Battle Against Fluctuating Quantum Noise: Compression-Aided Framework to Enable Robust Quantum Neural Network

***Abstract*—Recently, we have been witnessing the scale-up of superconducting quantum computers; however, the noise of quantum bits (qubits) is still an obstacle for real-world applications to leveraging the power of quantum computing. Although there exist error mitigation or error-aware designs for quantum applications, the inherent fluctuation of noise (a.k.a., instability) can easily collapse the performance of error-aware designs. What's worse, users can even not be aware of the performance degradation caused by the change in noise. To address both issues, in this paper we use Quantum Neural Network (QNN) as a vehicle to present a novel compression-aided framework, namely *CE-RQnn*, which will adapt a trained QNN to fluctuating quantum noise. Emulation results on an earthquake detection dataset show that CE-RQnn can achieve 14.91% accuracy gain on average in 146 days over a noise-aware training approach. For the execution on a 7-qubit IBM quantum processor, ibm-jakarta, CE-RQnn can consistently achieve 12.52% accuracy gain on earthquake detection.**

I. Introduction

We are currently in the Noisy Intermediate-Scale Quantum (NISQ) era where noise and scalability have been two well-known and critical issues in quantum computing. Nowadays, we have been witnessing the rapid development of superconducting quantum computers and the scalability issue is gradually mitigated. As an example, IBM took only 6 years to scale up its quantum computers from 5 to 433 qubits. However, the high noise in quantum computing is still an obstacle for real-world applications to take advantage of quantum computing. There are many sources of noise in quantum computing, such as gate errors and readout errors. As shown in Fig. 1, the color of qubits and their connections indicates the Pauli-X and CNOT gate error from the IBM Belem backend.

Unlike CMOS noise that is within a small range under 10^{-15}, the noise on qubits can reach 10^{-2} to 10^{-4}. Moreover, as shown in Fig. 1, noise from 1-year-long profiling on IBM Belem backend is fluctuating in a wide range, called "fluctuating quantum noise". Although there exist works to improve the robustness of quantum circuits to noise, such as error mitigation [1] and noise-aware designs [2–4], these works commonly perform the optimization based on the noise at one moment. The fluctuating quantum noise can easily make the quantum circuit lose its robustness. Thus, new innovations are needed to deal with the fluctuating noise.

In this paper, we propose a novel framework, namely "CE-RQnn", to address the above issues. To illustrate our framework, we use Quantum Neural Network (QNN) — a.k.a, variational quantum circuit (VQC) as an example — since the learning approach has been shown to be an effective way for a wide range of applications from different domains (such as chemistry [5], healthcare [6], and finance [7]); and meanwhile, [8] recently has shown the potential quantum speedups using VQC. To deal with fluctuating quantum noise, a straightforward method is to apply a noise-aware training approach to retrain QNN before each inference; however, it will obviously incur high costs. More importantly, as the quantum noise changes in a wide range, we observe that a set of parameters will

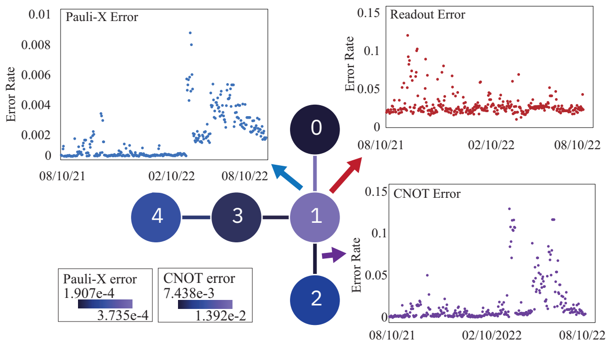


Fig. 1. The fluctuating noise observed on IBM backend belem

deviate from the loss surface, which impedes the noise-aware training to find optimal solutions.

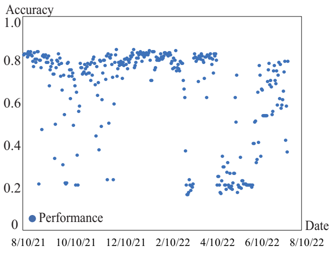
To address this problem, we made an investigation and found that these parameters will lead to short circuits after the logical-to-physical compilation. Equivalent to classical neural networks, these QNN parameters can be regarded as compression levels. Therefore, QNN compression can be helpful to optimize the model in a noisy environment. Based on our observations, in CE-RQnn, we develop a novel noise-aware QNN compression algorithm. It will interplay with another component: A model adapter to the fluctuating noise. As such, CE-RQnn can automatically and efficiently obtain the QNN models adapted to run-time noise for high performance.

The main contributions of this paper are as follows.

