Quantum readout error mitigation via deep learning

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

Deutsche Physikalische Gesellschaft DPG
IOP Institute of Physics

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

OPEN ACCESS

RECEIVED
30 January 2022

REVISED
20 June 2022

ACCEPTED FOR PUBLICATION
22 June 2022

PUBLISHED
8 July 2022

PAPER

Quantum readout error mitigation via deep learning

Jihye Kim[1], Byungdu Oh[1], Yonuk Chong[1,*], Euyheon Hwang[1,*] and Daniel K Park[2,3,*]

[1] SKKU Advanced Institute of Nanotechnology and Department of Nano Engineering, Suwon, 16419, Republic of Korea
[2] Department of Applied Statistics, Yonsei University, Seoul, 03722, Republic of Korea
[3] Department of Statistics and Data Science, Yonsei University, Seoul, 03722, Republic of Korea
[*] Authors to whom any correspondence should be addressed.

E-mail: yonuk@skku.edu, euyheon@skku.edu and dkd.park@yonsei.ac.kr

Keywords: quantum computing, error mitigation, quantum control, deep learning

Supplementary material for this article is available online

Abstract

Quantum computing devices are inevitably subject to errors. To leverage quantum technologies for computational benefits in practical applications, quantum algorithms and protocols must be implemented reliably under noise and imperfections. Since noise and imperfections limit the size of quantum circuits that can be realized on a quantum device, developing quantum error mitigation techniques that do not require extra qubits and gates is of critical importance. In this work, we present a deep learning-based protocol for reducing readout errors on quantum hardware. Our technique is based on training an artificial neural network (NN) with the measurement results obtained from experiments with simple quantum circuits consisting of singe-qubit gates only. With the NN and deep learning, non-linear noise can be corrected, which is not possible with the existing linear inversion methods. The advantage of our method against the existing methods is demonstrated through quantum readout error mitigation experiments performed on IBM five-qubit quantum devices.

1. Introduction

The theory of quantum computation opens up tremendous opportunities as it promises drastic computational benefits for solving commercially relevant problems [1–5]. One of the major technological hurdles against practical quantum advantage is the unavoidable noise and imperfections. Although the theory of quantum error correction and fault-tolerance guarantees that imperfections are not the fundamental objections to quantum computation [6–10], the size of quantum circuits needed to realize the theory is beyond the reach of the near-term technology. Consequently, in near-term quantum computing, all physical qubits are expected to be operating as logical qubits, leading to the noisy intermediate-scale quantum (NISQ) era [5]. Therefore, to bridge the gap between theoretical results and experimental capabilities, developing algorithmic means at the software level to reduce quantum computational errors without increasing the quantum resource overhead, such as the number of qubits and gates, is desired [11–15].

This work addresses the problem of reducing the readout errors that occur during the final step of the quantum computation. We propose a machine learning-based protocol to reduce the quantum readout error. Our method trains an artificial neural network (NN) with real measurement outcomes from quantum circuits with known final states as input and ideal measurement outcomes of the same circuit as output. After training, the NN is used to infer the ideal measurement outcomes from real measurement outcomes of arbitrary quantum computation. Since our quantum readout error mitigation (QREM) protocol only relies on training a classical NN with known measurement outcomes, the extra quantum resource arises only for gathering noisy measurement results. This is done with a random state prepared by applying an arbitrary single-qubit gate to the default initial state of qubits that are subject to measurement

in a quantum algorithm. Thus our method does not increase the number of qubits and gates beyond what is needed by the quantum algorithm itself.

Using the NN and deep learning, non-linear readout errors can be corrected. This feature places our QREM method above the predominantly used methods based on linear inversion (LI) that do not take non-linear effects into account. Moreover, while the LI method often produces non-physical results and requires post-optimization [16, 17], our method always produces physically valid results. We compare our QREM with the LI method through experiments with superconducting quantum devices available on the IBM cloud platform for two to five qubits. The experimental results demonstrate that the amount of readout error reduced by our method exceeds that of the previous method in all instances differentiated by the number of qubits and the error quantifier selected from mean squared error (MSE), Kullback–Leibler divergence (KLD) and infidelity (IF).

The remainder of the paper is organized as follows. Section 2 describes the theoretical framework of this work, such as the quantum readout error, existing mitigation methods for it, and the underlying idea of our NN-based approach to the problem. Section 3 explains the machine learning algorithm used in this work for the QREM. Section 4 presents the proof-of-principle experiment performed with three five-qubit quantum computers available on the IBM quantum cloud platform. Conclusions and directions for future work are provided in section 5.

2. Theoretical framework

Quantum computation in the circuit-based framework can be thought of as a three-step procedure consisting of input state initialization, unitary transformation, and readout. In experiments, the readout step is usually implemented as the projective measurement in the computational basis, which collapses an n-qubit final state $|\psi_j\rangle \in \mathbb{C}^{2^n}$ to $|i\rangle$ with the probability $p(i|j) = |\langle i|\psi_j\rangle|^2$. Without loss of generality, we denote $p(i) := p(i|j)$. In the ideal (error free) case, the final result of a quantum algorithm is determined by the probability distribution of the measurement outcomes, denoted by

$$\boldsymbol{p} = \{p(0), p(1), \ldots, p(2^n - 1)\}.$$

Equivalently, $\boldsymbol{p}$ is the set of diagonal elements of the final density matrix ρ produced at the end of the quantum computation. Many quantum algorithms are designed in a way that the solution to the problem is encoded in the probability distribution $\boldsymbol{p}$. In particular, estimating an expectation value of some observable from such probability distribution is central to many NISQ algorithms, such as variational quantum eigensolver [18–21], quantum approximate optimization algorithm [22], quantum machine learning [23–25], and simulation of stochastic processes [26, 27].

