# SSQEM: Semi-Supervised Quantum Error Mitigation

Alperen Sayar
*Mef University, Tam Finans Agm*
Istanbul, Türkiye
sayara@mef.edu.tr

Şuayb Ş. Arslan
*Mef University*
Istanbul, Türkiye
arslans@mef.edu.tr

Tuna Çakar
*Mef University, Tam Finans Agm*
Istanbul, Türkiye
cakart@mef.edu.tr

2022 7th International Conference on Computer Science and Engineering (UBMK) | 978-1-6654-7010-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/UBMK55850.2022.9919474

***Abstract*—One of the fundamental obstacles for quantum computation (especially in noisy intermediate-scale quantum (NISQ) era) to be a near-term reality is the manufacturing gate/measurement technologies that make the system state quite fragile due to decoherence. As the world we live in is quite far away from the ideal, complex particle-level material imperfections due to interactions with the environment are an inevitable part of the computation process. Hence keeping the accurate state of the particles involved in the computation becomes almost impossible. In this study, we posit that any physical quantum computer system manifests more multiple error source processes as the number of qubits as well as depth of the circuit increase. Accordingly, we propose a semi-supervised quantum error mitigation technique consisting of two separate stages each based on an unsupervised and a supervised machine learning model, respectively. The proposed scheme initially learns the error types/processes and then compensates the error due to data processing and the projective measurement all in the computational basis.**

***Keywords*—Quantum Error Mitigation, Semi-supervised Learning, Clustering***

## I. Introduction

The amount of controlled hardware qubits in the era of noisy intermediate-scale quantum (NISQ) devices is insufficient to enable quantum error correction (QEC) due to the increased number of physical qubits [1] and complex interactions in between. As the main philosophy behind NISQ is to reduce the calculation load by implementing some parts of the algorithms using classical computation, alternative fault tolerance methods have gained attraction. Quantum error mitigation (QEM), on the other hand, can reduce manufacturing and measurement mistakes by repeating experiments and postprocessing data in classical means. In other words, QEM does not necessarily call for extra physical qubits to use for error rectification. There are several QEM methods available in the literature. Extrapolation [2], probabilistic error cancellation [3], quantum subspace expansion [4], and symmetry verification [5] are some of the examples of QEM techniques that are shown to be promising. In addition, the learning-based approaches are recently shown to be effective against the composite (typically non-linear [6]) error cases and better handle such adversities [7] in a quantum computation setting.

Despite the exact methods given for QEM, the applicability of such methods to real quantum computers is pretty limited. Developing QEM methods irrespective of gate-independent depolarizing noise and/or physical noise characterization frameworks [8] might be key to the success of future NISQ computers. Moreover, the characterization of the composite error due to applying various gates and measurements is shown to be of non-linear nature in a number of past studies [6]. However, all such learning approaches are typically fully-supervised and focus on a single model to fit all types of errors, independent of the number of qubits involved as well as the depth and other features of the circuits.

In this work, we have hypothesized that quantum computers (particularly the ones manufactured based on superconducting materials) tend to manifest multiple error processes acting on the input being processed. To validate/test the hypothesis, we carefully designed an experiment and collected data to form a dataset, and then present an error mitigation strategy based on two traditional machine learning techniques applied in sequential order. The main difference compared to past literature is that our approach is based on an unsupervised error process characterization which is followed by a supervised classification, hence the name semi-supervised quantum error mitigation (SSQEM). More specifically, we primarily apply a clustering approach to group error types/processes based on the error/distance quantifier using Johnson-Shannon (JS) divergence. Then, for a selected cluster, a specific classifier is trained/cross-validated based on the data that belongs to that group only. Later, supervised classical classifiers (such as Logistic Regression, LDA, etc.) and a deep learning framework are used to estimate the overall error due to multiple stages of gates (depth of the circuit) and the final measurement operation at the end.

The rest of the paper is organized as follows. In section II, the background, as well as the details of the proposed method, are provided. In our study, we have created our own dataset based on IBM's quantum computers. The details of the data collection process are also provided. In section III, we have provided our experiments and the results to validate the hypothesis of the paper. Finally, we conclude our paper in Section IV with a few future directions.

