IEEE Access
Multidisciplinary ⁝ Rapid Review ⁝ Open Access Journal

Received 4 April 2023, accepted 15 May 2023, date of publication 22 May 2023, date of current version 1 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3278600

RESEARCH ARTICLE

Trainable PQC-Based QRAM for Quantum Storage

KOUSTUBH PHALAK, (Student Member, IEEE), JUNDE LI, (Student Member, IEEE), AND SWAROOP GHOSH, (Senior Member, IEEE)
Computer Science and Engineering Department, The Pennsylvania State University, State College, PA 16802, USA

Corresponding author: Koustubh Phalak (krp5448@psu.edu)

This work was supported in part by NSF under Grant CNS-1722557, Grant CNS-2129675, Grant CCF-2210963, Grant CCF-1718474, Grant OIA-2040667, Grant DGE-1723687, Grant DGE-1821766, and Grant DGE-2113839; and in part by the Seed Grants from Penn State Institute for Computational and Data Sciences (ICDS).

ABSTRACT Quantum Machine Learning (QML) is a new domain of Machine Learning (ML) and Quantum Computing (QC) that uses a trainable Variational Quantum Circuit (VQC) to solve various learning problems. Classical data is transformed to the quantum Hilbert space using an embedding scheme and the VQC performs quantum operations on this transformed quantum data using parametric quantum gates. In this work, we present a new embedding scheme using Quantum Random Access Memory (QRAM) that takes address as input, and gives data as the output similar to a classical Random Access Memory (RAM). We propose a basic circuit-based QRAM architecture and its application in, (a) storage for ML usage and, (b) storage of binary data. For ML, we store images of digits dataset into the QRAM and perform binary classification using a Quantum Neural Network (QNN). We observe that QNN driven by the proposed QRAM converges faster (at 6^{th} epoch) with 100% classification accuracy compared to QNN with normal embedding and classical Fully Connected Neural Network (FCNN) setups which converge with same accuracy at around 10^{th} and 15^{th} epochs, respectively. We obtain $\approx$ 63% classification accuracy on real hardware from IBM. For storage of binary data, we measure Hamming Distance (HD) and percentage correct predictions for quality evaluation of the proposed QRAM. We observe that the HD and percentage correct predictions worsens as the number of address lines increases. We propose two techniques to improve the QRAM accuracy namely, (i) clustering of raw data and allocating one QRAM per cluster and, (ii) bit splitting to divide wider data into smaller chunks and allocating one QRAM per split. Clustering provides best results by improving the HD for 9-address QRAM by $\approx$1.95X than pure QRAM and $\approx$1.74X improvement compared to bit-split (which provides $\approx$1.11X improvement than pure QRAM).

INDEX TERMS Quantum RAM, machine learning, classification.

I. INTRODUCTION

Quantum computing (QC) has grown rapidly in recent years due to technological advancements in quantum hardware. A sub-field of QC that has recently generated lot of interest is Quantum Machine Learning (QML) which is the augmentation of QC with Machine Learning (ML) to solve various learning problems. QML can potentially offer quantum speed up of algorithms like quantum clustering, quantum decision trees, quantum support vector machines and quantum neural networks [1]. A key aspect in these algorithms is the conversion of classical data to the quantum Hilbert space. As per [2], [3], supervised learning in the quantum domain is a kernel method similar to classical support vector machines which uses a non-linear kernel to map classical data to the large Hilbert space. The non-linear kernel used is an embedding scheme to encode classical data such as, amplitude embedding which maps 2^n features on n qubits, and angle embedding and basis embedding that map n features on n qubits [4]. These quantum embeddings can be considered as a primitive way of storing classical data in quantum format prior to sending it to the actual quantum gates of the QML model

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

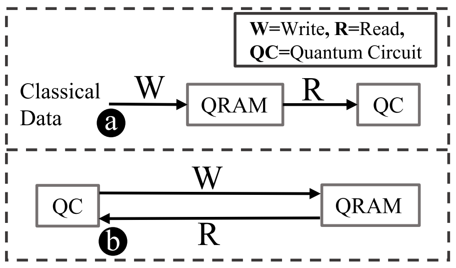


FIGURE 1. QRAM usage scenarios, (a) writing classical data onto QRAM and forwarding to a quantum circuit and (b) storing intermediate quantum data in QRAM and retrieve as needed.

during training/inference. However, they are not addressable like classical memories. Instead, having a Quantum Random Access Memory (QRAM) that stores and loads quantum data would be much more effective for QML problems. The user should be able to store new classical data or update it onto the QRAM and load the stored quantum data as needed. The existing QRAM architectures [5], [6], [7] write and read data into superposition.

