantization on the normalized results to further denoise the measurement outcomes. We first clip the outcomes to $[p_{min}, p_{max}]$, where p are pre-defined

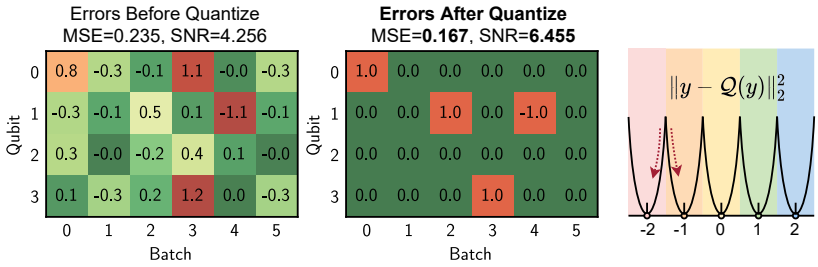

Figure 6: *Left*: Error maps before and after post-measurement quantization. Most errors can be corrected. *Right*: 5-level quantization buckets with a quadratic penalty loss to encourage measurement outcomes to be near to the centroids.

thresholds, and then perform uniform quantization. The quantized values are later passed to the next block's encoder. Figure 6 shows one real example from Fashion-4 on IBMQ-Santiago with five quantization levels and $p_{min} = -2, p_{max} = 2$. The left/middle matrices show the error maps between noise-free and noisy outcomes before/after quantization. Most errors can be corrected back to zero with few exceptions of being quantized to a wrong centroid. The MSE is reduced from 0.235 to 0.167, and the SNR is increased from 4.256 to 6.455. We also add a loss term $||y - Q(y)||_2^2$ to the training loss, as shown on the right side, to encourage outcomes to be near to the quantization centroids to improve error tolerance and reduce the chance of being quantized to a wrong centroid. Besides improving robustness, quantization also brings an additional benefit: the control complexity of rotation gates using those quantized values can be largely reduced.

4 Experiments

4.1 Experiment Setups

Datasets. We conduct experiments on 8 classification tasks including MNIST (Lecun et al., 1998) 10-class, 4-class (`0, 1, 2, 3`). and 2-class (`3, 6`); Vowel (Deterding, 1989) 4-class (`hid, hId, had, hOd`); Fashion (Xiao et al., 2017) 10-class, 4-class (`t-shirt/top, trouser, pullover, dress`), and 2-class (`dress, shirt`), and CIFAR (Krizhevsky et al.) 2-class (`frog, ship`). MNIST, Fashion, and CIFAR use 95% images in 'train' split as training set and 5% as the validation set. Due to the limited real QC resources, we use the first 300 images of 'test' split as test set. Vowel-4 dataset (990 samples) is separated to train:validation:test = 6:1:3 and test with the whole test set. MNIST and Fashion images are center-cropped to 24×24; and then down-sample to 4×4 for 2- and 4-class, and 6×6 for 10-class; CIFAR images are converted to grayscale, center-cropped to 28×28, and down-sampled to 4×4. All down-samplings are performed with average pooling. For vowel-4, we perform feature principal component analysis (PCA) and take 10 most significant dimensions.

QNN models. The first quantum block's encoder embeds images and vowel features. For 4×4 images, we use 4 qubits and 4 layers with 4 `RY`, 4 `RX`, 4 `RZ`, and 4 `RY` gates in each layer, respectively. For 6×6 images, 10 qubits and 4 layers are used with 10 `RY`, 10 `RX`, 10 `RZ`, and 6 `RY` gates in each layer, respectively. 10 vowel features, uses 4 qubits and 3 layers with 4 `RY`, 4 `RX`, and 2 `RZ` gates on each layer for encoding. For trainable quantum layers, we use `U3` and `CU3` layers interleaved as in Figure 2 except for Table 2. For measurement, we measure the expectation values on Pauli-`Z` basis and obtain a value [-1, 1] from each qubit. The measurement outcome goes through post-measurement normalization and quantization and is used as rotation angles for `RY` gates in the next block's encoder. After the last block, for two-classifications, we sum the qubit 0 and 1, 2 and 3 measurement outcomes, respectively, and use Softmax to get probabilities. For 4 and 10-class, Softmax is directly applied to measurement outcomes.

The number of parameters can be computed as $N_{Block} \times N_{params\_per\_block}$. For instance, for QNN using 4 qubits, 1 `U3`, and 1 `CU3` layer in each block, since one `U3` and `CU3` gates both have 3 parameters, $N_{params\_per\_block} = 3\times4\times1\times2 = 24$. A model with 5 blocks has 120 parameters. We implement a library for construction and noise-aware training of QNN models in PyTorch (Paszke et al., 2019), and all model training in this work is performed with it. For baselines and RoQNN, we use Adam optimizer with a linear learning rate warm-up from 0 to 5e-3 in the first 30 epochs then cosine decay and weight decay $\lambda = 1e-4$. We train 200 epochs with batch size 256 for image

Table 1: Post-measurement normalization improves accuracy and SNR.

Quantum Devices ↓	QNN Models →	2 Blocks				3 Blocks				4 Blocks			
		×2 Layers		×8 Layers		×2 Layers		×4 Layers		×2 Layers		×4 Layers	
		Acc.	SNR	Acc.	SNR	Acc.	SNR	Acc.	SNR	Acc.	SNR	Acc.	SNR
Santiago	Baseline	0.61	6.15	0.52	1.79	0.71	**5.47**	0.61	5.32	0.57	6.96	0.62	4.20
	+Norm	**0.66**	**15.69**	**0.79**	**4.85**	**0.71**	4.80	**0.80**	**8.45**	**0.70**	**11.36**	**0.68**	**6.55**
Quito	Baseline	0.58	6.64	0.35	1.43	**0.72**	2.55	0.66	1.85	0.60	3.98	0.29	1.73
	+Norm	**0.66**	**13.92**	**0.71**	**2.98**	0.69	**7.38**	**0.76**	**7.15**	**0.74**	**12.26**	**0.72**	**4.54**
Athens	Baseline	0.59	8.91	0.60	2.14	0.63	**8.26**	0.62	3.76	0.63	9.52	0.55	3.54
	+Norm	**0.64**	**20.27**	**0.78**	**3.47**	**0.68**	6.50	**0.78**	**5.91**	**0.74**	**14.07**	**0.69**	**6.09**