## II. Methods

### A. Theory - Background

In an $n$-qubit quantum circuit there are $2^n$ possible outcomes that can be observed after measurement using computational bases. Let $\boldsymbol{p} = [p(0) \quad p(1) \ldots \quad p(2^n - 1)]$ be the distribution of the possible measurement outcomes. To solve a specific problem, many quantum algorithms are built with the goal of encoding the solution in the probability distribution. The observed probability distribution, however, can differ from the ideal probability distribution, sometimes pretty dramatically, due to interactions with the environment during execution and the final readout errors. If we let the measurement result to be $\hat{\boldsymbol{p}} = \{\hat{p}(i)\}$, then the main objective

978-1-6654-7010-0/22/$31.00 ©2022 IEEE

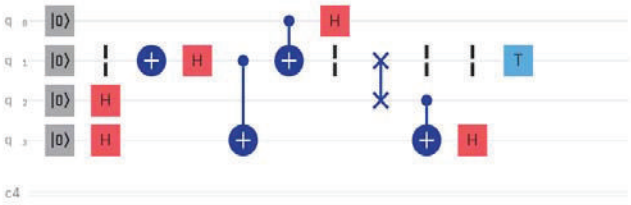

Fig. 1: An example depth-10 random QC with $n = 4$.

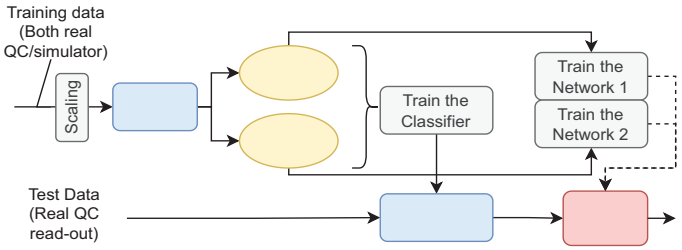


Fig. 2: The semi-supervised QEM pipeline. QC: Quantum Computer `ibmq_belem`.

is to find a transformation $\mathcal{F}$ (denoted by $\mathcal{F}^*$) such that the error between $\boldsymbol{p}$ and $\hat{\boldsymbol{p}}$ is minimized. In otherwords,

$$\mathcal{F}^* =_{\mathcal{F}} D(\boldsymbol{p}, \mathcal{F}(\hat{\boldsymbol{p}})) \tag{1}$$

where $D(.,.)$ is an appropriate distance metric. For instance, in our study we use Johnson-Shannon (JS) divergence in place of this measure. A number of studies assumed the transformation to be linear and hence estimated $\Lambda^{-1}$ (namely $\Lambda^{*-1}$) based on the data (such as least squares or a more generic regression model etc.) such that $\hat{\boldsymbol{p}} = \Lambda \boldsymbol{p}$. Thus, the estimate would be given by $\Lambda^{*-1}\hat{\boldsymbol{p}}$ [9].

In this work, we categorize the error into $m$ groups and hence we look for a collection of transformations, $\{\mathcal{F}_i\}_{i=1}^m$ and an unsupervised category mapper $f(.)$ whose output is an integer from the set $\{1, 2, \ldots, m\}$ indicating the group index. The mapping $f()$ is untrained. Hence the final quantum error mitigation will output $\mathcal{F}_{f(\hat{\boldsymbol{p}})}(\hat{\boldsymbol{p}}) \approx \boldsymbol{p}$.

### B. Dataset Generation

To be able to generate our dataset, IBM's Qiskit framework was utilized on the IBMQ's portal that provides access to real quantum computers. Python was used as the scripting language. In the data set preparation phase, to simplify our analysis, one, two, three and four-qubit quantum circuits with depth of 10 were created randomly and the contents of the dataset (the measurement outcomes) are generated by running these circuits on both simulators as well as quantum computers (five–qubit `ibmq_belem`). Ten one and two-qubit quantum gates, namely `X` gate, `Y` gate, `Z` gate, Hadamard (`H`) gate, phase (`S`) gate, `T` gate, `CX` gate, `CZ` gate, `SWAP` gate, Toffoli (`CCX`) gate are used, which are typically utilized in the implementation of various quantum algorithms. Number of gates in each circuit along with the number of stages are uniformly randomly chosen for each quantum circuit. In addition, some of the irrelevant/impractical circuits (such as concatenation of the same gate 10 times in a row) are removed from the dataset. Such filtering helped generate more practical circuits. An example random circuit for $n = 4$ qubits is shown in Fig. 1, where multiple single and two-input quantum gates are used.