However, due to readout errors, the observed probability distribution can deviate from the ideal probability distribution. Hereinafter, we denote the probability to observe $|i\rangle$ in the experiment as $\hat{p}(i)$ and the error map as $\mathcal{N}$ that transforms the ideal probability vector to an observed one as $\hat{\boldsymbol{p}} = \mathcal{N}(\boldsymbol{p})$. Since $\mathcal{N}$ describes the transformation of a probability distribution, it must preserve the l_1-norm. The goal of QREM is to minimize the loss function

$$D\big(\boldsymbol{p}, \mathcal{N}(\boldsymbol{p})\big), \tag{1}$$

defined by some distance measure D for probability distributions. The goal of QREM is conceptually depicted at the top of figure 1, with an example of a two-qubit case given in the bottom.

Since QREM is of critical importance especially for NISQ computing, several protocols have been proposed recently to address this issue. In essence, these methods assume that the noise function is linear and relies on solving the linear equation

$$\hat{\boldsymbol{p}} = \Lambda \boldsymbol{p}, \tag{2}$$

where the linear response matrix Λ is estimated by some tomographic means [16, 17, 20, 28–32]. This usually demands $O(2^n)$ measurements with an assumption that the computational basis state preparation error is negligible compared to the readout error. The number of experiments can be reduced with certain assumptions about the noise model, such as those dominated by few-qubit correlations [30, 31, 33] and by independent single-qubit Pauli errors acting on the final state during the finite time between the last gate and the detection event [34]. After Λ is found, correct measurement outcomes are predicted by calculating $\Lambda^{-1}\hat{\boldsymbol{p}}$. We refer to this technique as LI based QREM (LI-QREM). This approach often requires extra classical post-processing [16, 17, 31], since the inversion may produce a vector that is not a probability distribution. In general, the readout noise is not static. Hence the QREM procedure described above needs to be frequently implemented as a part of the calibration routine of the given experimental setup.

While this work shares the same goal as the previous methods, we leverage classical deep learning techniques to approximate $\mathcal{N}^{-1}(\boldsymbol{p})$. Namely, we train a NN denoted by $\mathcal{F}$ which describes the map

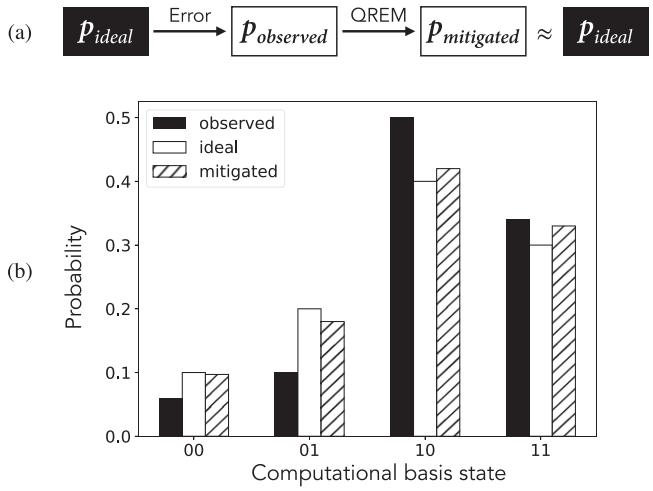


Figure 1. An illustration of the QREM procedure. In the noiseless case, the underlying quantum algorithm should generate a probability distribution $\boldsymbol{p}_{\text{ideal}}$ upon the measurement in the computational basis. However, due to imperfections and noise, one obtains the noisy distribution $\boldsymbol{p}_{\text{observed}}$, which deviates from the true distribution. The QREM method constructs a probability distribution $\boldsymbol{p}_{\text{mitigated}}$ based on $\boldsymbol{p}_{\text{observed}}$ that is as close to $\boldsymbol{p}_{\text{ideal}}$ as possible. The bar graph in (b) is an artificially generated two-qubit example that illustrates the role of QREM. After a QREM algorithm, $\boldsymbol{p}_{\text{mitigated}}$ is a more accurate description of the true probability distribution than the raw data.