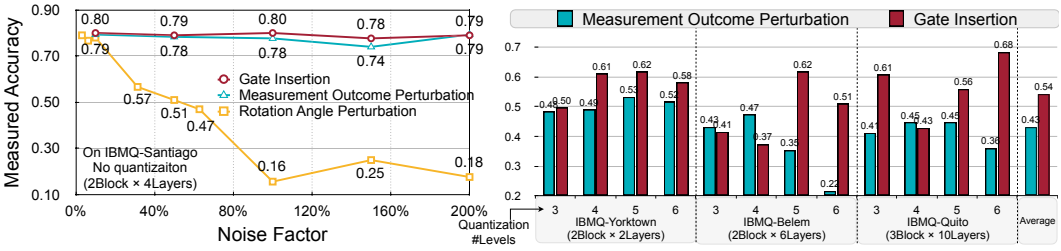

Figure 7: Ablation on different noise injection methods. *Left:* Without quantization, gate insertion and measurement perturbation performs similar, both better than rotation angle perturbation. *Right:* With quantization, gate insertion is better as perturbation effect can be canceled by quantization.

classification and 4 for vowel. For 4-qubit QNN models, the overall training time is typically less than 2 hours on an Nvidia TITAN RTX 2080 ti GPU machine.

Quantum hardware and compiler configurations.. We use IBMQ quantum computers via Qiskit (IBM) APIs. We study 6 devices, with #qubits from 5 to 15 and Quantum Volume from 8 to 32. We also employ Qiskit for compilation. The optimization level is set to 2 for all experiments, except for Table 6. All experiments run 8192 shots. The noise models we used are off-the-shelf ones updated by IBMQ team with noise characterization techniques such as randomized benchmarking.

4.2 Experimental Results

Ablation on post-measurement normalization Table 1 compares the accuracy and signal-to-noise ratio (SNR) before and after post-measurement normalization on MNIST-4. We study 6 different QNN architectures and evaluate on 3 devices. The normalization can significantly increase SNR, thus improving accuracy with rare exceptions on 3Block × 2Layer models.

Ablation on different noise injection methods. Figure 7 compares different noise injection methods. Gaussian noise statistics for perturbations are obtained from error benchmarking. The left side shows accuracy without quantization. With different noise factors T, the gate insertion and measurement outcome perturbation have similar accuracy, both better than rotation angle perturbation. A possible explanation is that the rotation angle perturbation does not consider non-rotation gates such as `X` and `SX`. The right side further investigates the first two methods' performance with quantization. We set noise factor $T = 0.5$ and alter quantization levels. Gate insertion outperforms perturbation by 11% on average on 3 different devices and QNN models. The reason is: directly added perturbation on measurement outcomes can be easily canceled by quantization, and thus it is harder for noise injection to take effect.

Main results. From ablations above, we decide to apply post-measurement normalization to RoQNN, and use gate insertion to inject noise. We experiment with four different QNN architectures on 8 tasks running on 5 quantum devices to demonstrate RoQNN's effectiveness. For each benchmark, we experiment with noise factor $T = \{0.1, 0.5, 1, 1.5\}$ and quantization level among $\{3, 4, 5, 6\}$ and select one out of 16 combinations with the lowest loss on the *validation set* and test on the *test set*. Normalization and quantization are not applied to the last block's measurement outcomes as they are directly used for classification. As in Figure 8, RoQNN consistently achieves the highest accuracy on 26 benchmarks. The third bars of Athens are unavailable due to the machine's retirement. On average, normalization, noise injection and quantization improve accuracy

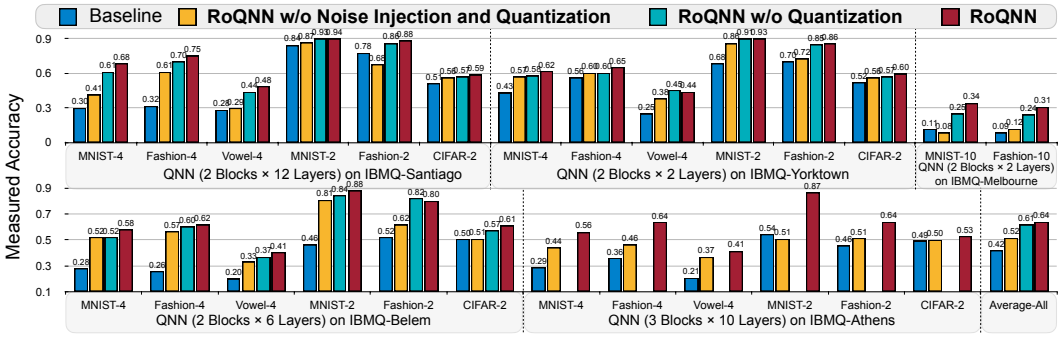

Figure 8: RoQNN consistently achieves the highest accuracy, with on average 22% improvements.

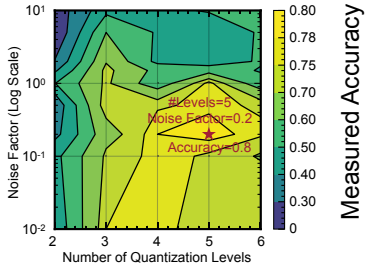

Figure 9: Acc. contours of quant. levels & noise factors.

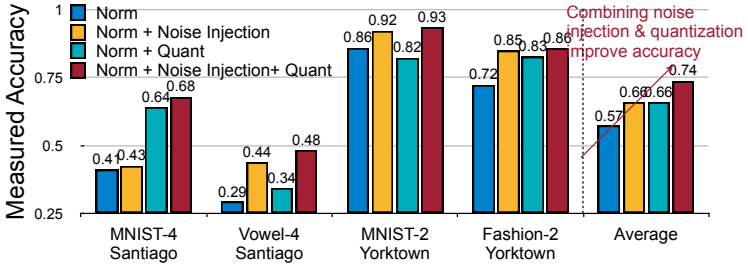

Figure 10: Ablation of applying noise injection and quantization individually or jointly. Combining two together brings the best accuracy.

by 10%, 9%, and 3%, respectively. Another observation is that a larger model does not necessarily have higher accuracy. For example, Athens has a smaller single-qubit gate error rate (2.9×10^{-4}) than Yorktown (1.0×10^{-3}), and Athens' QNN model is 7.5× larger than Yorktown with higher noise-free accuracy. However, because of more gate errors introduced by the larger model, the real accuracy is often lower. The detailed hyperparameters are in Appendix A.5.