- We reveal that the fluctuating quantum noise will collapse the performance of quantum neural networks (QNNs).
- We develop a noise-aware QNN compression algorithm to adapt pretrained QNN model to a given noise.
- On top of the noise-aware compression algorithm, we further propose a 2-stage framework to adapt QNN model to fluctuating quantum noise automatically.

Evaluation is conducted on both IBM Aer simulator and IBM quantum processor ibm-jakarta. Experiment results on MNIST, earthquake detection dataset, and Iris show the effectiveness and efficiency of CE-RQnn. Especially, CE-RQnn achieves superior performance among the competitors on all datasets. It achieves 16.32%, 38.88%, and 15.36% accuracy gain compared with the default configuration without optimization. Even compared with noise-aware training before every execution, CE-RQnn can still achieve over a 15% accuracy improvement on average; meanwhile, CE-RQnn can reduce the training time to over 100×. For the execution on a 7-qubit IBM quantum processor ibm-jakarta, CE-RQnn can consistently achieve a 13.7% accuracy gain on the earthquake detection dataset.

The remainder of the paper is organized as follows. Section 2 reviews the related work and provides observations, challenges and motivations. Section 3 presents the proposed CE-RQnn framework. Experimental results are provided in Section 4 and concluding remarks are given in Section 5.



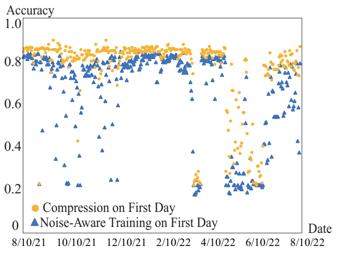


Fig. 2. The accuracy of QNN on 4-class MNIST from August 2021 to August 2022 on IBM backend belem using Qiskit Simulation.

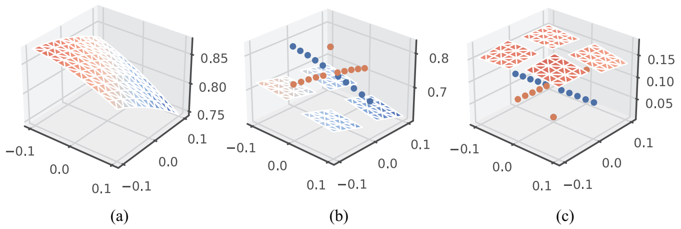


Fig. 3. Noise-aware training may miss optimal solution: (a) Optimization surface of 2-parameter VQC under noise free environment. (b) Optimization surface of the same VQC under a noisy environment. (c) Difference between (a) and (b).

II. Related Work, Challenge, and Motivation

A. Related Work

There are two commonly applied approaches to mitigate the effect of quantum noise. One is to adjust quantum circuits by noise-aware training [4] and mapping [3]. Noise-aware training is an adversary method, which injects noise into the training process so that the parameter of model can be learned from both datasets and devices. Noise-aware mapping is to minimize the sum of accumulated noise by changing the mapping from logical qubits to physical qubits. Another method is to estimate the ideal outputs by post-processing measurement data, e.g., zero-error noise extrapolation [9], probabilistic error cancellation[10], and virtual distillation[11]. However, these two kinds of approaches are based on noise at one moment, so we can hardly maintain the performance without redoing these methods as noise changes anytime.

There are a few works addressing the fluctuating-noise issue, which can be quantitatively analyzed in two aspects: the noise of device and the distance between ideal results and noisy results. [12] evaluated the stability of NISQ devices with multiple metrics that characterize stability. [13, 14] defined Hellinger distance, computational accuracy and result reproducibility to expound the distance between ideal result and noisy observation. These works show the significance of reproducing results in a long-term execution on a quantum device. However, it is still not clear what effects the fluctuating noise will make on quantum applications.

B. Observation, Challenge and Motivation

This section will provide our observations on the simulation performance of a quantum neural network model on a noisy quantum computer over a period of one year.

Observation 1: Fluctuating noise can collapse the model accuracy of a noise-aware trained QNN model.

Fig. 2 (a) shows the daily accuracy of a QNN model on the quantum processor in Fig. 1, which demonstrates that the accuracy can be varied significantly along with the fluctuating noise. The QNN model is trained using the noise-aware training method [4] based on the calibration (noise) data on day 1 (8/10/22). Its accuracy is over 80% from day 1 to day 22; however, when error rates increased on day 24, the accuracy decreased to 22%. This observation shows that the noise-aware design can obtain a robust quantum circuit to a certain range of noise, but the performance can be collapsed when the noise rate goes beyond a threshold.