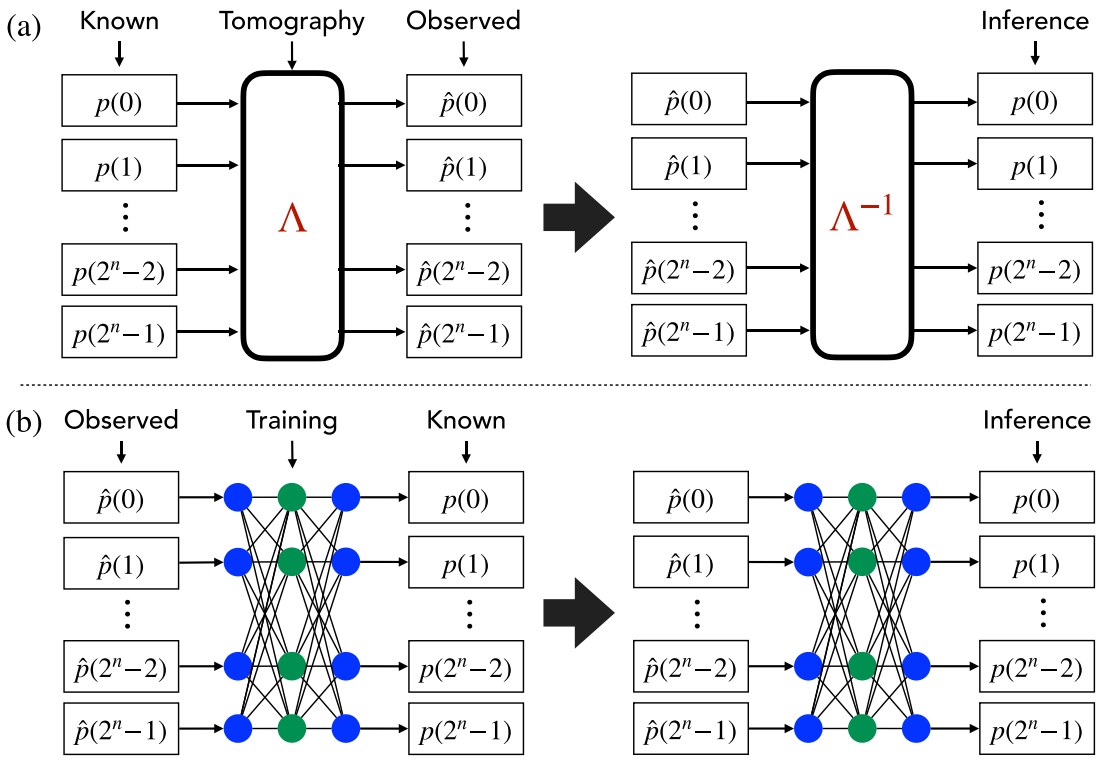


Figure 2. QREM based on (a) linear error model and readout tomography (LI-QREM) (b) deep learning (NN-QREM). The deep learning model is trained to implement the function $\mathcal{F}$, which serves as an approximation to the inverse of the true error map, i.e. $\mathcal{F} \approx \mathcal{N}^{-1}$.

$\boldsymbol{p} = \mathcal{F}(\hat{\boldsymbol{p}})$ such that $\mathcal{F} \approx \mathcal{N}^{-1}$. Since the deep learning model can take care of spurious non-linear effects, the amount of error suppression is expected to be beyond what the linear model can achieve. Moreover, by using the softmax function in the final layer of a NN, one can ensure that the output always represents a probability distribution. Figure 2 compares the basic idea of previous LI-QREM and the NN based method proposed in this work, which we refer to as NN-QREM. The following section explains the machine learning model in detail.

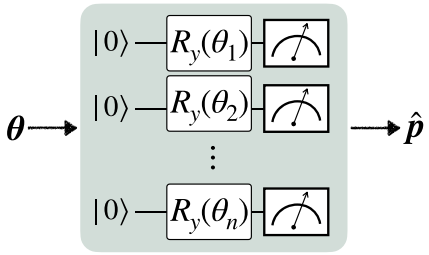


Figure 3. The quantum circuit for generating the noisy probability distribution $\hat{\boldsymbol{p}}$ used as an input to the NN during training. The training data is generated by applying an arbitrary single-qubit rotation $R_y(\theta)$ to each qubit and measuring qubits in the computational basis. Training of the NN also requires the ideal probability distribution $\boldsymbol{p}$, and this is calculated using the set of rotation angles $\boldsymbol{\theta}$ used in the quantum circuit.

3. The machine learning algorithm

3.1. Data collection for training

In order to train a deep NN for QREM, we need to have both $\boldsymbol{p}$ and $\hat{\boldsymbol{p}}$, meaning that qubits need to be prepared in some known state before measurement. Since single-qubit gate errors are usually negligible when compared to those of two-qubit gates and measurement in modern quantum devices [16, 29, 35], we prepare the training set of quantum states using single-qubit gates only. In particular, the training set is constructed by applying $R_y(\theta)$ gate, which corresponds to the rotation around the y-axis of the Bloch sphere, to all qubits in the system with randomly and independently generated angle $\theta \in [0, 2\pi)$. Since the measurement is performed in the computational basis, which is the σ_z basis by convention, there is no need to apply the R_z gate. Hence the quantum circuit depth for generating training data is one. Note that calibration of single-qubit rotation gates can be done independently of measurement errors [36]. Figure 3 represents the quantum circuit that generates $\hat{\boldsymbol{p}}$ as an input to the NN for training. The ideal probability distribution $\boldsymbol{p}$, which is inserted as the output of the NN during training, can easily be calculated from the rotation angles. For an n-qubit system, the probability to measure a computational basis state $b \in \{0, 1\}^n$ (i.e. an n-bit string) is

$$p(b) = \left| \prod_i^n \cos^{1-b_i}(\theta_i/2) \sin^{b_i}(\theta_i/2) \right|^2,$$

where b_i is the ith bit of the binary string b.