Performance on different design spaces. In Table 2, we evaluate RoQNN on different QNN design spaces. Specifically, the trainable quantum layers in one block of 'ZZ+RY' (Lloyd et al., 2020) space contains one layer of ZZ gate, with ring connections, and one RY layer. 'RXYZ' (McClean et al., 2018) space has five layers: $\sqrt{\text{H}}$, RX, RY, RZ, and CZ. 'ZX+XX' (Farhi & Neven, 2018) space has two layers: ZX and XX. 'RXYZ+U1+CU3' (Henderson et al., 2020) space, according to their random circuit basis gate set, has 11 layers in the order of RX, S, CNOT, RY, T, SWAP, RZ, H, $\sqrt{\text{SWAP}}$, U1 and CU3. We conduct experiments on MNIST-4 and Fashion-2 on 2 devices. In 13 settings out of 16, RoQNN can improve the accuracy of baseline designs. Thus, RoQNN is a general technique agnostic to QNN model size and design space.

Table 2: Accuracy on different design spaces.

Design Space	MNIST-4		Fashion-2	
	Yorktown	Santiago	Yorktown	Santiago
'ZZ+RY'	**0.43**	0.57	0.80	**0.91**
+RoQNN	0.34	**0.60**	**0.83**	0.86
'RXYZ'	0.57	0.61	0.88	0.89
+RoQNN	**0.61**	**0.70**	**0.92**	**0.91**
'ZX+XX'	0.29	0.51	**0.52**	0.61
+RoQNN	**0.38**	**0.64**	**0.52**	**0.89**
'RXYZ+U1+CU3'	0.28	**0.25**	0.48	0.50
+RoQNN	**0.33**	0.21	**0.53**	**0.52**

Noise factor and post-measurement quantization level analysis. We visualize the QNN accuracy contours on Fashion-4 on IBMQ-Athens with different noise factors and quantization levels. The best accuracy occurs for factor 0.2 and 5 levels. Horizontal-wise, the accuracy first goes up and then goes down. This is because too few quantization levels hurt the QNN model capacity; too many levels cannot bring sufficient denoising effect. Vertical-wise, the accuracy also goes up and then down. Reason: when the noise is too small, the noise-injection effect is weak, thus cannot improve the model robustness; while too large noise makes the training process unstable and hurts accuracy.

Breakdown of accuracy gain. Figure 10 shows the performance of only applying noise-injection, only applying quantization, and both. Using two techniques individually can both improve accuracy by 9%. Combining two techniques delivers better performance with a 17% accuracy gain. This indicates the benefits of synergistically applying three techniques in RoQNN.

Table 3: Scalable noise-aware training.

Machine	Bogota	Santiago	Lima
Noise-unaware	0.74	0.97	0.87
RoQNN	**0.79**	**0.99**	**0.90**

Table 4: Compatible with existing noise mitigation.

Method	MNIST-4	Fashion-4
Normalization only	0.78	0.81
Normalization + Extrapolation	**0.81**	**0.83**

Visualization of QNN extracted features. MNIST-2 classification result is determined by which feature is larger between the two: feature one is the sum of measurement outcomes of qubit 0 and 1; feature 2 is that of qubit 2 and 3. We visualize the two features obtained from experiments on Belem in a 2-D plane as in Figure 11. The blue dash line is the classification boundary. The circles/stars are samples of digit '3' and '6'. All the baseline points (yellow) huddled together, and all digit '3' samples are misclassified. With normalization (green), the distribution is significantly expanded, and the majority of '3' is correctly classified. Finally, after noise injection (red), the margin between the two classes is further enlarged, and the samples are farther away from the classification boundary, thus becoming more robust.

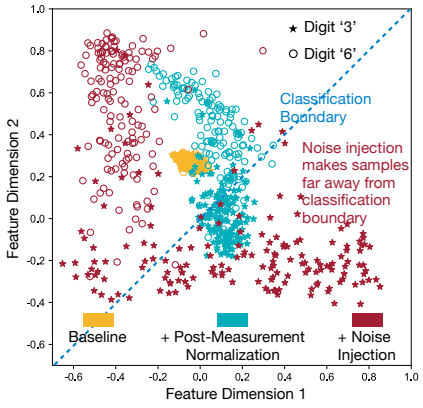

Figure 11: Feature visualization.

Scalability. When classical simulation is infeasible, we can move the the noise-injected training to real QC using techniques such as *parameter shift* (Crooks, 2019). In this case, the training cost is *linearly* scaled with qubit number. Post-measurement normalization and quantization are also *linearly* scalable because they are performed on the measurement outcomes. Gradients obtained with real QC are naturally noise-aware because they are directly influenced by quantum noise. To demonstrate the practicality, we train a 2-class task with two numbers as input features (Jiang et al., 2021) (Table 3). The QNN has 2 blocks; each with 2 RY and a CNOT gates. The noise-unaware baseline trains the model on classical part and test on real QC. In RoQNN, we train the model with parameter shift and test, both on real QC. We consistently outperform noise-unaware baselines.

Compatibility with existing noise mitigation. RoQNN is orthogonal to existing noise mitigation such as extrapolation method. It can be combined with post-measurement normalization (Table 4). The QNN model has 2 blocks, each with three U3+CU3 layers. For "Normalization only", the measurement outcomes of the 3-layer block are normalized across the batch dimension. For "Extrapolation + Normalization", we use extrapolation to estimate the standard deviation of noise-free measurement outcomes. We firstly train the QNN model to convergence and then repeat the 3 layers to 6, 9, 12 layers and obtain four standard deviations of measurement outcomes. Then we linearly extrapolate them to obtain noise-free std. We normalize the measurement outcomes of the 3-layer block to make their std the same as noise-free and then apply the proposed post-measurement norm. Results show that the extrapolation can further improve the QNN accuracy thus being orthogonal.

Additional experiments. Appendix A.4.1 shows that using hardware-specific noise model is beneficial to accuracy; A.4.2 shows high compatibility with latest noise-adaptive quantum compilations; A.4.3 shows high effectiveness on fully quantum (single block) models; A.4.4 shows effect of number of intermediate measurements between blocks. There exists a best measurement number, given the same total model layers; A.4.5 shows the small accuracy gap between using noise model and real QC, indicating high reliability of noise models; A.4.6 shows high effectiveness for difficult tasks such as 10-classification; A.4.7 shows accuracy when using validation set statistics for test set.