The measurements are conducted both on the simulator (`simulator_statevector`) and a real quantum computer. After the measurement at the end of each circuit, the probabilities are recorded into a database. The same set of operations were also executed on the real quantum device `ibmq_belem`. In order for our methods to give more meaningful results, the number of entries generated in the dataset is expected to be large. However, a large number of circuits could not be run due to overloading and queuing mechanism used to schedule real quantum computers on IBMQ portal. We could have successfully completed running a total of 750 circuits on the portal.

A data frame was created to store the data in which seven main variables, namely, *quantum circuit*, *number of qubits*, *number of states*, *state vector*, *conjugate of the state vector*, *measurement result* and the *measurement error*, were recorded. In addition to these variables, due to the maximum number of qubits used is four, a total of 16 state vectors, their conjugates, complex conjugates, simulator probabilities and real quantum device probabilities were stored in the database.

### C. SSQEM: Clustering

After the dataset is generated, we put it through the semi-supervised QEM pipeline as illustrated in Fig. 2. The first step of the pipeline is to conduct error characterization. Since we did not know a priori about the number of error processes prevalent in a quantum computer acting on the circuit, we have used clustering methods to group measurement errors (real quantum device v.s. simulator outputs). We have clustered circuits based on the number of qubits, the depth as well as error processes they operate on. The overall processing layout as well as the details of the implementation is given next.

An automated clustering infrastructure has been created for the clustering process. Before clustering, some of the known scaling methods were first applied to the raw data and their comparisons were made in a loop. These are standardization, min-max scaling, robust scaling, Box-Cox transformation, Yeo-Johnson scaling, Gaussian-pdf scaling, max-abs scaling, L2 normalization and uniform-pdf scaling methods. These methods are previously shown to be effective before further processing [10], [11].

TABLE I: CLASSIFIER PERFORMANCE FOR CIRCUITS WITH $n = 3$. DECISION TREES ARE OBSERVED TO BE QUITE POWERFUL

| $n = 3$ | | Machine Learning Scoring Metrics | | | |
|---|---|---|---|---|---|
| Labels | Models | Accuracy | Precision | Recall | **F1-Score** |
| 0 | 2*KNN | 2*0.83 | 0.88 | 0.81 | 0.84 |
| 1 | | | 0.77 | 0.85 | 0.81 |
| 0 | 2*Logistic Regression | 2*0.76 | 0.78 | 0.81 | 0.79 |
| 1 | | | 0.74 | 0.70 | 0.72 |
| 0 | 2*Decision Tree | 2*0.85 | 0.91 | 0.81 | 0.86 |
| 1 | | | 0.78 | 0.90 | 0.84 |

TABLE II: CLASSIFIER PERFORMANCE FOR CIRCUITS WITH $n = 4$. ETC and XGBoost PERFORMANS THE BEST AMONG.

| $n = 4$ | | Machine Learning Scoring Metrics | | | |
|---|---|---|---|---|---|
| Labels | Classifier | Accuracy | Precision | Recall | F1-Score |
| 0 | 2*KNN | 2*0.73 | 0.73 | 0.70 | 0.71 |
| 1 | | | 0.73 | 0.76 | 0.75 |
| 0 | 2*Logictic Regression | 2*0.69 | 0.68 | 0.65 | 0.67 |
| 1 | | | 0.69 | 0.72 | 0.71 |
| 0 | 2*Extra Tree Classifier | 2*0.82 | 0.77 | 0.87 | 0.82 |
| 1 | | | 0.86 | 0.76 | 0.81 |
| 0 | 2*XGBoost | 2*0.79 | 0.78 | 0.78 | 0.78 |
| 1 | | | 0.80 | 0.80 | 0.80 |

We have realized that a standard *K-means clustering* is more than sufficient for the demonstration of the proposed scheme. Elbow method, Silhouette score, Davies–Bouldin and Calinski–Harabasz indexes were used for scoring the number of clusters that would best fit with the data. Based on The outputs of the methods used, an appropriate number of clusters is selected (the number of labels in our context). Note that this also determines the number of error processes we assume/identify in the quantum computer. Moreover, we have used Jensen–Shannon (JS) divergence to calculate the statistical distance between any two probability vectors such as simulator and real quantum device probabilities during the clustering operation. JS divergence between two probability vectors $\boldsymbol{p}$ and $\boldsymbol{q}$ is defined as,

$$\int(\boldsymbol{p}||\boldsymbol{q}) = \sqrt{\frac{D(\boldsymbol{p}||\boldsymbol{m}) + D(\boldsymbol{q}||\boldsymbol{m})}{2}} \tag{2}$$

where $\boldsymbol{m}$ is is the pointwise mean of $\boldsymbol{p}$ and $\boldsymbol{q}$, and $D(.||.)$ is the Kullback-Leibler (KL) divergence. Note that we have

$$0 \leq \int(\boldsymbol{p}||\boldsymbol{q}) \leq 1. \tag{3}$$

From our experiements, we have determined that two clusters is a simple rule of thumb selection and the example in Fig. 1 is based on this assumption. The same figure can be expanded to apply to more clusters.