3.2. Model construction

The deep learning model is constructed with an input layer, hidden layers, and an output layer (see figure 2(b)). Input layer and output layer have 2^n nodes, whose numerical values represent the probability of measuring the computational basis states in the actual experiment and the ideal case, respectively. All hidden layers are fully connected layers, and each hidden node employs the rectified linear unit (ReLU) as the activation function. The output layer uses the softmax function for activation, which ensures that the output represents a probability distribution. The loss function for optimizing the weights and biases of the NN is the categorical cross entropy, which is normally used in multi-label classification problems. The free parameters are updated by the Adam optimizer [37] whose hyperparameters, such as the learning rate, are empirically chosen (see section 4.1).

3.3. Inference

The trained NN represents the function $\mathcal{F} \approx \mathcal{N}^{-1}$. Thus, the error-mitigated probability distribution, which is denoted by $\tilde{\boldsymbol{p}}$, is obtained by inserting $\hat{\boldsymbol{p}}$ from an experiment of interest as the input to the trained NN. The inference can be expressed as $\tilde{\boldsymbol{p}} = \mathcal{F}(\hat{\boldsymbol{p}}) \approx \mathcal{N}^{-1}(\hat{\boldsymbol{p}})$.

4. Experiment

4.1. Experimental setup

Two QREM techniques, namely LI-QREM and NN-QREM were tested on three different five-qubit quantum computers available on IBM quantum experience. Training of the NN and inference are carried out as described in section 3 using Keras library of Python. The LI-QREM results are obtained by using the default readout error mitigation package in Qiskit Ignis [38] (i.e. `CompleteMeasFitter`). We use the

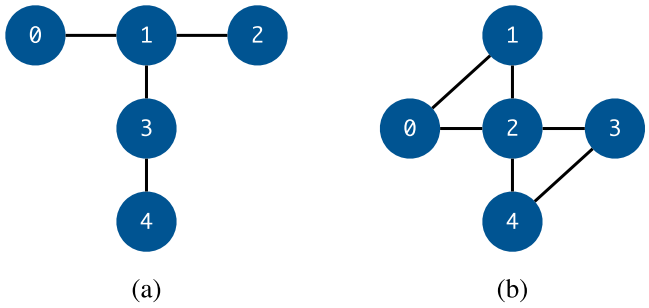


Figure 4. Coupling map (qubit connectivity) of the quantum devices used in this work. `ibmq_quito` and `ibmq_belem` have the arrangement shown in (a), and `ibmq_qx2` has the arrangement shown in (b).

Table 1. The hyperparameters and training details of the neural networks used in the NN-QREM experiments with two to five qubit in case of `ibmq_belem` and `ibmq_quito`. The number of training data indicates the size of the entire training dataset.

Num. of qubits	2	3	4	5
Num. of training data	1175	3472	9700	9700
Num. of test data	200	200	200	200
Num. of hidden layer	7	4	8	5
Num. of nodes in each hidden layer	20	40	80	160
Num. of parameters	2700	5600	48 000	113 440
Num. of epochs	300	300	300	300
Learning rate	0.001	0.001	5×10^{-5}	5×10^{-5}

least squares method to calibrate the resulting matrix from the Qiskit library to ensure physical results.

On the five-qubit devices, QREM of two to five qubits are carried out. The quantum devices with smaller quantum volume were chosen in this study since the error mitigation is more critical for such devices. Furthermore, we chose devices of different qubit connectivity as shown in figure 4. For the first arrangement (figure 4(a)), two quantum devices were selected based on the amount of queue on the cloud service, namely `ibmq_quito` and `ibmq_belem`, to minimize the experimental time. The two- and three-qubit QREM were performed on `ibmq_quito`, and the four- and five-qubit QREM experiments were performed on `ibmq_belem`. For the second arrangement, `ibmq_qx2` was the only device available for us to use. Thus for this device, all two- to five-qubit experiments were implemented. A typical set of device properties and calibration data for `ibmq_quito`, `ibmq_belem`, and `ibmq_qx2` are provided in the supplementary information (https://stacks.iop.org/NJP/24/073009/mmedia).

Several hyperparameters need to be fixed when training a NN. In our experiments, we set the number of nodes in each hidden layer to be 5×2^n to ensure that the number of nodes scales only linearly with the size of the probability distribution. The learning rate for the Adam optimization algorithm was fine-tuned for each number of qubits. The number of hidden layers was optimized using five-fold cross-validation (see supplementary information). The number of epochs was fixed to 300 and the convergence was sufficient within that window. The hyperparameter values of the NNs used in the NN-QREM experiments are provided in tables 1 and 2 for the two device types shown in figure 4, respectively. The training data size is the number of data collected through the cloud access for a fixed amount of time and was not subject to optimization. In the supplementary information, we show that NN-QREM performs better than LI-QREM even when the number of training data is reduced by a factor of two.