5 Conclusion

QNN is a promising candidate to demonstrate practical quantum advantages over classical approaches. The road to such advantage relies on: (1) the discovery of novel feature embedding that encodes classical data non-linearly, and (2) overcome the impact of quantum noise on computation. In this work, we focus on the latter and show analytically and empirically that a noise-aware training pipeline with post-measurement normalization, noise injection, and post-measurement quantization can elevate the QNN robustness against arbitrary, realistic quantum noises. We anticipate that such robust QNN will be useful in the near-term experiments exploring QML applications.

ETHICS STATEMENT

We do not find insights, methodologies of this work potentially harmful to ethnicity. The usage of quantum computing has the potential to solve current computational challenge problems with much higher efficiency and speed. That means potentially lower energy and time cost for certain computation tasks which will lower the burden of computing industry to the environment in terms of energy consumption, carbon dioxide emission, etc.

REPRODUCIBILITY STATEMENT

Since the quantum computing process is intrinsically stochastic, when we run the experiments, we use maximum number of shots on IBMQ machines (8192) to reduce the impact of quantum randomness as mentioned in Section 4.1. For easy reproducing our experimental results, we open-source our QNN training library and training scripts in an anonymous link as stated in Appendix A.3. For the theoretical results regarding noise impact on QNN model measurement results, we add detailed full proof in Appendix A.2.

REFERENCES

Mohammad H Amin, Evgeny Andriyash, Jason Rolfe, Bohdan Kulchytskyy, and Roger Melko. Quantum boltzmann machine. *Physical Review X*, 8(2):021050, 2018.

Johannes Bausch. Recurrent quantum neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1368–1379. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/0ec96be397dd6d3cf2fecb4a2d627c1c-Paper.pdf.

Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, M Sohaib Alam, Shahnawaz Ahmed, Juan Miguel Arrazola, Carsten Blank, Alain Delgado, Soran Jahangiri, et al. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint arXiv:1811.04968*, 2018.

Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

Colin D Bruzewicz, John Chiaverini, Robert McConnell, and Jeremy M Sage. Trapped-ion quantum computing: Progress and challenges. *Applied Physics Reviews*, 6(2):021314, 2019.

Yudong Cao, Jonathan Romero, Jonathan P Olson, Matthias Degroote, Peter D Johnson, Mária Kieferová, Ian D Kivlichan, Tim Menke, Borja Peropadre, Nicolas PD Sawaya, et al. Quantum chemistry in the age of quantum computing. *Chemical reviews*, 119(19):10856–10915, 2019.

Gavin E Crooks. Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition. *arXiv preprint arXiv:1905.13311*, 2019.

Piotr Czarnik, Andrew Arrasmith, Patrick J Coles, and Lukasz Cincio. Error mitigation with clifford quantum-circuit data. *arXiv preprint arXiv:2005.10189*, 2020.

D.H. Deterding. Speaker normalisation for automatic speech recognition. PhD thesis, University of Cambridge, 1989.

Yongshan Ding and Frederic T Chong. Quantum computer systems: Research for noisy intermediate-scale quantum computers. *Synthesis Lectures on Computer Architecture*, 15(2):1–227, 2020.

Suguru Endo, Simon C Benjamin, and Ying Li. Practical quantum error mitigation for near-future applications. *Physical Review X*, 8(3):031027, 2018.

Suguru Endo, Zhenyu Cai, Simon C Benjamin, and Xiao Yuan. Hybrid quantum-classical algorithms and quantum error mitigation. *Journal of the Physical Society of Japan*, 90(3):032001, 2021.

Edward Farhi and Hartmut Neven. Classification with quantum neural networks on near term processors. *arXiv preprint arXiv:1802.06002*, 2018.

Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 212–219, 1996.

Jiaqi Gu, Zheng Zhao, Chenghao Feng, Hanqing Zhu, Ray T Chen, and David Z Pan. Roq: A noise-aware quantization scheme towards robust optical neural networks with low-bit controls. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1586–1589. IEEE, 2020.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.

Vojtěch Havlíček, Antonio D Córcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow, and Jay M Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, 2019.

Zhezhi He, Adnan Siraj Rakin, and Deliang Fan. Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 588–597, 2019.

Maxwell Henderson, Samriddhi Shakya, Shashindra Pradhan, and Tristan Cook. Quanvolutional neural networks: powering image recognition with quantum circuits. *Quantum Machine Intelligence*, 2(1):1–9, 2020.

Mingxia Huo and Ying Li. Self-consistent tomography of temporally correlated errors. *Communications in Theoretical Physics*, 73(7):075101, 2021.

IBM. Ibm quantum. URL https://quantum-computing.ibm.com/.

Qiskit IBM, Apr 2021. URL https://qiskit.org/textbook/ch-quantum-hardware/calibrating-qubits-pulse.html.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2704–2713, 2018.

Weiwen Jiang, Jinjun Xiong, and Yiyu Shi. A co-design framework of neural networks and quantum circuits towards quantum advantage. *Nature communications*, 12(1):1–13, 2021.

Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M Chow, and Jay M Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549(7671):242–246, 2017.

Philip Krantz, Morten Kjaergaard, Fei Yan, Terry P Orlando, Simon Gustavsson, and William D Oliver. A quantum engineer's guide to superconducting qubits. *Applied Physics Reviews*, 6(2): 021318, 2019.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Ying Li and Simon C Benjamin. Efficient variational quantum simulator incorporating active error minimization. *Physical Review X*, 7(2):021050, 2017.

Ji Lin, Chuang Gan, and Song Han. Defensive quantization: When efficiency meets robustness. *arXiv preprint arXiv:1904.08444*, 2019.

Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. *arXiv preprint arXiv:1711.11294*, 2017.

Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum algorithms for supervised and unsupervised machine learning. *arXiv preprint arXiv:1307.0411*, 2013.

Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. *Nature Physics*, 10(9):631–633, 2014.

Seth Lloyd, Silvano Garnerone, and Paolo Zanardi. Quantum algorithms for topological and geometric analysis of data. *Nature communications*, 7(1):1–7, 2016.

Seth Lloyd, Maria Schuld, Aroosa Ijaz, Josh Izaac, and Nathan Killoran. Quantum embeddings for machine learning. *arXiv preprint arXiv:2001.03622*, 2020.