A QRAM can be used in following scenarios: (i) to write classical data onto the QRAM and forwarding the data in quantum format to a quantum circuit as a read operation (Fig. 1(a)) and (ii) to store intermediate quantum data from a quantum circuit into QRAM and reading that data from the QRAM back to the quantum circuit. (Fig. 1(b)). The second scenario is challenging to achieve given the technology limitations of the existing quantum hardware. Nevertheless, higher order states of qubits (called qudits) as storage elements have been proposed [8], [9]. A major limitation of qudits is the quadratic increase in the error rates due to presence of more than two states which in turn can degrade the fidelity. Orthogonal to above approaches, quantum sensors perform readout from optical memories via quantum illumination [10], [11] to directly obtain the data in quantum form for further processing. We propose a Parametric Quantum Circuit (PQC)-based trainable approximate QRAM. The proposed QRAM falls under the first scenario in Fig. 1(a). We also show the application of the proposed QRAM for classification and storage of binary data. *To the best of knowledge, this is the first practically realizable approximate QRAM architecture which can be used to load arbitrary data in quantum form to any quantum circuit.*

In the rest of the paper Section II describes background and related works. Sections III, IV and V presents QRAM architecture and its applications. We present discussion in Section VI and conclude in Section VII.

II. BACKGROUND AND RELATED WORKS

A. BACKGROUND

1) QUBITS

Quantum bits or qubits are the quantum equivalent of classical bits that are fundamental units of a quantum computer. While a classical bit can store only 1 or 0 at a time, a qubit can store both 0 and 1 at the same time. To put it in formal terms, a qubit has a *quantum state* that can be represented as $|\psi\rangle = \begin{bmatrix} a \\ b \end{bmatrix}$ where a^2 is the probability of qubit being measured to 0 and b^2 is the probability of qubit being measured to 1 ($a^2 + b^2 = 1$). A qubit has two special states called basis states, where one state has pure 0 ($a = 1, b = 0$ denoted as $|0\rangle$) and other state has pure 1 ($a = 0, b = 1$, denoted as $|1\rangle$).

2) QUANTUM GATES

Quantum gates are unitary operations performed on qubits to change the state of said qubits. These gates can act either on a single qubit (Hadamard gate, Pauli X/Y/Z, RX/RY/RZ gates, reset, measurement) or on more than a single qubit (CNOT gate, SWAP gate, controlled Pauli gates). Reset and measurement are two special gates, where reset gate in a literal sense resets the qubit state to $|0\rangle$ and measurement measures the qubit state classically to either 0 or 1.

3) QUANTUM CIRCUIT PQC

A quantum circuit is an ordered sequence of quantum gate operations that are performed in time units. All quantum gates present in a quantum circuit are decomposed to equivalent native gates. A PQC is a trainable quantum circuit which can be trained like a normal ML model. The user can initialize the parameter values and train the PQC using any classical optimizer.

B. RELATED WORKS

1) BUCKET BRIGADE QRAM [5]

This QRAM (Fig. 2(a)) incorporates bifurcation graph-based classical RAM structure into quantum domain with the help of qutrits. A qutrit is a ternary version of a qubit that stores three states primarily: $|left\rangle$, $|right\rangle$, and $|wait\rangle$ states. The $|left\rangle$ state denotes binary 0 and $|right\rangle$ state denotes binary 1. Each leaf in the QRAM is a memory cell, where the data is stored. At the start, all the qutrits are set in $|wait\rangle$ state and then, from the root node the input states of the address to be accessed are sent one by one to each qutrit, and the qutrit state is changed accordingly. Fig. 2(a) shows memory cell at 010 address being accessed.

2) CIRCUIT-BASED QRAM [6]

This QRAM [6] is a quantum circuit that stores integer data into superposition states (Fig. 2(b)). For M datapoints each of size N bits, N qubits are used as data lines and $log_2(M)$ qubits are used as address lines. Additionally, a register line is kept to store angle of the data. The QRAM stores the integer data one by one, so the circuit depth is high, but circuit width is linear. For storing each data, there are 3 stages: flip, register and flop stages. The flip stage has all classically controlled not gates to set all the bits pertaining to a particular address and data to 1. This is a compute stage and the flop does the reverse of this using the same gates, so it is an uncompute stage. In the middle of both is the register stage, where the angle of the data is stored using a multi-controlled angle gate. Fig. 2(b) shows one special such case of QRAM where the address and the data lines are the same. In a general scenario, the address and

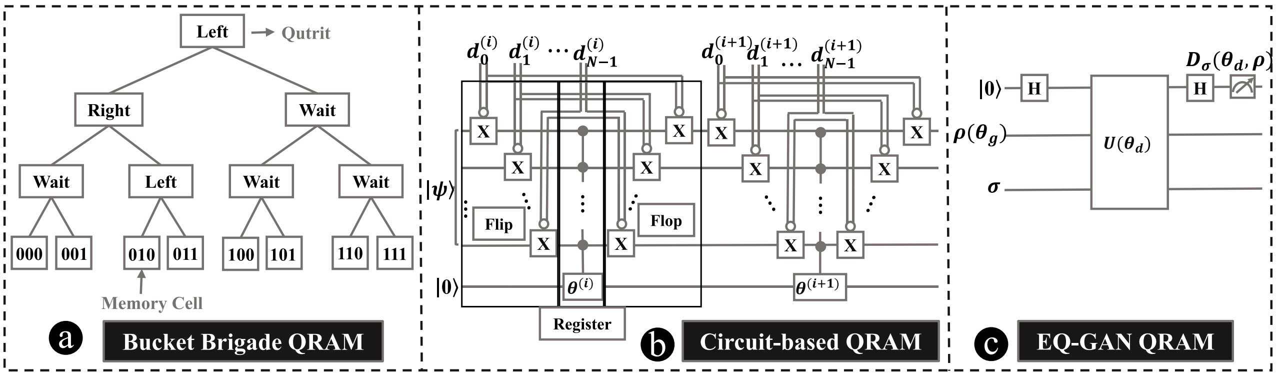


FIGURE 2. Existing QRAM architectures. (a) bucket brigade QRAM, (b) quantum circuit-based QRAM (c) EQGAN QRAM. Reproduced from [5], [6], and [7].