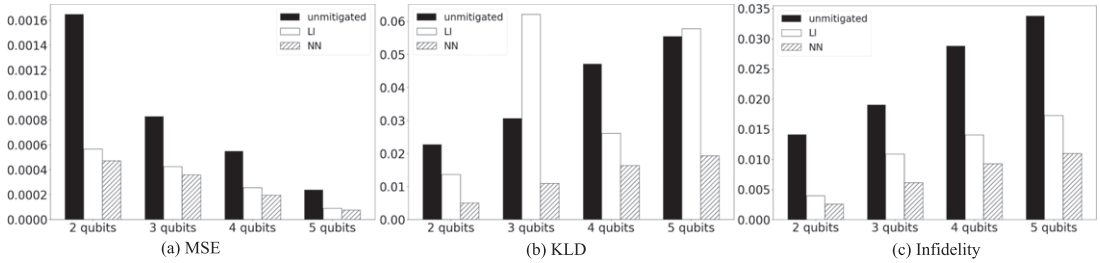
The probability distribution $\hat{\boldsymbol{p}}$ is obtained by counting the number of n-bit binary string obtained from the measurement and dividing it by the number of shots (i.e. repetitions) taken for gathering the statistics. The number of the shots is chosen to be 8192, which is much larger than 2^5, the largest size of the sample space of the probability distribution.

4.2. Loss functions

In this work, we evaluate three different measures (see equation (1)) to quantify the amount of readout error mitigation and to compare the performance of different QREM methods. The metrics used for

Table 2. The hyperparameters and training details of the neural networks used in the NN-QREM experiments with two to five qubit in case of `ibmq_qx2`. The number of training data indicates the size of the entire training dataset.

Num. of qubits	2	3	4	5
Num. of training data	1800	3800	7800	9850
Num. of test data	200	200	200	200
Num. of hidden layer	5	2	7	5
Num. of nodes in each hidden layer	20	40	80	160
Num. of parameters	1860	2320	41 520	113 440
Num. of epochs	300	300	300	300
Learning rate	0.001	0.001	5×10^{-5}	5×10^{-5}



(a) MSE (b) KLD (c) Infidelity

Figure 5. Experimental QREM results for reducing (a) MSE, (b) KLD and (c) infidelity from `ibmq_quito` and `ibmq_belem`. Filled bars represent unmitigated results, unfilled bars represent the LI-QREM results, and the hatched bars represent NN-QREM results.

comparison are MSE

$$D_{\mathrm{MSE}} = \frac{1}{2^n}\sum_{i=0}^{2^n-1} |\tilde{\boldsymbol{p}}_i - \boldsymbol{p}_i|^2,$$

Kullback–Leibler divergence (KLD)

$$D_{\mathrm{KLD}} = \sum_{i=0}^{2^n-1} \boldsymbol{p}_i \log \frac{\boldsymbol{p}_i}{\tilde{\boldsymbol{p}}_i},$$

and IF

$$D_{\mathrm{IF}} = 1 - \left(\sum_{i=0}^{2^n-1} \sqrt{\boldsymbol{p}_i \tilde{\boldsymbol{p}}_i}\right)^2,$$

where $\boldsymbol{p}_i$ and $\tilde{\boldsymbol{p}}_i$ are the ith elements of the ideal and the mitigated probability distributions, respectively. For all these measures, a smaller value indicates a better performance as they represent the dissimilarity. Moreover, the IF is equivalent to the quantum state IF of two diagonal density matrices.

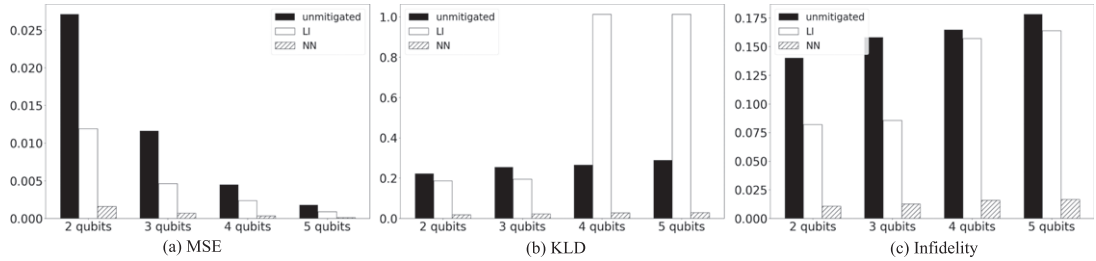
4.3. Results

This section reports MSE, KLD and IF of (1) the raw (noisy) probability distribution $\hat{\boldsymbol{p}}$ (2) LI-QREM results $\tilde{\boldsymbol{p}}_{\mathrm{LI}}$ and (3) NN-QREM results $\tilde{\boldsymbol{p}}_{\mathrm{NN}}$ that are averaged over 200 test data. The QREM results are presented in figures 5 and 6 for the device type (a) and (b), respectively. The results clearly show that the NN-QREM can reduce the readout noise more effectively than the LI-QREM. Interestingly, LI-QREM is unable to reduce KLD in some cases, while NN-QREM reduces all quantifiers of the error in all cases.

To quantitatively compare the two methods, we define the performance improvement ratio for each loss function D_i with the subscript i being MSE, KLD, or IF, as

$$R_i = \frac{D_i^{\mathrm{LI}} - D_i^{\mathrm{NN}}}{D_i^{\mathrm{NN}}} \times 100(\%), \tag{3}$$

where the superscript indicates whether the result is from LI-QREM or NN-QREM. By the definition of R_i, $R_i > 0$ indicates that NN-QREM is better than LI-QREM, and vice versa. Table 3 shows the performance improvement ratio for all metrics. The values in the table show that NN-QREM outperforms LI-QREM in all instances. A notable observation is that NN-QREM works particularly better for the device type (b). We speculate that the readout noise in device (b) has higher non-linearities than that of device (a), albeit rigorous device-dependence analysis is left our for future work.