Easwar Magesan, Jay M Gambetta, and Joseph Emerson. Characterizing quantum gates via randomized benchmarking. *Physical Review A*, 85(4):042311, 2012.

K. Matsuoka. Noise injection into inputs in back-propagation learning. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(3):436–440, 1992. doi: 10.1109/21.155944.

Jarrod R McClean, Mollie E Kimchi-Schwartz, Jonathan Carter, and Wibe A De Jong. Hybrid quantum-classical hierarchy for mitigation of decoherence and determination of excited states. *Physical Review A*, 95(4):042308, 2017.

Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature communications*, 9(1):1–6, 2018.

Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, 5(1):1–7, 2014.

Rajeev R. Rao, Kaviraj Chopra, David T. Blaauw, and Dennis M. Sylvester. Computing the soft error rate of a combinational logic circuit using parameterized descriptors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(3):468–479, 2007. doi: 10.1109/TCAD.2007.891036.

Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Physical review letters*, 113(13):130503, 2014.

Michael L Seltzer, Dong Yu, and Yongqiang Wang. An investigation of deep neural networks for noise robust speech recognition. In *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 7398–7402. IEEE, 2013.

Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

Marcus Silva, Easwar Magesan, David W Kribs, and Joseph Emerson. Scalable protocol for identification of correctable codes. *Physical Review A*, 78(1):012347, 2008.

Armands Strikis, Dayue Qin, Yanzhu Chen, Simon C Benjamin, and Ying Li. Learning-based quantum error mitigation. *arXiv preprint arXiv:2005.07601*, 2020.

Kristan Temme, Sergey Bravyi, and Jay M Gambetta. Error mitigation for short-depth quantum circuits. *Physical review letters*, 119(18):180509, 2017.

Joel J Wallman and Joseph Emerson. Noise tailoring for scalable quantum computation via randomized compiling. *Physical Review A*, 94(5):052325, 2016.

Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8612–8620, 2019.

Tianzhe Wang, Kuan Wang, Han Cai, Ji Lin, Zhijian Liu, Hanrui Wang, Yujun Lin, and Song Han. Apq: Joint search for network architecture, pruning and quantization policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2078–2087, 2020.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Yong Xu, Jun Du, Li-Rong Dai, and Chin-Hui Lee. Dynamic noise aware training for speech enhancement based on deep neural networks. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.

Richard M Zur, Yulei Jiang, Lorenzo L Pesce, and Karen Drukker. Noise injection for training artificial neural networks: A comparison with weight decay and early stopping. *Medical physics*, 36(10):4810–4818, 2009.

A Appendix

A.1 Quantum Basics and Quantum Noise

A quantum circuit uses quantum bit (*qubit*) to carry information, which is a linear combination of two basis state: $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, for $\alpha, \beta \in \mathbb{C}$, satisfying $|\alpha|^2 + |\beta|^2 = 1$. An n-qubit system can represent a linear combination of 2^n basis states. A 2^n-length complex statevector of all combination coefficients is used to describe the circuit state. In quantum computations, a sequence of quantum gates are applied to perform unitary transformation on the statevector, i.e., $|\psi(\boldsymbol{x}, \boldsymbol{\theta})\rangle = \cdots U_2(\boldsymbol{x}, \theta_2)U_1(\boldsymbol{x}, \theta_1)|0\rangle$, where $\boldsymbol{x}$ is the input data and $\boldsymbol{\theta}$ is the trainable parameters of rotation quantum gates. As such, the input data and trainable parameters are embedded in the quantum state $|\psi(\boldsymbol{x}, \boldsymbol{\theta})\rangle$. Finally, the computation results are obtained by qubit readout/measurement which measures the probability of a qubit state $|\psi\rangle$ collapsing to either $|0\rangle$ (i.e., output $y = +1$) or $|1\rangle$ (i.e., output $y = -1$) according to $|\alpha|^2$ and $|\beta|^2$. With sufficient samples, we can compute the expectation value: $\mathbb{E}[y] = (+1)|\alpha|^2 + (-1)|\beta|^2$. By cascading multiple blocks of quantum gates and measurements, a non-linear network can be constructed to perform ML tasks.

In real quantum computer systems, errors would likely occur due to imperfect control signals, unwanted interactions between qubits, or interference from the environment (Bruzewicz et al., 2019; Krantz et al., 2019). As a result, qubits undergo *decoherence error* (spontaneous loss of its stored information) over time, and quantum gates introduce *operation errors* (e.g., coherent errors and stochastic errors) into the system. These noisy systems need to be characterized (Magesan et al., 2012) and calibrated (IBM, 2021) frequently to mitigate the impact of noise on computation. Noise modeling helps to paint a realistic picture of the behavior and performance of a quantum computer and enables noisy simulations (Ding & Chong, 2020). While exact modeling and simulation is challenging, many approximate strategies (Magesan et al., 2012; Wallman & Emerson, 2016) have been developed based on Pauli/Clifford Twirling (Nielsen & Chuang, 2002; Silva et al., 2008).

A.2 General Framework for Quantum Noise Analysis

In this work, we examine how to characterize and mitigate the impact of quantum noises on quantum neural networks. We observe that the trainable quantum gates and post-measurement information

processing play a huge role in boosting the algorithmic robustness to realistic quantum noises. In the following analysis, we restrict attention to: (1) a general (mixed) quantum state $\rho(\boldsymbol{x}, \boldsymbol{\theta})$ resulting from a QNN for input data $\boldsymbol{x}$ and trainable parameters $\boldsymbol{\theta}$, (2) single-qubit measurement output, and (3) any fixed but unknown quantum noise. For multi-qubit quantum neural networks, similar analysis follows when considered qubit by qubit.

A.2.1 Measurement of Quantum Neural Networks

Definition A.1. (Measurement procedure)**.** We measure a quantum state ρ in the computational basis $|b\rangle : b \in \{0, 1\}$ and output $z = +1$ if we obtain $|0\rangle \langle 0|$ and $z = -1$ if we obtain $|1\rangle \langle 1|$.

The expectation value of such measurement contains useful information about the quantum state ρ:

$$E_Z \equiv \mathbb{E}[z] = tr(Z\rho), \tag{1}$$

where Z is the Pauli-Z matrix: $Z = (+1)|0\rangle \langle 0| + (-1)|1\rangle \langle 1|$ and $tr(\cdot)$ is the trace. We can estimate the expectation value by repeating the experiment by s times, obtaining $z_1, \ldots, z_s$, with each $z_j \in \{+1, -1\}$, and calculate their empirical mean: $y = \sum_{j=1}^{s} \frac{z_j}{s}$. Throughout this work, we use $s = 8192$ shots for the experiments to keep the variance low.