### D. SSQEM: Classification

The second stage of the proposed semi-supervised approach is to conduct a classification based on the labeling information thanks to the clustering algorithm. This is followed by the estimation of the error and applying it to the output of a real quantum computer measurement operator.

*1) Classification:* For each dataset generated for each $n$-qubit circuit (with $n = 1, 2, 3, 4$), $20\%$ of the data is reserved for testing and $80\%$ for training. We have tested various classical machine learning models, including nearest neighbors, logistic regression, decision trees, random forests, XGBoost, ExtraTrees, SVM and LDA classifiers. Cross-validation is used for hyperparameter tuning to avoid data leakage and overfitting. We have employed *ShuffleSplit* method with grid search in **sklearn** python library for accurate/strong cross validation. One of the things we have realized that as $n$ increases the classification/clustering as well as QEM processes are harder to tune. For $n = 3$ and $n = 4$, classification results are given in Table **??** and Table **??**. As can be seen, ExtraTrees and XGBoost (more complex classifiers) show good performance for $n = 4$, whereas decision trees are good enough for $n = 3$.

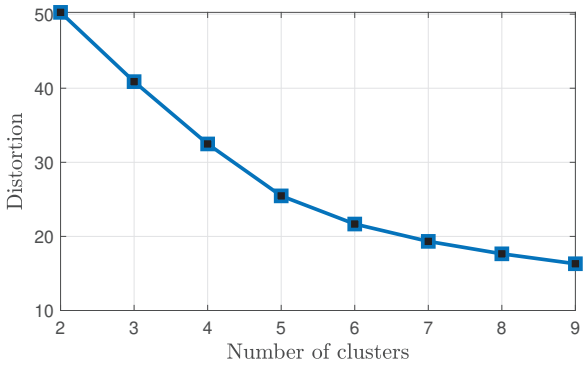


Fig. 3: Number of clusters v.s. the resulting distortion based on the Elbow method.

On the other hand, for $n = 1, 2$, we are able to achieve over $0.92$ average accuracy using LDA and Random Forests as well.

*2) Error mitigation via Deep Learning:* The classification result does not provide the QEM by itself. We need to estimate the error in the outcome probabilities to be able to rectify the read-out values of a quantum computer. The most basic approach is to use the mean value of all probability discrepancies i.e., $\frac{1}{|\mathcal{T}|}\sum_t \hat{\boldsymbol{p}}_t - \boldsymbol{p}_t$ for all training data $t \in \mathcal{T}$ that belongs to a specific cluster. Interestingly, this straightforward approach works quite well for $n = 1$ and $n = 2$. However, for $n \geq 3$, the QEM amplifies the error and makes the probabilities worse than the original quantum computer read-outs.

In our study, we further proposed to use a deep learning model as a "transformer", trained specifically for all $m$ different clusters/groups. That is to say, we have used the network architecture given in Fig. 5 for all the data in each cluster but parameters of which are optimized (using Adam optimizer) based on the data in each cluster only. For instance, for $m = 2$, we have two different network models that learn the mapping between the quantum computer read-outs and the simulator outputs (expected probabilities). After training the network, it is used to do the QEM for the classified test probability vector sample. Note that after clustering an imbalance may occur in the count of data points in each cluster. Thus, different deep learning networks may be employed for better performance. However, we are able to show improved performance with this network as well by confirming our hypothesis.