(a) MSE (b) KLD (c) Infidelity

Figure 6. Experimental QREM results for reducing (a) MSE, (b) KLD and (c) infidelity from `ibmq_qx2`. Filled bars represent unmitigated results, unfilled bars represent the LI-QREM results, and the hatched bars represent NN-QREM results.

Table 3. Performance improvement ratio R_i for all metrics tested in this work (i.e. MSE, KLD and IF) for each number of qubits. These numbers are categorized under two different device structures shown in figure 4, labeled as type (a) and type (b).

	Num. of qubits	2	3	4	5
Device type (a)	R_{MSE}	19.9	18.4	30.4	17.6
	R_{KLD}	174	466	59.8	200
	R_{IF}	52.5	76.6	51.2	57.2
Device type (b)	R_{MSE}	648	564	611	592
	R_{KLD}	888	775	3469	3305
	R_{IF}	657	571	892	890

Table 4. Performance improvement ratio R_i for three metrics (i.e. MSE, KLD and IF) for mitigating the readout error of `ibmq_belem` using the datasets acquired from `ibmq_quito`, and vice versa.

	Num. of qubits	2	3	4	5
`ibmq_quito` to `ibmq_belem`	R_{MSE}	6.55	−10.8	−29.1	−32.1
	R_{KLD}	379	470	78.9	11.6
	R_{IF}	82.5	71.4	4.15	−12.9
`ibmq_belem` to `ibmq_quito`	R_{MSE}	1.58	3.78	−10.1	−24.1
	R_{KLD}	209	141	114	80.5
	R_{IF}	25.2	23.7	0.19	−5.13

While the main purpose of our experiment is to demonstrate that the NN-based method is capable of mitigating the readout error beyond the existing LI based methods, we point out that the NNs used in this work can be further improved. For instance, one can choose to optimize the number of nodes in each hidden layer, instead of using some fixed number. One may also choose to use different activation functions and a different optimization algorithm, such as the Nesterov moment optimizer [39]. One may also explore various loss functions for optimization. For example, one can choose the MSE, KLD or IF as the loss function to minimize through training, instead of the cross entropy loss used in this work.

Another interesting question is whether the NN trained based on the data generated from one quantum device can be used to mitigate the noise of other quantum devices. This method is expected to work well, especially for devices with similar hardware characteristics. We tested this idea on `ibmq_quito` and `ibmq_belem` since they have similar hardware designs with the same qubit connectivity. In particular, we applied the NN trained with the datasets generated by `ibmq_quito` to mitigate the readout error in the dataset of `ibmq_belem`, and vice versa. The corresponding QREM results are presented in table 4. The experimental results show that the NN-QREM is better than the LI-QREM in most cases, especially for KLD and IF. For MSE, the NN-QREM performed better only for smaller numbers of qubits.

In general, the readout noise is not static. Hence QREM must be carried out frequently as a part of the experimental calibration routine. Since both LI- and NN-QREM procedures can be costly, it is desirable to design QREM to be robust to the drift. We tested how long the linear error matrix and the NN determined at one time point continue to produce useful QREM results over time. More specifically, we performed five-qubit QREM on `ibmq_belem` once per day over eleven days using a pre-determined error matrix and a pre-trained NN provided from earlier times. Both QREM methods reduced all loss functions D_i with the subscript i being MSE, KLD, or IF for all eleven days, while NN-QREM always performed better than LI-QREM. More details are provided in supplementary information.

5. Conclusion and discussion

This work proposes QREM protocol based on deep learning with NNs. The deep learning is known to be useful for finding non-linear relationships in a given dataset. This feature is utilized in our work to mitigate non-linear effects in the readout error. Hence our approach can achieve the level of error mitigation that is not attainable with the previous methods that rely on the linear error model. The advantage was clearly demonstrated through proof-of-principle experiments with two to five superconducting qubits. In all instances of experiments performed in this work, the NN based QREM (NN-QREM) outperformed the linear inversion based QREM (LI-QREM) in reducing the error, which is quantified with three different measures: MSE, KLD and IF. The improvement in error suppression is particularly more significant when KLD and IF are considered. We also tested whether the data collected from one quantum device can be used to mitigate the readout error of the other devices. For example, we performed NN-QREM and LI-QREM to mitigate the readout error of `ibmq_belem` using the data obtained from `ibmq_quito`, and vice versa. Both methods can successfully mitigate the readout error when MSE and IF are considered, while only NN-QREM can reduce KLD. Moreover, NN-QREM performed better than LI-QREM for this task in most cases.

Discussions on open problems and directions for future work are provided as follows. An important challenge in the domain of quantum error mitigation is the scalability of the algorithm, i.e. the total computational cost should not grow superpolynomially with the number of qubits. In our current construction, even if the number of training data is restricted to grow as $O(\mathrm{poly}(n))$, the number of input and output nodes is 2^n. This is because our work considers the general problem of inferring the probability distribution of a multivariate random variable whose dimension grows exponentially with the number of qubits. Let us consider a NN consisting of five layers as an example. Then NN-QREM for 16 qubits requires 472 billion parameters to be optimized. This is comparable to the large-scale generative language model Megatron-Turing NLG (MT-NLG) [40], which uses 530 billion parameters. An open question is whether and how classical computing architectures, such as those related to the efficient use of memories and graphics processing units, can be improved to break the 16-qubit limit.