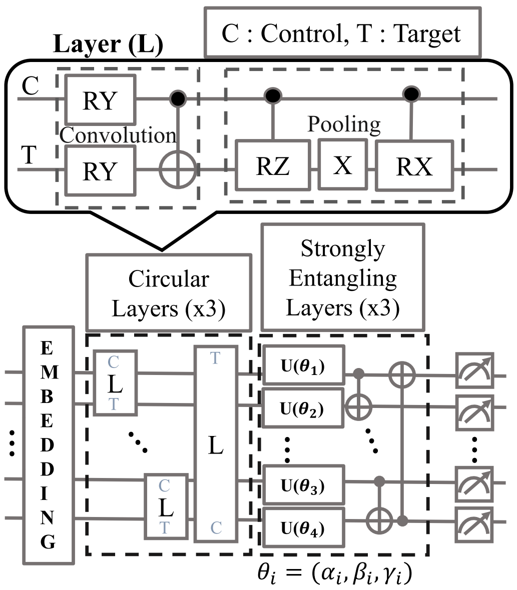


FIGURE 3. Basic QRAM architecture. Strongly entangling layers are replicated from [12] which in turn are inspired from [13].

data lines will be separate and work in the same fashion of flip-register-flop stages to store angle for a particular data in an address.

3) ENTANGLING QUANTUM GENERATIVE ADVERSARIAL NETWORK (EQGAN) QRAM [7]

This QRAM prepares quantum state for classical data before sending them for classification to a Quantum Neural Network (QNN). The EQGAN is a trainable QGAN circuit which learns to generate data from two normal distributions. The EQGAN (Fig. 2(c)) training reaches its optimal point when $\rho(\theta_g) = \sigma$, where θ_g are generator parameters, θ_d are discriminator parameters, $\rho(\theta_g)$ is the generated data state, σ is the original data state. This quantum data from EQGAN is then sent to quantum classifier.

i) CHALLENGES WITH EXISTING QRAM

Qutrit technology used for bucket brigade QRAM [5] is currently not easily achievable. The QRAM circuit [6] is simple to recreate, but the read operation is not well-defined and the

TABLE 1. Comparison of various quantum memories.

QRAM Type	Requirements	Pros	Cons	Complexity (n = #qubits)
Fanout QRAM [14]	Less decoherence error, TI qubits	Less interactions (O(1)) with quantum bus to load data	Susceptible to decoherence error	Depth: O(2^n), #Interactions: O(n)
Bucket-Brigade QRAM [5]	Qutrits	Solves decoherence problem of fanout QRAM	Qutrits are not easily realizable	Depth: O(2^n), #Interactions: O(n^2)
Circuit-based QRAM [6]	Superconducting /TI qubits	Easy to realize, uses separate qubits to store addresses & data in superposition	Limited applications (eg. quantum forking)	Depth: O(2^n), #Interactions: O(n)
EQ-GAN QRAM [7]	Superconducting /TI qubits	Yields 20% better classification accuracy	Data is not addressable	Depth: O(1), #Interactions: O(n)
Proposed QRAM	Superconducting /TI qubits	Addressable data stored in quantum Hilbert space	Cannot be scaled for large datasets	Depth: O(1), #Interactions: O(n)

stored data can only be integer. The authors employ quantum forking to use values stored in superposition in QRAM ([15]). However, the scope of its applicability is constrained. The authors in [7] preprocess data in quantum and demonstrate superior results for classification as opposed to without preprocessing. But unlike a conventional RAM, the EQGAN does not hold addressable data and only reproduces a synthetic data. Other related works that alter the existing QRAM and/or explore better implementation are also proposed [15], [16]. Various QRAMs and their pros/cons and other features are compared in Table 1.

III. PROPOSED QRAM ARCHITECTURES

A. BASIC ARCHITECTURE

We define basic architecture for our QRAM as a PQC with n address lines for less than 2^n datapoints (Fig. 3) that has an $f : n \rightarrow n$ embedding mapping like angle or basis embedding, three circular layers with each layer consisting of one convolution ansatz (2 RY gates, 1 CNOT gate) and one pooling ansatz (1 controlled RZ gate, 1 X gate and 1 controlled RX gate) respectively (shown in bubble in Fig. 3) and three strongly entangling layers [12] followed by a measurement on