## III. Numerical Results

For all quantum circuits ($n = 1, 2, 3, 4$), we have applied various scaling options and picked the one that performed well. We have tested LDA, XGBoost and Extreme Tree Classifiers, and the mean of cross validation scores averaged nearly around 0.8. After the clustering, we obtain the mean error of probabilities between the simulator result and read-out value of the real quantum device. In addition to normal averaging, we have also used weighted average of the probability errors by taking the inverse of their JS divergence value. After QEM, the corrected probability values are compared to the outcome of the ideal simulator results in statistical sense using JS divergence. Since as the number of qubits increase, the circuit becomes more complex and the error processes will be harder

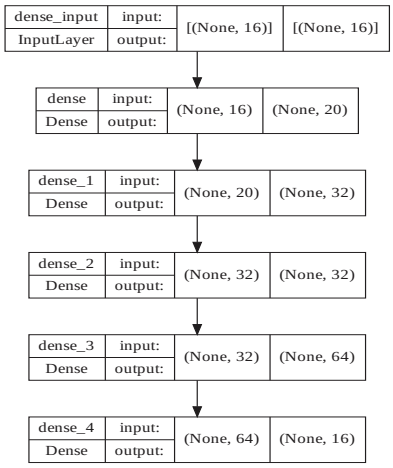


Fig. 4: The deep learning model used for QEM phase.

to deal with, we focus on $n = 4$ case which has 25 samples for label 0 (first cluster) and 23 samples for label 1 (second cluster) ensuring a balanced dataset in terms of labels.

To improve the QEM part, we have also used deep learning (DL) as illustrated in Fig. 2 to improve the performance of error estimation processes. In order to assess the effect of DL, we turned off the clustering. Without using deep learning, the JS divergence results turn out to be 0.07, 0.17, 0.26, 0.32 for $n = 1, 2, 3$ and 4, respectively. One of the issues we have realized that the divergence increases for growing $n$, probably due to the complexity of the circuit and potential multiple error processes occurring. On the other hand, using DL, we improve our performance observations dramatically such as 0.02, 0.14, 0.16 and 0.31, again in the same order of $n$. Note that for more complex circuits (such as $n > 4$ and depth$> 10$), we may have to increase the number of clusters to better match with/combat against the multitude of error processes that will likely be present.

TABLE III: JL DIVERGENCE with/without DL QEM with/ without CLUSTERING for $n = 4$.

| $n = 4$ | Jensen-Shannon Divergence Results | |
|---|---|---|
| Clusters | without DL | with DL |
| 0 | 0.4365 | 0.3479 |
| 1 | 0.2327 | 0.1763 |
| No clustering | 0.324 | 0.310 |

On the other hand, using the idea of clustering, distinct error characterization and a final DL for QEM promises a progress in the whole literature of QEM as shown in Table **??**. Despite the results being better for the second cluster (label 1), the overall average performance improvement with clustering outperforms the scheme with no clustering, supporting our original hypothesis. We have also presented a measurement and post-processing result in Fig. 5 for visual comparison.

Although the tasks taken into account in quantum supremacy are often theoretically abstract issues, the science must finally advance to demonstrate actual quantum advantage,

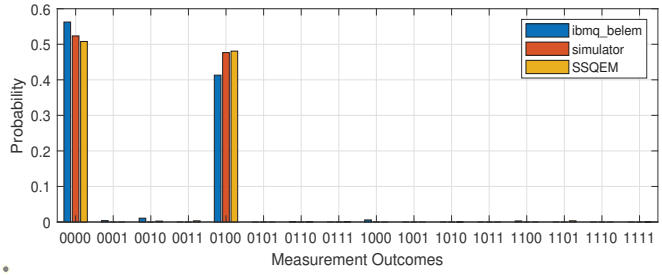


Fig. 5: An example probability distribution using a simulator and the proposed QEM.

that is, to solve a problem of practical importance with a quantum device more effectively than with any other method. For the implementation of classical quantum algorithms like Shor's factoring algorithm [12], Ruth's phase estimation algorithm [13], and Hamiltonian simulation algorithms [14], current quantum hardware only has a small number (tens) of qubits with a non-negligible gate error rate. These algorithms typically demand one to precisely control millions of qubits when taking fault-tolerance into account [15].

The so-called noisy intermediate-scale quantum (NISQ) regime, where we control tens to thousands of noisy qubits with gate errors that may be on the order of 103 or lower, is a more realistic scenario for current and near-term quantum computing before realizing a universal fault-tolerant quantum computer [16]. Although NISQ computers are not omnipresent, we may use them to combine quantum and classical computers to handle some computational tasks [17], such chemical modeling, much more quickly than with classical computers [18]. Fully coherent deep quantum circuits might not be necessary since a significant amount of the computing load is handled by the classical computer. These simulation techniques are known as hybrid quantum-classical algorithms because they employ both quantum and classical computers [19]. Quantum error mitigation strategies can also be applied after the experiment data has been processed in order to correct calculation mistakes [20]–[22]. Because quantum error mitigation does not need the encoding of qubits, as complete error correction does, it adds to a significant reduction in the amount of qubits needed for NISQ simulation [23], [24].