The number of input and output nodes, and hence the size of the NN, can be reduced significantly under certain conditions and assumptions. For example, if the goal is to mitigate the readout error when estimating the expectation value of an observable, the NN-QREM scheme can be constructed in a scalable way as long as the number of Pauli operators that describe the observable does not increase superpolynomially with the number of qubits. The size of the NN can be reduced when the correlation between the measurements of individual qubits is negligible and such information is provided *a priori*. In this case, the joint probability distribution of the random variable representing the measurement outcomes is described as a product of independent probability mass functions. Thus the number of input and output nodes for n qubits with independent measurement errors grows linearly with n. Another promising strategy for reducing the computational cost is transfer learning [41–44]. Transfer learning aims to improve the training of a new machine learning model by utilizing a reference machine learning model that is pre-trained for a different but related task with a different dataset. Indeed, the potential benefit of transfer learning was demonstrated when we used the NN trained with the data from `ibmq_quito` to mitigate errors in the dataset of `ibmq_belem` and vice versa. Implementing the NN-QREM by fine-tuning the NN pre-trained with a reference quantum hardware can significantly reduce the number of layers to be trained. Transfer learning can also be useful when system parameters of quantum processors drift over time. In this case, utilizing transfer learning to re-use a NN trained at a reference time point may significantly reduce the overall computational cost. Detailed analysis on how much the transfer learning can reduce the overall computational cost while maintaining the good QREM performance is left as future work. Although transfer learning has the potential to reduce the number of deep learning layers subject to training, the number of input and output nodes remains to grow exponentially with the number of qubits. Note that the probability vector constructed by sampling from a quantum circuit is sparse as long as the number of repetitions does not grow exponentially with n. Based on this observation, we plan to investigate machine learning methods for sparse data to explore the possibility of designing a scalable QREM algorithm.

Another interesting future work is to develop a machine learning-based QREM protocol for the single-shot settings, which is important for many applications such as quantum teleportation, quantum error correction, and quantum communication.

After completion of the present manuscript, we became aware of a related work by Lienhard *et al* [45], which improved the qubit-state-assignment fidelity by applying deep NNs as nonlinear filters directly to digitized signals in a superconducting qubit system. Interesting future work is to compare whether it is more advantageous to train a NN with noisy signals observed from physical qubits or with noisy

qubit-state-assignment. The latter is more relevant to the cloud-based quantum computing environment. One could also investigate whether the latter can be applied in addition to the former to provide further improvement.

Acknowledgments

This research is supported by the National Research Foundation of Korea (Grant No. 2019R1I1A1A01050161, No. 2021M3H3A1038085 and No. 2022M3E4A1074591). We acknowledge the use of IBM Quantum services for this work. The views expressed are those of the authors, and do not reflect the official policy or position of IBM or the IBM Quantum team.

Data availability statement

The data that support the findings of this study are available upon reasonable request from the authors.

ORCID iDs

Byungdu Oh https://orcid.org/0000-0002-2709-3177
Daniel K Park https://orcid.org/0000-0002-3177-4143