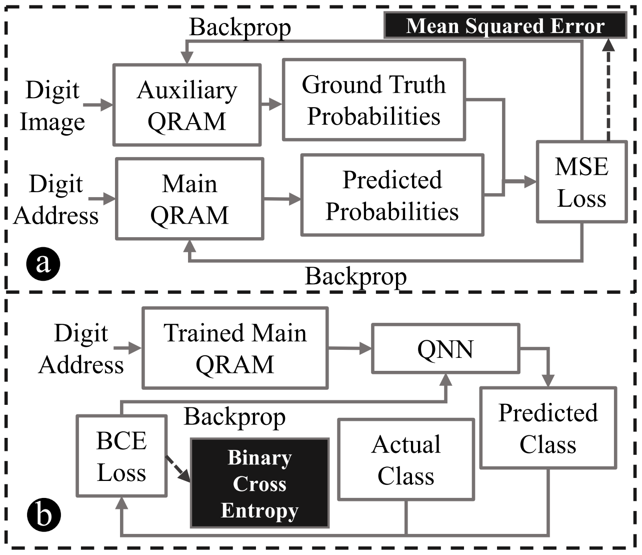


FIGURE 4. Classification using QRAM + QNN setup. (a) Training of QRAM with auxiliary and main PQCs and (b) training of QNN using trained QRAM.

each qubit. The reason for choosing this particular ansatz is its lowest circuit depth and gate count which in turn can provide better fidelity of computation on real quantum hardware. The type of measurement can vary depending on the application. We use probability measurement of all basis states for ML task and expectation value measurement for storage of binary data. The write operation for this basic QRAM involves training the model whereas read operation involves loading binary address to the trained model. Although the proposed PQC-based QRAM does not store quantum data in superimposed state it does provide addressability to the data that is stored in quantum format.

B. TRAINING ARCHITECTURE OF QRAM

We use the basic QRAM architecture defined above to design QRAMs for each application. For the ML task, we use an auxiliary QRAM which takes digit image and generates ground truth probabilities for the qubits and the main QRAM which takes address of the digit image and outputs predicted probabilities (Fig. 4(a)). Training of QRAM is done in two steps; in the first step both the auxiliary QRAM and main QRAM parameters are trained so that the ground truth probabilites and the predicted probabilities are as close to each other as possible, and in the second step the parameters of the auxiliary QRAM are fixed and only the main QRAM parameters are trained to further reduce the loss. Both steps involve training the QRAM for 100 epochs each. After training, the auxiliary QRAM is discarded and the output of the main QRAM for each address is provided to a QNN in quantum format for classification (Fig. 4(b)). For the binary data QRAM, a single basic architecture PQC is used which outputs Pauli-Z expectation value measurement to output individual data bits on all qubits. For both the scenarios, the proposed QRAM is approximate since it predicts the output in each address approximately rather than accurately. For ML QRAM, the approximation error is difference between the predicted probabilities and ground truth probabilities for each image, and for binary data QRAM, the approximation error arises from the Hamming Distance (HD) between predicted data bits and actual data bits. Note that the proposed QRAM architecture can handle only sequential access of data and does not scale for large datasets for which we can use multiple parallel banks of QRAMs.

IV. QRAM FOR MACHINE LEARNING TASKS

A. EXPERIMENTAL SETUP

We perform binary classification on 0 and 1 digits (from digits dataset by UCI Machine Learning Repository) using three different setups: (i) proposed QRAM architecture augmented QNN, (ii) QNN with embedding of input images using amplitude encoding, and (iii) classical Fully Connected Neural Network (FCNN). Later, we also perform multi-class classification on all 10 classes and inferencing of binary classification on real quantum hardware. For the above setups, we report the training and testing losses and, training and testing accuracies. We also compare the three cases and interpret the results. For all the setups, we maintain a batch size of 16 and learning rate of 0.001. We use 360 total images with 85-15 train-test split. Further details of the three cases are provided below.

1) QRAM + QNN CLASSIFICATION

While training the QRAM, the digit image is embedded into the auxiliary QRAM using amplitude embedding, and the digit address is embedded into the main QRAM using angle embedding. Both the auxiliary QRAM and main QRAM are PQCs that use 9 qubits each and output classical probabilities from all 9 qubits ($2^9 = 512$ probability values). The auxiliary QRAM probabilities are considered as ground truth, and main QRAM probabilities are considered as predictions. Mean Squared Error of all individual probabilities are calculated, and then gradients of parameters are calculated with respect to the calculated loss. The training of the QRAM is performed in two stages. In the first stage, we make both the auxiliary QRAM and main QRAM parameters trainable. In the second stage, we fix the auxiliary QRAM parameters and train only the main QRAM (Fig. 4(a)). Both the stages utilize 100 epochs for training. Training auxiliary and main QRAMs in the first stage prepares the auxiliary QRAM to generate good ground truth probabilities and in the second stage, the main QRAM is trained so that the predicted probabilities are as close to these generated ground truth probabilities. We refer these two stages of training as the first phase of classification.