Challenge 1: Noise on qubits can create breakpoints beyond the optimization surface, creating difficulties in training

Relying on training only to adapt to noise may miss the opportunity to find the optimal solution. Fig. 3(a)-(b) shows an example of the results of a quantum circuit with 2 parameters (x-axis and y-axis) in a perfect environment (simulation) and in a noisy environment (realistic). From the difference shown in Fig. 3(c), we can see that the breakpoints in the loss landscape are with much lower noise. Given the optimization started from a random point, in order to reach these breakpoints, it requires a dedicated learning rate for each parameter, which is almost impossible in a training algorithm.

Motivation 1: Not only relying on training but also involving compression to battle against quantum noise.

Looking closer at these breakpoints in Fig. 3(c), we observe that the parameter value of 0 creates these breakpoints. We also observe such breakpoints around other specific values, like $\frac{\pi}{2}$, π, $\frac{3\pi}{2}$, etc. We further investigate the root cause and we found the improvement in performance is caused by the reduction of physical circuit length. More specifically, due to the transpilation (i.e., logical-to-physic qubit mapping), the logical quantum gate with different values will result in varied circuit lengths on physical qubits. Analog to the classical neural networks, the reduction of the network complexity is known as compression. Based on the above understanding, it motivates us to employ compression techniques to address the quantum noise. For the same noise setting as Fig. 2(a), we perform the model compression on Day 1, and results are reported in Fig. 2(b). It is clear that the results of compression (i.e., yellow points) are much better than the noise-aware training (i.e., blue points). However, we still observe a significant accuracy drop between March 15 to May 29.

Observation 2: Behaviors of fluctuating noise on different qubits has heterogeneity.

To explore the root cause of the accuracy drop, we investigate the change of CNOT noise on all pairs of qubits on three dates: (1) Feb. 12, (2) March 15, and (3) April 25. Fig.4 (a) reports the results. It is clear that the qubits on March 15 and April 25 have much higher noise than that on Feb. 12. A more interesting observation is that the fluctuating noise on different qubits has heterogeneity. Specifically, on Feb. 12, $\langle q_3, q_4\rangle$ has the highest noise, while $\langle q_1, q_2\rangle$ becomes the noisiest one on March 15 and April 25. Along with the heterogeneous changes of noise on qubits, the compressed QNN model may lose its robustness, which causes the accuracy degradation between March 15 to May 29.

Motivation 2: Adding noise awareness in compression to battle against fluctuating quantum noise

To address the above challenges, only conducting noise-agnostic compression with the objective of minimizing circuit length is not enough; on the other hand, we need to take the heterogeneous noise on qubits into consideration. This motivates us to develop a "noise-aware compression" on QNNs, which can involve the noise level at each qubit to guide how the model will be compressed. Details will be introduced in Sec ??. We use an example to illustrate the necessity of noise-aware compression. As shown in Fig. 4 (b), on March 15 and April 25, we observe significant changes in the noise of qubits, which makes the original compressed model suffer an accuracy drop from 79% to 22.5% and 56.5%. By using a noise-aware compression on these dates, we resume the accuracy to 38.5% and 80% on March

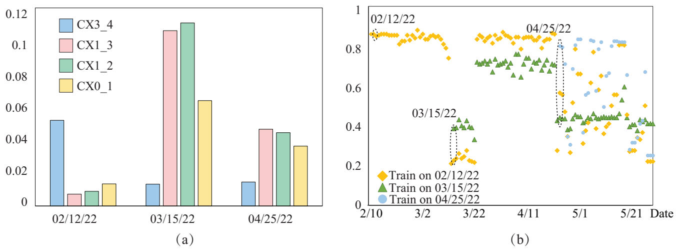


Fig. 4. Noise-aware compression is needed: (a) CNOT gate noise on three days. (b) Noise-aware compressing and tuning models on the three days in (a) and testing the on the following days.

15 and April 25, respectively.

With noise-aware compression, now, the problem is how frequently should we perform the compression. Due to the fluctuation of quantum noise, one straightforward idea is to leverage the noise-aware compression before using it, so that it can adapt to the new noise. However, this can be too costly.

Observation 3: Models can be re-utilized.

Let's see the original model in Fig. 4(b), which experienced a dramatic accuracy drop on March 15. However, after 9 days, on March 24, the accuracy of this model resumed to 80.5%. Therefore, the previous model can be re-utilized later.

Motivation 3: A model repository can avoid model optimization every day to improve efficiency.

The above observation inspires us to build a model repository to keep a set of models. At run time, instead of directly performing optimization (i.e., compression) for a new noise, we can first check if models in the repository can adapt to the noise. In this way, it is possible to reuse pre-optimized models and significantly reduce the optimization cost at run time.

With the above observations and motivations, we propose a novel framework CE-RQnn in the following section to devise a noise-aware compression algorithm, build a model repository upon historical data, and manage it at run time.