## IV. Conclusions

In this study, we have hypothesized that the number of distinct error processes that act on the data in quantum computers increase as the number of qubits as well as the depth of the circuit grows. We have done extensive testing in one of the quantum computers on IBMQ portal and verified our hypothesis through a two-stage semi-supervised QEM methodology. This approach may have the potential to segregate the error processes and attack each separately in the future. We believe the results of this study, when extended and tested on a variety of quantum computers, will lead to improved QEM methods for future NISQ computers.

## References

[1] A. R. Calderbank and P. W. Shor, "Good quantum error-correcting codes exist," *Physical Review A*, vol. 54, p. 1098–1105, Aug 1996.

[2] Y. Li and S. C. Benjamin, "Efficient variational quantum simulator incorporating active error minimization," *Physical Review X*, vol. 7, no. 2, p. 021050, 2017.

[3] 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.

[4] J. R. McClean, Z. Jiang, N. C. Rubin, R. Babbush, and H. Neven, "Decoding quantum errors with subspace expansions," *Nature communications*, vol. 11, no. 1, pp. 1–9, 2020.

[5] S. McArdle, X. Yuan, and S. Benjamin, "Error-mitigated digital quantum simulation," *Physical review letters*, vol. 122, no. 18, p. 180501, 2019.

[6] J. Kim, B. Oh, Y. Chong, E. Hwang, and D. K. Park, "Quantum readout error mitigation via deep learning," *arXiv preprint arXiv:2112.03585*, 2021.

[7] C. Kim, K. D. Park, and J.-K. Rhee, "Quantum error mitigation with artificial neural network," *IEEE Access*, vol. 8, pp. 188853–188860, 2020.

[8] T. J. Proctor, A. Carignan-Dugas, K. Rudinger, E. Nielsen, R. Blume-Kohout, and K. Young, "Direct randomized benchmarking for multiqubit devices," *Physical review letters*, vol. 123, no. 3, p. 030503, 2019.

[9] F. B. Maciejewski, Z. Zimborás, and M. Oszmaniec, "Mitigation of readout noise in near-term quantum devices by classical post-processing based on detector tomography," *Quantum*, vol. 4, p. 257, 2020.

[10] S. A. Othman and H. T. M. Ali, "Improvement of the nonparametric estimation of functional stationary time series using yeo-johnson transformation with application to temperature curves," *Advances in Mathematical Physics*, vol. 2021, 2021.

[11] S. S. Arslan and E. Zeydan, "On the distribution modeling of heavy-tailed disk failure lifetime in big data centers," *IEEE Transactions on Reliability*, vol. 70, no. 2, pp. 507–524, 2021.

[12] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, p. 1484, 1997.

[13] A. Y. Kitaev, "Quantum measurements and the abelian stabilizer problem," *Quantum Physics*, 1995.

[14] S. Lloyd, "Universal quantum simulators," *Science*, vol. 273, pp. 1073–1078, 1996.

[15] J. O'Gorman and E. T. Campbell, "Quantum computation with realistic magic-state factories," *Phys. Rev. A.*, vol. 95, 2012.

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

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

[18] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'brien, "A variational eigenvalue solver on a photonic quantum processor," *Nature communications*, vol. 5, no. 1, pp. 1–7, 2014.

[19] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, "The theory of variational hybrid quantum-classical algorithms," *New Journal of Physics*, vol. 18, no. 2, p. 023023, 2016.

[20] J. G. E. Farhi and S. Gutmann, "A quantum approximate optimization algorithm," *Quantum Physics*, 2014.

[21] N. Hatami, "Efficient variational quantum simulator incorporating active error minimization," *Phys. Rev. X*, vol. 7, 2017.

[22] G. C. G. Torlai, G. Mazzola and A. Mezzacapo, "Precise measurement of quantum observables with neural-network estimators," *Phys. Rev. Research*, vol. 2, 2020.

[23] S. B. K. Temme and J. M. Gambetta, "Error mitigation for short-depth quantum circuits," *Phys. Rev. Lett.*, vol. 119, 2017.

[24] S. C. B. S. Endo and Y. Li, "Practical quantum error mitigation for near-future applications," *Phys. Rev. X*, vol. 8, 2018.