Definition A.2. (Noise processes)**.** A physical process (such as quantum noises) that can happen to a mixed quantum state ρ can be described as a linear map: $\rho \to \mathcal{E}(\rho)$, such that

$$\mathcal{E}(\rho) = \sum_k O_k \rho O_k^\dagger. \tag{2}$$

The O_k's are Kraus operators satisfying $\sum_k O_k^\dagger O_k = I$. The noise process for a quantum neural network can be challenging to characterize, as it depends not only on the input to the network but also on the qubits and quantum gates used in the network. We wish to analyze this noise process by decoupling its dependence on the input data. We assume that each O_k has no explicit dependence on the classical input data; this is reasonable when we fix the model architecture.

A.2.2 Proof of Theorem 3.1

Now we are ready to analyze the effect of quantum noises on the measurement result from a quantum neural network. For classical data $\boldsymbol{x_i}$ from the input data set $\boldsymbol{x}$, we construct a quantum neural network that embeds the classical data in the quantum state $\rho_i \equiv \rho(\boldsymbol{x_i}, \boldsymbol{\theta_i})$ where $\boldsymbol{\theta_i}$ is some training parameters. The output of the network is the expectation value of the measurement outcome $E^*_{z,i} \equiv tr(Z\rho_i)$. However, in reality, the results are transformed by some unknown process $\rho_i \to \mathcal{E}(\rho_i)$. The goal is to quantify the impact of the quantum noise on the expectation value.

Theorem 3.1. (formal version)*. There exists some real parameters β_i and γ, such that the expectation value of the measurement results $E_{z,i}$ for input data $\boldsymbol{x_i}$ with the presence of any valid quantum noise $\mathcal{E}(\rho)$ can be described as a linear map from the noiseless value $E^*_{z,i}$:*

$$E_{z,i} = \gamma E^*_{z,i} + \beta_i. \tag{3}$$

Here γ is a scaling constant independent of the input data $\boldsymbol{x_i}$.

Proof. Suppose in the noiseless scenario the quantum state obtained from a quantum neural network is denoted as ρ, and the expectation value of its measurement results is $E^*_z = \mathbb{E}(\rho) = tr(\rho Z)$. Assume now the ρ undergoes some quantum noise processes $\mathcal{E}(\rho) = \sum_k O_k \rho O_k^\dagger$. Therefore, in the presence of noise, the expectation value becomes:

$$E_z = \mathbb{E}[\mathcal{E}(\rho)] = tr(\mathcal{E}(\rho)Z) = \sum_k tr(O_k \rho O_k^\dagger Z) = \sum_k tr(\rho O_k^\dagger Z O_k), \tag{4}$$

where the third and fourth equality is from the properties of trace. We can further utilize the fact that an arbitrary quantum state can be expanded as:

$$\rho = \frac{1}{2}\left(tr(\rho)I + tr(X\rho)X + tr(Y\rho)Y + tr(Z\rho)Z\right). \tag{5}$$

Table 5: Hardware-specific noise model can achieve best accuracy.

Use noise model of → / Inference on ↓	Santiago	Yorktown	Lima
Santiago	**0.90**	0.55	**0.91**
Yorktown	0.41	**0.55**	0.5
Lima	0.76	0.76	**0.89**

Table 6: MNIST-2 accuracy with noise-adaptive compilation enabled (Qiskit optimization level=3).

Method	Santiago	Yorktown	Belem	Athens
Baseline	0.68	0.83	0.83	0.54
+Norm	0.87	0.86	0.91	0.51
+Noise & Quant	**0.92**	**0.92**	**0.91**	**0.93**

If we denote $\Omega = \sum_k O_k^\dagger Z O_k$, we obtain

$$E_z = \frac{1}{2}tr(\Omega) + \frac{1}{2}tr(X\Omega)tr(X\rho) + \frac{1}{2}tr(Y\Omega)tr(Y\rho) + \frac{1}{2}tr(Z\Omega)tr(Z\rho). \tag{6}$$

Notice that $tr(\Omega) = 0$ and $tr(Z\rho) = E_z^*$. We can set $\gamma = \frac{1}{2}tr(Z\Omega) \in [-1, 1]$ and $\beta_\rho = \frac{1}{2}tr(X\Omega)tr(X\rho) + \frac{1}{2}tr(Y\Omega)tr(Y\rho)$. We arrive at the linear map as desired. □

A.3 Open-Sourced QNN Library

To accelerate QNN model training, we build a PyTorch library named `torchquantum`. Its APIs are implemented similar to existing operations in PyTorch. So it makes quantum circuit construction as easy as a standard neural network model. It supports all common quantum gates. The state vector and unitary matrix of each gate are implemented with a native `torch.Tensor` data type. The quantum simulation is achieved with complex-valued differentiable matrix multiplication operators such as `torch.bmm`.

Compared with existing training frameworks such as PennyLane (Bergholm et al., 2018), it has several unique advantages: (1) It supports training in batch mode to accelerate training, while PennyLane cannot support batched training. (2) It supports dynamic and static computational graphs. Dynamic mode simulates each gate individually, so the state vector after each gate can be obtained for easy debugging. Static mode optimizes tensor network simulation by fusing the unitary of multiple gates before applying to the state vector, reducing the computation amount. (3) With PyTorch's GPU acceleration support, all the simulations can be accelerated with GPUs. PyTorch's native automatic differentiation can be applied to train the gate parameters. (4) It supports easy extraction of the noise models from QC device providers and can perform noise injection during the training. (5) It supports easy conversion between PyTorch QNN model and IBM Qiskit QuantumCircuit, such that we can perform end-to-end training-to-deployment flow. It contains multiple ready-to-use circuit templates such as random and strongly-entangled layers. All the steps in RoQNN are implemented with the `torchquantum`. The library has great potential to accelerate research in parameterized QC, especially for QNN models and Variational Quantum Eigensolver (VQE), etc.

We include the library within the supplementary materials. It can also be directly accessed with this anonymous link.

A.4 Additional Experiments

A.4.1 Importance of hardware-specific noise model.