III. Compression-Aided Framework

This section will introduce our proposed compression-aided framework, namely CE-RQnn, to battle against fluctuating quantum noise. Before introducing details, we first formally formulate the problem as follows: Given a QNN model M and a model repository Q, the quantum processor Q with history calibration (noise) data D_t, the current calibration data D_c of Q, the problem is how to leverage the calibration data (i.e., D_t and D_c) to find a model M' with the objective of maximizing the accuracy of M' on Q with D_c. Our framework contains 2 main components: (1) a noise-aware compression algorithm, and (2) a model adapter to leverage noise-aware compression.

A. Noise-Aware Compression

Quantum neural network compression was recently proposed [15] to reduce circuit length, where an Alternating direction method of multipliers (ADMM) approach is applied. In this paper, we will also employ ADMM as the optimizer for noise-aware compression, and the new challenge here is how to involve noise awareness in the compression process. In the previous work, the authors conducted compression upon the logical quantum circuit. However, to involve noise in the compression of quantum gates, we need to fix the physic qubits of each quantum gate. *Therefore, we will take the quantum circuit after routing on restricted topology as input, instead of the logical quantum circuit.*

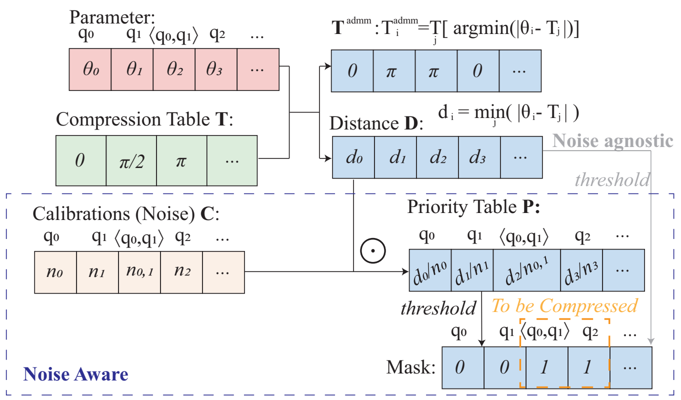


Fig. 5. Noise-aware mask generation in ADMM process.

Before introducing our proposed algorithm, we first define the notations and the optimization problem which will be used in ADMM. We denote $\boldsymbol{T}$ as a table of compression-level, which are breakpoints in the example of *Motivation 1*. We denote the function of a VQC under a noise-free (a.k.a., perfect, denoted by p) environment as $W_p(\boldsymbol{\theta})$, where $\boldsymbol{\theta} = [\theta_1, \theta_2, \cdots, \theta_n]$ are a set of trainable parameters, and θ_i is the parameter of gate g_i. With the consideration of noise, the function of VQC will be changed to $W_n(\boldsymbol{\theta})$. On top of these, the deviation caused by noise can be defined as $N(\boldsymbol{\theta}) = W_n(\boldsymbol{\theta}) - W_p(\boldsymbol{\theta})$. For a quantum gate g_i, it is associated with a physic qubit q_k or a pair of physic qubits $\langle q_k, q_l \rangle$. For simplicity, we denote such an association as a function $\langle q_k, q_l \rangle = A(g_i)$, and we use $k = l$ to represent g_i is associated with one qubit q_i. We denote $\boldsymbol{C}$ as a table of calibration data, and the notation $n_{k,l} \in \boldsymbol{C}$ or the function $C(q_k, q_l)$ to represent the noise rate on qubit q_k and q_l (or q_k if $k = l$). Then, the problem can be formulated as below.

$$\min_{\boldsymbol{\theta}} \quad W_p(\boldsymbol{\theta}) + N(\boldsymbol{\theta}) \tag{1}$$

Now, to enable using ADMM to perform noise-aware compression, we first decompose the optimization problem in Eq. 1 to two sub-problems and solve them separately: (1) maximize accuracy and (2) minimize the deviation caused by noise. The first subproblem can be solved by a gradient-based optimizer. For the second problem, we will use a set of auxiliary variables and an indicator function to resolve it, which will be introduced later. Therefore, we reformulate the optimization problem that can be solved by ADMM as below.

$$\min_{\{\theta_i\}} \quad f(W_p(\boldsymbol{\theta})) + N(\boldsymbol{Z}) + \sum_{\forall z_i \in \boldsymbol{Z}} s_i(z_i), \tag{2}$$

where $\boldsymbol{Z}$ is a set of auxiliary variables for subproblem decomposition and $z_i \in \boldsymbol{Z}$ is corresponding to $\theta_i \in \boldsymbol{\theta}$; function f represents training loss on the given dataset; $\boldsymbol{T^{admm}}$ is gate-related compression table build on T; and $s_i(z_i)$ is an indicator, which will indicate whether the parameter θ_i will be pruned or not.