After the QRAM training, a QNN is added after the QRAM, and the probability outputs are forwarded to the QNN in quantum format (Fig. 4(b)). The QNN then learns to classify the class of the image by distinguishing between the probability output for digit 0 images and probability output for digit 1 images. We call this stage as second phase of

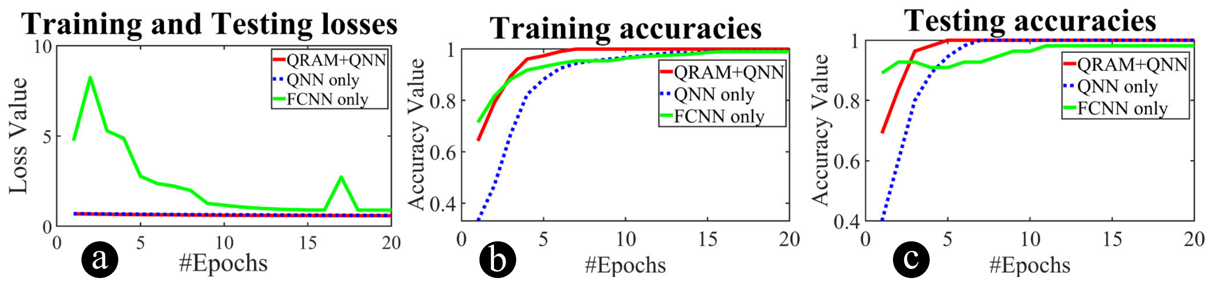


FIGURE 5. Experimental results for binary classification task for all setups, (a) training and testing losses, (b) training accuracies, and (c) testing accuracies.

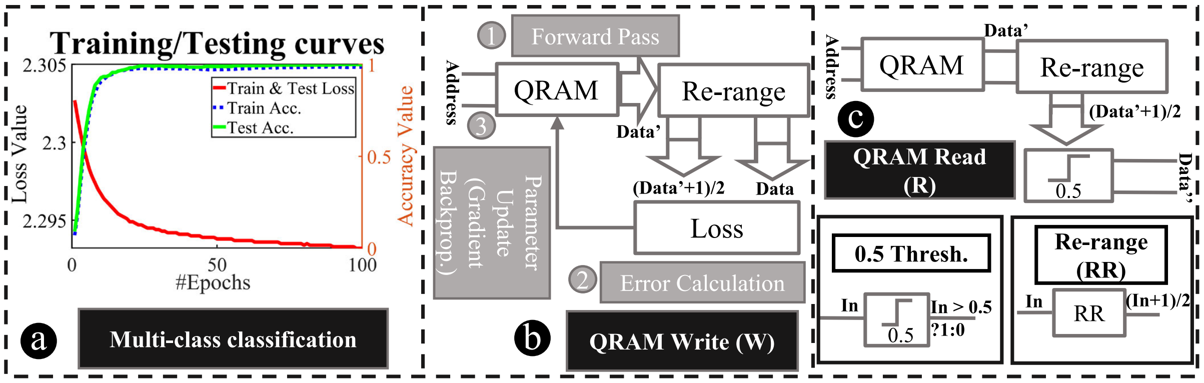


FIGURE 6. (a) Multi-class classification results, (b) QRAM PQC training as QRAM W operation, (c) QRAM access as QRAM R operation. Thresholding at 0.5 and re-range error correction methods are defined as shown in the figure.

classification which trains for 100 epochs. For best outcome, we use two separate QRAMs, one only for digit 0, and one only for digit 1. Using two different QRAMs create more distinguishable probability outputs that the QNN can easily classify. The QNN outputs the predicted class, which is then compared to the actual class using binary cross entropy loss. The QRAM and QNN PQCs both use the basic QRAM architecture shown in Fig. 3, with the exception that QNN does not have embedding since it is already getting data from QRAM in quantum format.

2) QNN ONLY CLASSIFICATION

We use the same QNN structure used in QRAM + QNN classification but directly embed images using amplitude embedding. Similar to the previous case, the QNN outputs the predicted class of the image and is compared with the actual class using binary cross entropy loss. The training is done for 100 epochs.

3) FCNN ONLY CLASSIFICATION

In this case, we send the images directly through an FCNN which gives a single output of the predicted class. Binary cross entropy loss is calculated between predicted and actual classes. The FCNN has 3 layers, with 64 (input), 16 (hidden) and 1 (output) neurons respectively and is trained for 100 epochs.

B. CLASSIFICATION RESULTS

For each case, the training and testing loss curves (Fig. 5(a)) are nearly identical. This is because the logarithmic factor in binary cross entropy brings the training and testing loss values very close to each other. Fig. 5(b) and (c) reports the training and testing accuracies respectively. We observe 100% classification accuracy for all the three setups. For the QRAM + QNN setup, we observe QRAM loss reduces from 0.0203 to 0.0028. We note that the QRAM + QNN and QNN with embedding only setups outperform FCNN in terms of loss and convergence time. QRAM + QNN converges the fastest at around 6th epoch, QNN with embedding converges around 10th epoch, and FCNN converges at around 15th epoch. The final loss for QRAM + QNN is around 0.53, for QNN with embedding is 0.48 and for FCNN only is 0.89. From the results, following points could be inferred, (i) QRAM storage in first phase of classification could pre-process the *write* operation, thereby accelerating the second phase of classification. The first phase relates more to *write* of QRAM, whereas second phase more to *read* operation and, (ii) the faster convergence of two quantum algorithms (i.e., QRAM+QNN and QNN with embedding) relative to FCNN indicates potential quantum computing advantages with stronger representation powers of QNNs, as extensively demonstrated in the QNN community.