References

[1] Lloyd S 1996 Universal quantum simulators *Science* **273** 1073–8
[2] Zalka C 1998 Simulating quantum systems on a quantum computer *Proc. R. Soc. London* A **454** 313–22
[3] Shor P W 1999 Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer *SIAM Rev.* **41** 303–32
[4] Harrow A W, Hassidim A and Lloyd S 2009 Quantum algorithm for linear systems of equations *Phys. Rev. Lett.* **103** 150502
[5] Preskill J 2018 Quantum computing in the NISQ era and beyond *Quantum* **2** 79
[6] Shor P W 1995 Scheme for reducing decoherence in quantum computer memory *Phys. Rev.* A **52** R2493
[7] Steane A M 1996 Error correcting codes in quantum theory *Phys. Rev. Lett.* **77** 793
[8] Calderbank A R and Shor P W 1996 Good quantum error-correcting codes exist *Phys. Rev.* A **54** 1098
[9] Fowler A G, Mariantoni M, Martinis J M and Cleland A N 2012 Surface codes: towards practical large-scale quantum computation *Phys. Rev.* A **86** 032324
[10] Terhal B M 2015 Quantum error correction for quantum memories *Rev. Mod. Phys.* **87** 307–46
[11] Feng G, Wallman J J, Buonacorsi B, Cho F H, Park D K, Xin T, Lu D, Baugh J and Laflamme R 2016 Estimating the coherence of noise in quantum control of a solid-state qubit *Phys. Rev. Lett.* **117** 260501
[12] Song C, Cui J, Wang H, Hao J, Feng H and Li Y 2019 Quantum computation with universal error mitigation on a superconducting quantum processor *Sci. Adv.* **5** eaaw5686
[13] Endo S, Benjamin S C and Li Y 2018 Practical quantum error mitigation for near-future applications *Phys. Rev.* X **8** 031027
[14] Temme K, Bravyi S and Gambetta J M 2017 Error mitigation for short-depth quantum circuits *Phys. Rev. Lett.* **119** 180509
[15] Kim C, Park K D and Rhee J-K 2020 Quantum error mitigation with artificial neural network *IEEE Access* **8** 188853–60
[16] Maciejewski F B, Zimborás Z and Oszmaniec M 2020 Mitigation of readout noise in near-term quantum devices by classical post-processing based on detector tomography *Quantum* **4** 257
[17] Nachman B, Urbanek M, de Jong W A and Bauer C W 2020 Unfolding quantum computer readout noise *npj Quantum Inf.* **6** 84
[18] Peruzzo A, McClean J, Shadbolt P, Yung M-H, Zhou X-Q, Love P J, Aspuru-Guzik A and O'Brien J L 2014 A variational eigenvalue solver on a photonic quantum processor *Nat. Commun.* **5** 4213
[19] McClean J R, Romero J, Babbush R and Aspuru-Guzik A 2016 The theory of variational hybrid quantum-classical algorithms *New J. Phys.* **18** 023023
[20] Kandala A, Mezzacapo A, Temme K, Takita M, Brink M, Chow J M and Gambetta J M 2017 Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets *Nature* **549** 242–6
[21] Barron G S and Wood C J 2020 Measurement error mitigation for variational quantum algorithms (arXiv:2010.08520)
[22] Farhi E, Goldstone J and Gutmann S 2014 A quantum approximate optimization algorithm (arXiv:1411.4028)
[23] Havlícek V, Córcoles A D, Temme K, Harrow A W, Kandala A, Chow J M and Gambetta J M 2019 Supervised learning with quantum-enhanced feature spaces *Nature* **567** 209–12
[24] Blank C, Park D K, Rhee J-K K and Petruccione F 2020 Quantum classifier with tailored quantum kernel *npj Quantum Inf.* **6** 41
[25] Park D K, Blank C and Petruccione F 2020 The theory of the quantum kernel-based binary classifier *Phys. Lett.* A **384** 126422
[26] Blank C, Park D K and Petruccione F 2021 Quantum-enhanced analysis of discrete stochastic processes *npj Quantum Inf.* **7** 126
[27] Ho M, Takagi R and Gu M 2021 Enhancing quantum models of stochastic processes with error mitigation (arXiv:2105.06448)
[28] Asfaw A *et al* 2020 *Learn Quantum Computation Using Qiskit* http://community.qiskit.org/textbook
[29] Chen Y, Farahzad M, Yoo S and Wei T-C 2019 Detector tomography on ibm quantum computers and mitigation of an imperfect measurement *Phys. Rev.* A **100** 052315
[30] Hamilton K E, Kharazi T, Morris T, McCaskey A J, Bennink R S and Pooser R C 2020 Scalable quantum processor noise characterization *2020 IEEE Int. Conf. Quantum Computing and Engineering (QCE)* pp 430–40
[31] Bravyi S, Sheldon S, Kandala A, Mckay D C and Gambetta J M 2021 Mitigating measurement errors in multiqubit experiments *Phys. Rev.* A **103** 042605

[32] Smith A W R, Khosla K E, Self C N and Kim M S 2021 Qubit readout error mitigation with bit-flip averaging *Sci. Adv.* **7** eabi8009
[33] Geller M R and Sun M 2021 Toward efficient correction of multiqubit measurement errors: pair correlation method *Quantum Sci. Technol.* **6** 025009
[34] Kwon H and Bae J 2021 A hybrid quantum-classical approach to mitigating measurement errors in quantum algorithms *IEEE Trans. Comput.* **70** 1401–11
[35] Morello A, Pla J J, Bertet P and Jamieson D N 2020 Donor spins in silicon for quantum technologies *Adv. Quantum Technol.* **3** 2000005
[36] Sheldon S, Bishop L S, Magesan E, Filipp S, Chow J M and Gambetta J M 2016 Characterizing errors on qubit operations via iterative randomized benchmarking *Phys. Rev.* A **93** 012301
[37] Kingma D P and Jimmy B 2014 Adam: a method for stochastic optimization (arXiv:1412.6980)
[38] Anis M D S *et al* 2021 *Qiskit: An Open-Source Framework for Quantum Computing* (Zenodo) 10.5281/zenodo.2562111
[39] Nesterov Y 1983 A method for solving the convex programming problem with convergence rate $o(1/k^2)$ *Proc. USSR Acad. Sci.* **269** 543–7
[40] Smith S *et al* 2022 Using deepspeed and megatron to train megatron-turing NLG 530b, a large-scale generative language model (arXiv:2201.11990)
[41] Pan S J and Yang Q 2010 A survey on transfer learning *IEEE Trans. Knowl. Data Eng.* **22** 1345–59
[42] Goodfellow I, Bengio Y and Courville A 2016 *Deep Learning* (Cambridge, MA: MIT Press)
[43] Tan C, Sun F, Kong T, Zhang W, Yang C and Liu C 2018 A survey on deep transfer learning *Artificial Neural Networks and Machine Learning—ICANN 2018* ed V Kůrková, Y Manolopoulos, B Hammer, L Iliadis and I Maglogiannis (Berlin: Springer) pp 270−9
[44] Bozinovski S 2020 Reminder of the first paper on transfer learning in neural networks *Informatica (Slovenia)* **44** 3
[45] Lienhard B *et al* 2022 Deep-neural-network discrimination of multiplexed superconducting-qubit states *Phys. Rev. Appl.* **17** 014024