In the r^{th} iteration of ADMM optimization, one key step is to determine whether or not to compress a parameter. According to the parameter θ_i, compression table T, and noise data $n_{k,l}$, we will build a mask. Fig. 5 illustrates the process to create the mask, which is composed of three steps. First, by comparing the parameter θ_i with each compression level in table T, we generate two tables: $\boldsymbol{T^{admm}}$ and $\boldsymbol{D}$. The i^{th} element in $\boldsymbol{T^{admm}}$ is denoted as T_i^{admm}, which is the nearest compression level of parameter θ_i; while d_i is the minimum distance between parameter θ_i and any compression level. *Note that in a noise-agnostic compression [15], a mask will be generated by using table $\boldsymbol{D}$. In the second step, we further consider gate noise and generate a priority table $\boldsymbol{P}$.* Notation $p_i \in \boldsymbol{P}$ indicates the priority of gate g_i to be pruned; then, we have $p_i = \frac{C(A(g_i))}{d_i}$, where $A(g_i)$ is the qubits associated with g_i and $C(A(g_i))$ is the noise rate. Based on p_i, we formulate the mask $mask(g_i, \boldsymbol{C})^r$ for gate g_i on the given calibration data $\boldsymbol{C}$ in the r^{th} iteration as below.

$$mask(g_i, \boldsymbol{C})^r = \begin{cases} 0 & \text{if } p_i^r < threshold \\ 1 & \text{if } otherwise. \end{cases} \quad (3)$$

In the above formula, the mask equals 1 indicating that the gate g_i has a high priority, which is larger than a pre-set threshold, to be compressed. To maintain high accuracy, we will utilize the compression level T_i^{admm} if g_i is masked to be compressed.Based on these understandings, we define the indicator function $s_i(z_i)$:

$$s_i(z_i) = \begin{cases} 0 & \text{if } z_i = T_i^{admm} \times mask(g_i, \boldsymbol{C})^r \\ & \quad or\ mask(g_i, \boldsymbol{C})^r = 0 \\ +\infty & \text{if } otherwise. \end{cases} \quad (4)$$

Note that s_i is used in Eq. 2 to restrict the value of z_i. It requires its value to be 0 for a valid solution. In the above formula, we set $s_i(z_i) = 0$ in two cases: (1) if $mask(g_i, \boldsymbol{C})^r = 0$, indicating we do not require the compression on gate g_i; or (2) $\theta_i = T_i^{admm} \times mask(g_i, \boldsymbol{C})^r$, indicating that the parameter θ_i has to be compressed to be the compression level T_i^{admm}.

For each round, we will get $\boldsymbol{\theta}^r$ and $\boldsymbol{Z}^r$ alternately. At last, we will get the optimized parameters $\boldsymbol{\theta}$ to minimize Eq. 1. Then, we will employ noise injection to fine-tune parameters $\boldsymbol{\theta}$ to further improve the performance, where we will freeze the compressed parameters to not be tuned using the final $mask$.

B. Model Adapter

Given the model repository, the next question is how to use it for leveraging noise-aware compression. We provide the following guidance to use the repository efficiently.

Guidance 1: The cluster results can help to judge whether to generate new models into the model repository manager. We can get the average weighted distance:

$$\overline{(dist_{L1}^w)}_i = \frac{\sum_{\forall \mathbf{c} \in \mathbf{g_i}} dist_{L_1}(\mathbf{r_i}, \mathbf{c})}{n_i}$$

between the centroid($\mathbf{r_i}$) and all samples ($\forall \mathbf{c} \in \mathbf{g_i}$) in the i^{th} cluster, where n_i is the number of samples in the i^{th} cluster. We set $max_i(\overline{(dist_{L1}^w)}_i)$ as a threshold th_w to decide whether to add a new centroid (i.e., new representative) in the model repository. If the $min_j(dis_j) > th_w$ where dis_j is the $dist_{L1}^w$ distance between the j^{th} centroid and the current calibration data, we will add the current calibration data to the model repository.

Guidance 2: The cluster results can be utilized to predict the performance of the given model with current calibration data. Specifically, we can obtain the average accuracy of each cluster, say $\overline{acc_i}$ for the i^{th} cluster. According to users' requirement on QNN model accuracy A, we can set cluster g_i as an invalid cluster if its average accuracy $a\bar{c}c_i$ is less than A. Then, at this stage, if the current calibration matches the centroid in an invalid cluster, we will set the current calibration data as invalid data, and output a failure report.