C. MULTI-CLASS CLASSIFICATION

To further show the robustness of the QRAM-based classification, we perform multi-class classification of all digit classes. We allocate one QRAM per digit class (total 10 QRAMs for 10 classes) and individually train them using the auxiliary and main QRAM architecture. We use around 1800 total images (~180 images per class) with 85-15

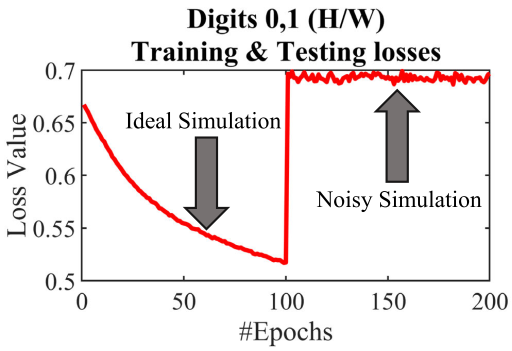


FIGURE 7. Loss value for 100 epochs on ideal simulator followed by 100 epochs on noisy simulator.

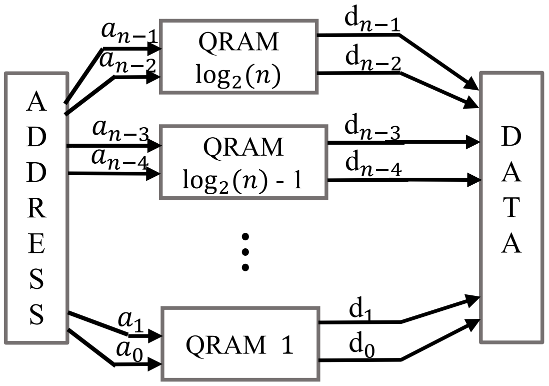


FIGURE 8. Training procedure to train binary data QRAM with bit-splitting method.

train-test split. Next, we send the data from all the QRAMs to the QNN one by one until they are classified. For the classification, we use Softmax activation function to generate the probabilities of all the 10 classes and the cross entropy loss is calculated for gradient calculation. The results are shown in Fig. 6 (a). The QRAM loss reduces from 0.0191 to 0.0031, and we observe convergence at 20^{th} epoch with higher final loss ($\approx$2.29) and $\approx 98 - 99\%$ training-testing accuracies.

D. INFERENCING RESULTS ON REAL HARDWARE

We perform experimental inferencing of the digits images using IBM Oslo and Jakarta and compared with Fake Jakarta backends which is calibrated with real hardware. Since both the real backends have only 7 qubits, we restricted our dataset size to $2^6 = 64$ images for classes 0 and 1. This dataset is split into training and testing in a roughly 80-20 split (53 training, 11 testing). We train the model on *default.qubit* simulator by Pennylane (100 epochs noiseless simulation followed by 100 epochs noisy simulation), and perform inferencing with the testing set on the backends. We also perform inferencing on Pennylane's default.qubit and IBM's QASM simulators for calibration.

We observe, (a) 100% testing accuracy on both default.qubit and QASM simulators and (b) roughly around 63% testing accuracy on IBM Oslo, IBM Jakarta hardware and Fake Jakarta. The reduction in the testing accuracy is due to the noises present in the quantum hardware and relatively larger circuit depth of the model converted to the native gate

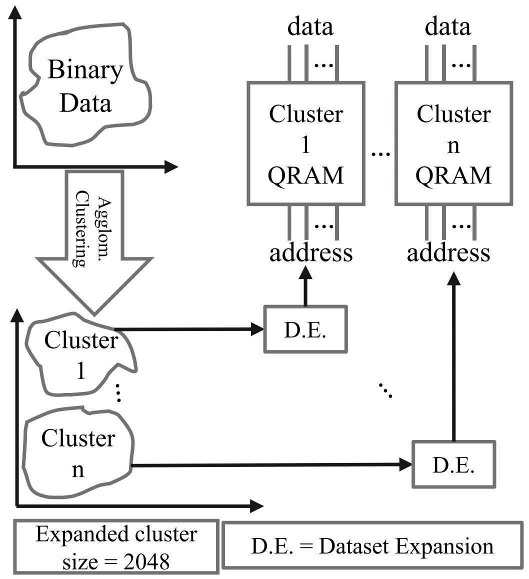


FIGURE 9. Training procedure to train binary data QRAM with agglomerative clustering pre-processing.

set of the hardware ($\approx$240 without decomposition, $\approx$1000 with decomposition). This can be noted from the loss value in Fig. 7 that shows jump at 100^{th} epoch when simulator is switched to fake backend while keeping the same model and training parameters. Note that this accuracy is similar to the EQGAN classification accuracy of 65% in [7], which was also performed on real Google Sycamore quantum processor. One can improve the results by reducing the noise characteristics of noise model of the quantum hardware by a constant factor (say, $\frac{1}{10}^{th}$ to $\frac{1}{100}^{th}$ of original hardware noise) and add modified noise model to simulator. We also note QRAM loss reduction from 0.0274 to 0.0078.