We train three QNN models for Fashion-2 with the same architecture but different noise models from 3 devices and then deploy each model. Results in Table 5 show a diagonal pattern: the best accuracy is achieved when the noise model and inference device are the same. This is due to various noise magnitude and distribution on different devices. For instance, the gate error of Yorktown is 5× larger than Santiago, so using Yorktown noise information for model running on Santiago is too large. Therefore, a hardware-specific noise model is necessary for proper noise injection. However, this also marks the limitation of this work, as repeated training may be required when the noise model is updated. A future direction is to explore how to finetune already trained QNN for fast adaption to a new noise setting, thus reducing the marginal cost.

A.4.2 Compatibility with existing noise-adaptive compilation.

We further show the compatibility of RoQNN with state-of-the-art noise-adaptive quantum compilation techniques. Specifically, we set the optimization level of Qiskit compiler to the highest 3, which

Table 7: Effect of RoQNN on fully quantum models.

IBMQ Machine	Model	Method	MNIST-4	Fashion-4	Vowel-4	MNIST-2	Fashion-2	Cifar-2
Santiago	3 Layer	Baseline	0.64	0.78	0.41	0.94	0.89	**0.59**
		RoQNN	**0.78**	**0.82**	**0.53**	**0.96**	**0.90**	0.58
Santiago	6 Layer	Baseline	0.61	0.37	0.22	0.51	0.52	0.52
		RoQNN	**0.62**	**0.69**	**0.22**	**0.84**	**0.89**	**0.56**
Yorktown	3 Layer	Baseline	0.49	0.53	0.4	0.88	0.85	0.51
		RoQNN	**0.55**	**0.66**	**0.42**	**0.9**	**0.91**	**0.55**
Yorktown	6 Layer	Baseline	0.22	0.33	**0.26**	0.73	0.80	**0.54**
		RoQNN	**0.42**	**0.35**	0.25	**0.78**	**0.80**	0.52
Belem	3 Layer	Baseline	0.53	**0.60**	0.37	0.64	0.81	0.51
		RoQNN	**0.58**	0.42	**0.39**	**0.93**	**0.85**	**0.55**
Belem	6 Layer	Baseline	0.27	0.18	0.21	0.54	0.48	0.43
		RoQNN	**0.43**	**0.31**	**0.22**	**0.54**	**0.54**	**0.52**

Table 8: Effect of number of intermediate measurements.

Task	1 Block $\times$ 6 Layers	2 Blocks $\times$ 3 Layers	3 Blocks $\times$ 2 Layers	6 Blocks $\times$ 1 Layer
MNIST-4	0.62	**0.74**	0.71	0.66
Fashion-4	0.69	**0.82**	0.78	0.68

enables noise-adaptive qubit mapping and instruction scheduling. Then we inference the RoQNN trained model and compare the accuracy of MNIST-2 in Table 6. With noise-adaptive compilation, the accuracy of baseline models is improved. While on top of that, the RoQNN can still provide over 10% accuracy improvements, demonstrating the extensive applicability of our methods.

A.4.3 Experiments on Fully Quantum Models

For the results in Section 4, the QNN models contain multiple blocks. Here we further experiment on *fully quantum* models which only contains one single block to show the strong generality of RoQNN as in Table 7. We select two fully quantum models, with three and six U3+CU3 layers, respectively, and experiment with six tasks on two machines. We apply the post-measurement normalization and quantization to the measurement outcomes of the last layer and use noise factor 0.5 and quantization level 6. No intermediate measurements are required. Our methods can still outperform baselines by **7.4%** on average. Therefore, The noise injection can be applied to different kinds of variational quantum circuits, no matter whether the output of one layer is measured and passed to the next layer. Furthermore, the post-measurement normalization and quantization can also benefit various quantum circuits because they reduce the noise impact on measurement outcomes.

A.4.4 Experiments on Effect of Number of Intermediate Measurements

We also explore under the same number of parameters whether a fully quantum model is the best choice in the NISQ era. There exists a *tradeoff* on the number of intermediate measurements as in Table 8. More measurements mean less noise impact because we can perform post-measurement normalization and quantization on measurement outcomes. However, measurements will collapse the state vector in the large Hilbert space back to the small classical space, hurting the model capacity. We perform experiments on the IBMQ-Santiago machine and find there exists a sweet spot to achieve the highest deployment accuracy: the best model contains 2 blocks and each has 3 layers.

Furthermore, we show direct accuracy comparisons between the original (with measurements in between) QNN and fully-quantum QNN in Table 9. In each row, they have exactly the same dataset, same hardware. They have nearly the same architecture: same encoder/measurement, same gate sets, same layers, same number of parameters; the only difference is whether being measured and encoded back to quantum in the middle.

Table 9: Direct comparison between QNN models with measurement in between and fully-quantum QNN models.

Machine	Task	Fully-Quantum (6 Layers)	Original (2 Blocks × 3 Layers)
Santiago	MNIST-4	0.62	**0.74**
Santiago	Fashion-4	0.69	**0.82**
Santiago	MNIST-2	0.84	**0.86**
Belem	MNIST-4	**0.43**	0.37
Belem	Fashion-4	0.31	**0.34**
Belem	MNIST-2	0.54	**0.60**

Table 10: Accuracy gap between evaluation using noise model and real QC.

Machine	Model	Method	MNIST-4	Fashion-4	Vowel-4	MNIST-2	Fashion-2	Cifar-2
Santiago	2 Blocks × 12 Layer	Noise model	0.73	0.74	0.51	0.95	0.92	0.65
		Real QC	0.68	0.75	0.48	0.94	0.88	0.59
Yorktown	2 Blocks × 2 Layer	Noise model	0.68	0.7	0.44	0.92	0.90	0.59
		Real QC	0.62	0.65	0.44	0.93	0.86	0.60
Belem	2 Blocks × 6 Layer	Noise model	0.64	0.72	0.41	0.96	0.82	0.64
		Real QC	0.58	0.62	0.41	0.88	0.8	0.61

From the experimental results, we can see that under the total 6-layer setting, the 2 Block x 3 Layer can have better accuracy in most cases. This is because we perform normalization and quantization in the middle that can mitigate the noise impacts.

We also would like to emphasize that how to design the best architecture is not the main focus of our work. RoQNN is architecture-agnostic and can be applied to various architectures to improve their robustness on real QC devices, as illustrated in paper Table 2.