IV. Experiments

A. Experiments Setup

Datasets and model. We evaluate our framework on three classification tasks. (1) We extract 4 class (0,1,3,6) from MNIST [16] with the former 90% samples for training and latter 200 samples for testing. To process MNIST data, we apply angle encoding [17] to encode 4×4 images to 4 qubits, and adopt 2 repeats of a VQC block (4RY +4CRY + 4RY +4RX +4CRX +4RX + 4RZ + 4CRZ +4RZ + 4CRZ) as the original model. (2) We extract 1500 samples

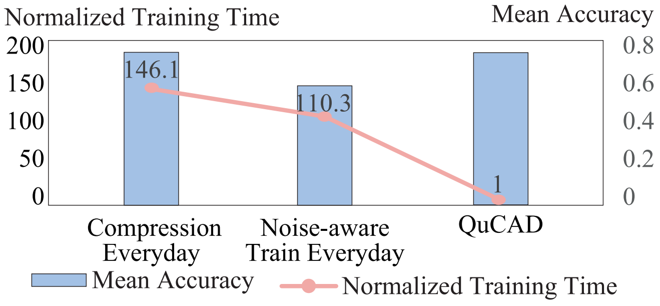


Fig. 6. The comparison of training time and accuracy.

of the earthquake detection dataset from FDSN [18]. Each sample has a positive or negative label. We utilize 90% and 10% samples for training and testing, respectively. We encode features to 4 qubits and employ the same VQC as MNIST. (3) We use Iris [19] dataset with 66.6% and 33.4% samples for training and testing. Features are encoded to 4 qubits with 3 repeats of VQC blocks.

Calibrations and Environment Settings. We pull history calibrations from Aug. 10, 2021 to Sep. 20, 2022 from IBM backend (ibm_belem) using Qiskit Interface. The front 243 days are used for optimization and the remaining 146 days' data are used for tests. We generate noise models from history calibration data and integrate them into Qiskit noise simulator for simulation. Besides, we also deploy CE-RQnn on ibm-jakarta and evaluate the output model of CE-RQnn on a real IBM quantum processor, ibm-jakarta. Our codes are based on Qiskit APIs and Torch-Quantum [20].

Competitors. We employ different approaches for comparison, including: (1) Baseline: training in a noise-free environment without optimization. (2) Noise-aware Training Once [4]: applying noise injection on the first day for training. (3) Noise-aware Training Everyday: extend the noise-aware training everyday. (4) One-time Compression[15]: applying compression with the objective of minimizing circuit length on the first day. (6) CE-RQnn: generating the models by our framework.

B. Main Results of our proposed CE-RQnn

Effective evaluation on noisy simulation. Table I reports the comparison results of different methods on MNIST, Iris, and earthquake detection datasets. Columns "Mean accuracy" and "Variance" gives the statistical information of model accuracy on the continuously 146 days. Column "Days over 0.8" stands for the number of days that the model accuracy is higher than 80%, which is similar to "Days over 0.7" and "Days over 0.5" columns. From this table, we can clearly see that our proposed CE-RQnn outperforms all competitors. Specifically, on these 3 datasets, CE-RQnn achieves improvements of 16.32%, 38.88%, and 15.36% respectively, compared with the baseline. Besides, CE-RQnn has the lowest variance except for the baseline on Iris, showing the stability of CE-RQnn. Kindly note that the mean accuracy of the baseline on Iris is much lower than CE-RQnn. We also observed that CE-RQnn outperforms competitors on the number of days over different accuracy requirements, which again demonstrates the effectiveness of our framework.

Compared with noise-aware training, even one-time compression has a significant improvement on all datasets, which validates our observation in Fig. 3. On the other hand, CE-RQnn outperforms one-time compression, indicating that adaptation is needed, showing the effectiveness of optimization in CE-RQnn.

Efficiency evaluation. We recorded the training time in Fig. 6 on 4-class MNIST. In the figure, the bars represent the mean accuracy (i.e., right axis) and each point with a number corresponds

TABLE I
PERFORMANCE COMPARISON OF DIFFERENT METHODS ON 3 DATASETS IN CONTINUS 146 DAYS WITH FLUCTUATING NOISE.