V. QRAM FOR STORAGE OF BINARY DATA

A. EXPERIMENTAL SETUP

For storage of binary data, we feed address value as input to the QRAM and obtain approximate classical data value as the output. The QRAM structure is defined based on the number of address lines and data lines used. For each setup, we keep the number of address lines and data lines equal. In order to embed each address bit into a single qubit, we keep the number of qubits same as the number of address and data lines. For an n-address QRAM, the number of address-data pairs will be 2^n, and for each address, the data is chosen randomly in the range of $\{0, 1, 2, \ldots .2^n - 1\}$. Since 2^n address-data pairs are too low to train the QRAM when n is low, we replicate the entire dataset of 2^n datapoints multiple times (called dataset expansion). For example in a 2-address QRAM, if the address-data pairs are $\{00-01, 01-11, 10-00, 11-01\}$, these 4 pairs are repeated continuously $\{00-01, 01-11, 10-00, 11-01, 00-01, 01-11, 10-00, 11-01, \ldots\}$ until the overall dataset size is large enough. Next, we train the QRAM PQC for 100 epochs and evaluate the average HD and percentage correct predictions per epoch. For any particular datapoint if the HD between the predicted data output and

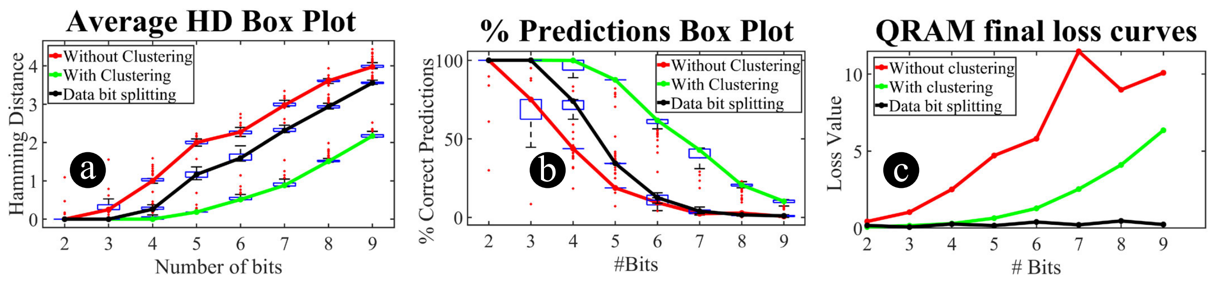


FIGURE 10. Binary QRAM results without clustering, with clustering and with data bit-splitting, (a) Hamming Distance (HD), (b) percentage correct prediction and (c) final loss curve for different number of address lines.

actual data is zero we assume that the prediction is correct otherwise we assume that the prediction is incorrect. These metrics are evaluated for 100 epochs from 2 to 9 address lines. We also propose two methods to improve the results namely, (i) a bit-splitting method to split the data bits and allocate to different PQCs and (ii) clustering the data and sending each clustered data to different QRAM.

1) BIT-SPLITTING

One way to reduce the average HD and increase the percentage correct predictions for a QRAM at higher address lines is to split the data bits into smaller chunks and allocate one QRAM per split. Here, we divide the data bits into subset of 2 bits per QRAM and measure the outputs on two qubits. The individual outputs of the QRAMs are then stitched together to get the overall data prediction and loss. For example, a 4-address QRAM will be split to two QRAMs such that the first QRAM will measure the first two bits of data and the second QRAM will measure the remaining two bits. We show the bit-splitting procedure for a general n-address QRAM in Fig. 8.

2) CLUSTERING

Another method of improving the metric values is to cluster the similar data i.e., data with less variance. We perform agglomerative clustering of the data such that intra-cluster average HD is minimized. Then for each cluster we define a separate QRAM PQC and train the expanded dataset of the cluster. We use elbow method [17] to note that $k = n + 1$ clusters is the elbow point (i.e., optimal) for n address lines. We show the process of clustering and storing the binary data in QRAM in Fig. 9.

B. EXPERIMENTAL RESULTS

We observe a general trend of increase in HD and decrease in percentage correct predictions with increase in number of address lines. This is expected because the increase in number of datapoints makes it difficult for the QRAM to predict all individual bits of datapoints correctly. For pure QRAM (without clustering/bit-splitting), we observe an average HD of 0 and 100% correct predictions only till 2-address QRAM. The HD then becomes non-zero (Fig. 10 (a)) and increases from 0.25 to 3.97. The percentage correct predictions reduce from 75% (3-address QRAM) to 0.58% (9-address QRAM) (Fig. 10 (b)). The bit-splitting method gives zero HD till 3-address QRAM and becomes non-zero from 4-address QRAM and varies from 0.25 to 3.55. Percentage correct predictions in this range vary from 74% to 0.97%. With clustering, we observe perfect predictions till 4-address QRAM. From 5-address QRAM the HD becomes non-zero (HD = 0.18) and keeps on going up till 9-address QRAM (HD = 2.03). The percentage correct predictions reduces from 87.5% (5-address QRAM) to 10.1% (9-address QRAM). We also show the loss plot of final loss values for each QRAM from 2 to 9 address lines for pure QRAM, QRAM w/ clustering, and QRAM w/ bit-splitting. We observe best results with clustering i.e., ≈1.95X improvement in HD compared to pure QRAM while bit-splitting only provides ≈1.11X improvement.