A.4.5 Accuracy Gap between Using Noise Model and Real QC

To demonstrate the reliability of noise models, we show the accuracy gap of QNN models evaluated with noise model and on real QC as in Table 10. We can see that the accuracy gaps are typically smaller than 5%, indicating *high reliability* of noise models.

A.4.6 Accuracy improvements comparison as number of classes increases

Since we have different tasks with various number of classes, we compare the average accuracy improvements between them in Table 11. We can see that the relative accuracy improvement on 10-class (230%) is significantly higher than 4-class and 2-class. That of 4-class is also higher than 2-class. To improve the same absolute accuracy, it is clearly more difficult on a 10-class task than on a 2-class task. So RoQNN is highly effective on 10-class tasks.

A.4.7 Experiments on Using Validation Set Statistics for Test Set

If the test batch size is small for the deployment on real QC hardware, then the statistics may not be accurate enough for post-measurement normalization. In this case, we can profile the statistics of the validation set on real hardware ahead of time and then use the validation set mean and std to normalize the test set measurement outcomes.

We experiment with three tasks, each on three quantum devices. We show the mean and std of measurement outcomes of each qubit on the validation set and test set as in Table 12. We can see that the statistics of validation and test sets are similar. The last column of Table 12 shows the *accuracy of test set* using statistics of the test set itself and validation set, respectively. In 9 benchmarks, the accuracy of two settings is very close. The average accuracy of using test set stats is 0.67; using validation set stats is 0.65.

Table 11: Improvements are still significant as the number of classes increases.

Task Average Accuracy	Baseline	RoQNN	Absolute Improvement	Relative Improvement
2-classification	0.58	0.76	0.28	48%
4-classification	0.31	0.57	0.26	84%
10-classification	0.1	0.33	0.23	230%

Table 12: Statistics of test and validation set; Accuracy of test set using test stats and validation stats.

Task	Stats	MEAN	STD	Accuracy
Fashion-4-Santiago	Test Stats	[0.0469, 0.0025, -0.0581, -0.0191]	[0.0868, 0.0496, 0.1021, 0.1152]	0.75
	Valid Stats	[0.0679, 0.0025, -0.0519, -0.0473]	[0.0915, 0.0448, 0.0884, 0.1114]	0.70
Fashion-4-Yorktown	Test Stats	[-0.0396, 0.0478, 0.0995, 0.1375]	[0.1279, 0.3368, 0.1761, 0.1538]	0.65
	Valid Stats	[-0.0362, 0.0771, 0.0965, 0.1535]	[0.1230, 0.3233, 0.1835, 0.1584]	0.65
Fashion-4-Belem	Test Stats	[0.1118, 0.0075, 0.0901, -0.0005]	[0.0868, 0.1511, 0.1391, 0.2039]	0.62
	Valid Stats	[0.1508, -0.0130, 0.0533, 0.0478]	[0.0882, 0.1298, 0.1315, 0.1401]	0.53
Vowel-4-Santiago	Test Stats	[0.1091, 0.0526, 0.0290, 0.2172]	[0.0551, 0.0260, 0.0554, 0.0422]	0.48
	Valid Stats	[0.1042, 0.0698, 0.0458, 0.1951]	[0.0418, 0.0226, 0.0443, 0.0362]	0.43
Vowel-4-Yorktown	Test Stats	[0.0900, -0.3700, -0.2524, 0.1645]	[0.0997, 0.0580, 0.0663, 0.1198]	0.44
	Valid Stats	[0.0841, -0.3869, -0.2948, 0.1736]	[0.0946, 0.0651, 0.0615, 0.1199]	0.41
Vowel-4-Belem	Test Stats	[0.0115, 0.0800, 0.1703, 0.1775]	[0.0171, 0.0411, 0.0518, 0.0293]	0.41
	Valid Stats	[-0.0213, 0.0459, 0.1930, 0.1628]	[0.0145, 0.0335, 0.0478, 0.0263]	0.40
MNIST-2-Santiago	Test Stats	[-0.0581, -0.0657, 0.0088, 0.0170]	[0.0737, 0.1090, 0.1561, 0.1351]	0.94
	Valid Stats	[-0.0739, 0.0001, -0.0113, 0.00239]	[0.0666, 0.0840, 0.1468, 0.1167]	0.95
MNIST-2-Yorktown	Test Stats	[0.0892, -0.0007, 0.0548, 0.0485]	[0.1281, 0.3501, 0.2100, 0.2975]	0.93
	Valid Stats	[0.0704, 0.0536, 0.0204, 0.1043]	[0.1377, 0.3813, 0.2596, 0.2955]	0.91
MNIST-2-Belem	Test Stats	[-0.0649, 0.1949, 0.0540, 0.1313]	[0.0856, 0.1137, 0.1553, 0.1688]	0.88
	Valid Stats	[-0.0540, 0.2074, 0.0744, 0.1872]	[0.0561, 0.1008, 0.1345, 0.1103]	0.91
Average	Test Stats	—	—	**0.67**
	Valid Stats	—	—	**0.65**

Therefore, using the statistics of validation set can bring similar accuracy to using statistics of test set itself; thus the RoQNN can support small test batch size using validation set stats.

A.5 Hyperparameters for main results

Table 13 shows the detailed noise factor and quantization level for all the tasks in Figure 8.

Table 13: Hyperparameters of Figure 8.

Task, (noise-factor, quantization level)	MNIST-4	Fashion-4	Vowel-4	MNIST-2	Fashion-2	Cifar-2
QNN (2 Blocks $\times$ 12 Layers) on Santiago	(1, 3)	(0.5, 6)	(0.5, 6)	(1, 4)	(1, 6)	(0.5, 6)
QNN (2 Blocks x 2 Layers) on Yorktown	(0.5, 6)	(1, 5)	(0.1, 5)	(0.5, 5)	(0.1, 6)	(0.1, 3)
QNN (2 Blocks x 6 Layers) on Belem	(0.5, 5)	(1.5, 6)	(0.5, 3)	(0.5, 5)	(0.1, 4)	(0.5, 6)
QNN (3 Blocks x 10 Layers) on Athens	(0.1, 6)	(0.1, 5)	(0.5, 6)	(0.1, 6)	(0.5, 6)	(0.1, 6)

Task, (noise-factor, quantization level)	MNIST-10	Fashion-10
QNN (2 Blocks x 2 Layers) on Melbourne	(0.1, 6)	(0.1, 5)