Dataset	Method	Mean Accuracy	vs. Baseline	Variance	Days over 0.8	vs. Baseline	Days over 0.7	vs. Baseline	Days over 0.5	vs. Baseline
4-class MNIST	Baseline	59.35%	0.00%	0.070	24	0	93	0	100	0
	Noise-aware Train Once [4]	58.69%	-0.65%	0.060	8	-16	92	-1	100	0
	Noise-aware Train Everyday	59.39%	0.05%	0.070	28	4	83	-10	99	-1
	One-time Compression [15]	68.44%	0.00%	0.050	80	56	102	9	117	17
	CE-RQnn (ours)	**75.67%**	**16.32%**	**0.020**	**100**	**76**	**134**	**41**	**134**	**34**
Iris	Baseline	37.85%	0.00%	**0.006**	0	0	0	0	8	0
	Noise-aware Train Once [4]	54.38%	16.53%	0.043	29	29	46	46	70	62
	Noise-aware Train Everyday	56.62%	18.78%	0.044	38	38	56	56	72	64
	One-time Compression [15]	69.20%	31.36%	0.043	84	84	90	90	103	95
	CE-RQnn (ours)	**76.73%**	**38.88%**	0.015	83	83	**108**	**108**	**141**	**133**
Seismic Wave	Baseline	68.40%	0.00%	0.014	18	0	70	0	137	0
	Noise-aware Train Once [4]	68.85%	0.45%	0.014	19	1	78	8	137	0
	Noise-aware Train Everyday	68.28%	-0.11%	0.013	22	4	69	-1	138	1
	One-time Compression [15]	78.99%	10.59%	0.007	80	62	130	60	144	7
	CE-RQnn (ours)	**83.75%**	**15.36%**	**0.001**	**133**	**115**	**146**	**76**	**146**	**9**

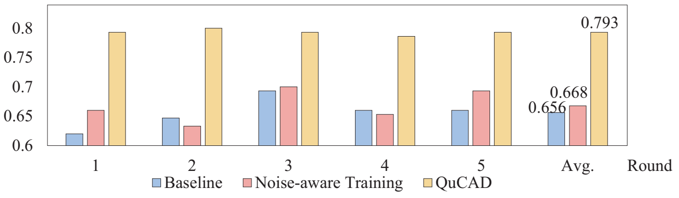


Fig. 7. On earthquake detection dataset, the performance of different approaches on the 7-qubit quantum device, ibm-jakarta.

TABLE II
COMPARISON OF DIFFERENT CLUSTER

Method	K	Mean Acc. of Clusters	Mean Acc. of Samples
K-Means with L2	6	72.94%	78.45%
Proposed K-Means with $dist_{L1}^{w}$	6	**75.83%**	**80.68%**

to a normalized training time (i.e., left axis). From the results, we can see that CE-RQnn can achieve 146× and 110.3× speedup over "compression everyday" and "noise-aware train every data", respectively. This shows the efficiency of our framework, which mainly comes from the reduction of the number of optimization. More specifically, the centroids generated can well represent the calibrations if the distribution of calibrations doesn't change much.

Evaluation on real quantum computers. We also evaluate CE-RQnn on the earthquake detection dataset using IBM quantum processor ibm-jakarta with different calibration data at different time. Results are shown in Fig. 7. CE-RQnn can consistently outperform the two competitors by 13.7% and 12.52% on average. Besides, we can see that the accuracy of CE-RQnn on different days is more stable than others, which reflects our methods can adapt QNN models to fluctuating noise.

C. Ablation Study

CE-RQnn vs. Practical Upper Bound. We further investigate the performance of 8 representative days in Fig. 8(a). We apply the result of noise-aware compression every day as a practical upper bound. From the results, we can observe that CE-RQnn has a competitive performance compared to the practical upper bound, showing CE-RQnn can get approximate optimal results.

Noise-Aware vs. Noise-Agnostic Compression. We further evaluate the proposed compression algorithm on those 8 representative days, as shown in Fig. 8(b). Results show that noise-aware compression can outperform noise-agnostic compression on most days, showing the effectiveness of our proposed noise-aware compression. We also observe both compression approaches obtained the same accuracy on 5/4 and 7/14. There are two possible reasons: (1) there

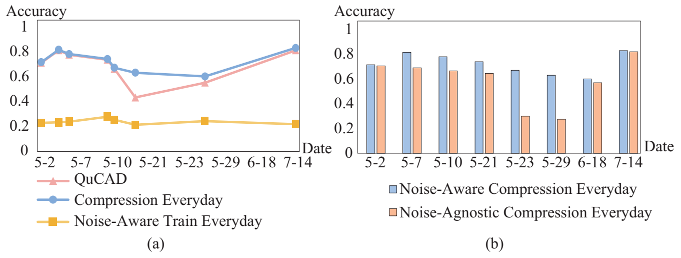


Fig. 8. Ablation Study: (a) Compression Everyday vs. CE-RQnn. (b)Noise-Aware vs. Noise-Aanostic training

is no great difference among qubits; and (2) The noise level is small and the simple compression is enough to get a good performance.

Model repository constructor. We also did an ablation study on the clustering algorithm developed in the model repository constructor. We compare our proposed noise-aware distance $dist_{L1}^{w}$ and the standard 'L2 norm' distance. Results reported in Table II show that our method can get 2.89% higher mean accuracy of 6 clusters on average and obtain 2.23% accuracy gain on that for all samples consistently. Results indicate that our proposed method can improve the quality of the centroids and make centroids represent other samples better in a cluster.

V. CONCLUSION

In this work, we reveal the fluctuating noise in quantum computing and it will significantly affect the performance of quantum applications. To battle against fluctuating noise, we further observe that the high noise level may create breakpoints in the loss surface, and in turn, the noise-aware training may find inferior solutions. By investigating the breakpoints, we observe that quantum neural network compression can be a hammer to the noise issue. And we build a compression-aided framework, namely CE-RQnn, which can automatically adapt a given model to fluctuating quantum noise. Evaluations on MNIST, earthquake detection dataset, and Iris show the effectiveness and efficiency of CE-RQnn. specifically, CE-RQnn can obtain stable performance on the earthquake detection task using a real IBM quantum processor.

REFERENCES

[1] R. Takagi, S. Endo, S. Minagawa, and M. Gu, "Fundamental limits of quantum error mitigation," *npj Quantum Information*, vol. 8, no. 1, pp. 1–11, 2022.
[2] Y. Ji, S. Brandhofer, and I. Polian, "Calibration-aware transpilation for variational quantum optimization," *arXiv preprint arXiv:2207.08205*, 2022.
[3] D. Bhattacharjee, A. A. Saki, M. Alam, A. Chattopadhyay, and S. Ghosh, "Muqut: Multi-constraint quantum circuit mapping on nisq computers," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–7, IEEE, 2019.

[4] H. Wang, J. Gu, Y. Ding, Z. Li, F. T. Chong, D. Z. Pan, and S. Han, "Quantumnat: quantum noise-aware training with noise injection, quantization and normalization," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pp. 1–6, 2022.

[5] M. Sajjan, J. Li, R. Selvarajan, S. H. Sureshbabu, S. S. Kale, R. Gupta, V. Singh, and S. Kais, "Quantum machine learning for chemistry and physics," *Chemical Society Reviews*, 2022.

[6] D. Maheshwari, B. Garcia-Zapirain, and D. Sierra-Sosa, "Quantum machine learning applications in the biomedical domain: A systematic review," *Ieee Access*, 2022.

[7] R. Orús, S. Mugel, and E. Lizaso, "Quantum computing for finance: Overview and prospects," *Reviews in Physics*, vol. 4, p. 100028, 2019.

[8] Y. Liu, S. Arunachalam, and K. Temme, "A rigorous and robust quantum speed-up in supervised machine learning," *Nature Physics*, vol. 17, no. 9, pp. 1013–1017, 2021.

[9] T. Giurgica-Tiron, Y. Hindy, R. LaRose, A. Mari, and W. J. Zeng, "Digital zero noise extrapolation for quantum error mitigation," in *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pp. 306–316, IEEE, 2020.

[10] S. Endo, S. C. Benjamin, and Y. Li, "Practical quantum error mitigation for near-future applications," *Physical Review X*, vol. 8, no. 3, p. 031027, 2018.

[11] W. J. Huggins, S. McArdle, T. E. O'Brien, J. Lee, N. C. Rubin, S. Boixo, K. B. Whaley, R. Babbush, and J. R. McClean, "Virtual distillation for quantum error mitigation," *Physical Review X*, vol. 11, no. 4, p. 041036, 2021.

[12] S. Dasgupta and T. S. Humble, "Stability of noisy quantum computing devices," *arXiv preprint arXiv:2105.09472*, 2021.

[13] S. Dasgupta and T. S. Humble, "Characterizing the reproducibility of noisy quantum circuits," *Entropy*, vol. 24, no. 2, p. 244, 2022.

[14] S. Dasgupta and T. S. Humble, "Assessing the stability of noisy quantum computation," *arXiv preprint arXiv:2208.07219*, 2022.

[15] Z. Hu, P. Dong, Z. Wang, Y. Lin, Y. Wang, and W. Jiang, "Quantum neural network compression," *arXiv preprint arXiv:2207.01578*, 2022.

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[17] R. LaRose and B. Coyle, "Robust data encodings for quantum classifiers," *Physical Review A*, vol. 102, no. 3, p. 032420, 2020.

[18] "International federation of digital seismograph networks." https://www.fdsn.org/webservices/.

[19] P. S. Hoey, "Statistical analysis of the iris flower dataset," *University of Massachusetts At Lowell, Massachusetts*, 2004.

[20] H. Wang, Y. Ding, J. Gu, Z. Li, Y. Lin, D. Z. Pan, F. T. Chong, and S. Han, "Quantumnas: Noise-adaptive search for robust quantum circuits," in *The 28th IEEE International Symposium on High-Performance Computer Architecture (HPCA-28)*, 2022.