VI. DISCUSSIONS

A. OVERHEAD OF THE PROPOSED QRAM

For classification, we require 1 QRAM per digit and for clustering and bit-splitting, we require 1 QRAM per cluster and per split, respectively. Note that the extra QRAMs do not require any additional qubit resources since the same qubits can be reused to create and train the other QRAMs. The QRAM parameters can be stored classically and loaded into the qubits as needed for an application. The only overhead of multiple parallel QRAMs is in terms of training time due to their individual training.

B. SCALABILITY

The number of QRAMs required scale linearly either with number of classes or with number of address lines. For ML applications, one can use one QRAM per class to scale to larger datasets. For scalable binary storage, one can use parallel banks of QRAMs each of which will store small fraction of data and address. For example, 8-bit QRAM can be realized using 16 parallel banks of the proposed QRAM (w/ clustering) each of which will store 16 address/data reliably.

C. QRAM IMPROVEMENTS

In this work, the PQC structure of the QRAM is kept fixed, and also the storage of incremental data is not supported. A detailed analysis of the impact of the choice of PQC ansatz and task-incremental learning [18] of the QRAM PQC can be explored in the future.

VII. CONCLUSION

We presented a PQC-based QRAM for ML task and storage of binary data. Compared to classification without QRAM, the proposed QRAM reduced the training time by up to 60% at iso-accuracy for classification. To circumvent the poor HD at wider address/data for storage of binary data, we proposed data bit-splitting and agglomerative clustering-based preprocessing approaches. The proposed QRAM with clustering can reliably predict binary data for up to 4-bit address and data widths.

REFERENCES

[1] M. Schuld, I. Sinayskiy, and F. Petruccione, "An introduction to quantum machine learning," *Contemp. Phys.*, vol. 56, no. 2, pp. 172–185, Apr. 2015.

[2] M. Schuld, "Supervised quantum machine learning models are kernel methods," 2021, *arXiv:2101.11020*.

[3] M. Schuld and N. Killoran, "Quantum machine learning in feature Hilbert spaces," *Phys. Rev. Lett.*, vol. 122, no. 4, Feb. 2019, Art. no. 040504.

[4] M. Schuld and F. Petruccione, *Supervised Learning With Quantum Computers*, vol. 17. Cham, Switzerland: Springer, 2018.

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

[6] D. K. Park, F. Petruccione, and J.-K.-K. Rhee, "Circuit-based quantum random access memory for classical data," *Sci. Rep.*, vol. 9, no. 1, pp. 1–8, Mar. 2019.

[7] M. Y. Niu, A. Zlokapa, M. Broughton, S. Boixo, M. Mohseni, V. Smelyanskyi, and H. Neven, "Entangling quantum generative adversarial networks," *Phys. Rev. Lett.*, vol. 128, no. 22, Jun. 2022, Art. no. 220505.

[8] P. Gokhale, J. M. Baker, C. Duckering, N. C. Brown, K. R. Brown, and F. T. Chong, "Asymptotic improvements to quantum circuits via qutrits," in *Proc. ACM/IEEE 46th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2019, pp. 554–566.

[9] J. M. Baker, C. Duckering, and F. T. Chong, "Efficient quantum circuit decompositions via intermediate qudits," in *Proc. IEEE 50th Int. Symp. Multiple-Valued Log. (ISMVL)*, Nov. 2020, pp. 303–308.

[10] S. Pirandola, B. R. Bardhan, T. Gehring, C. Weedbrook, and S. Lloyd, "Advances in photonic quantum sensing," *Nature Photon.*, vol. 12, no. 12, pp. 724–733, Dec. 2018.

[11] B. J. Lawrie, P. D. Lett, A. M. Marino, and R. C. Pooser, "Quantum sensing with squeezed light," *ACS Photon.*, vol. 6, no. 6, pp. 1307–1318, Jun. 2019.

[12] Xanadu. (2022). *Qml.Stronglyentanglinglayers, Pennylane Documentation*. [Online]. Available: https://pennylane.readthedocs.io/en/latest/code/api/pennylane.StronglyEntanglingLayers.html

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

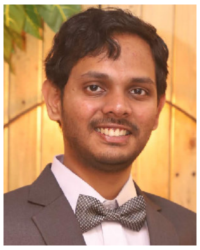
[14] V. Giovannetti, S. Lloyd, and L. Maccone, "Architectures for a quantum random access memory," *Phys. Rev. A, Gen. Phys.*, vol. 78, no. 5, Nov. 2008, Art. no. 052310.

[15] D. K. Park, I. Sinayskiy, M. Fingerhuth, F. Petruccione, and J.-K.-K. Rhee, "Parallel quantum trajectories via forking for sampling without redundancy," *New J. Phys.*, vol. 21, no. 8, Aug. 2019, Art. no. 083024.

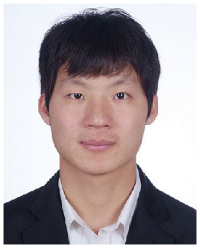
[16] C. T. Hann, C.-L. Zou, Y. Zhang, Y. Chu, R. J. Schoelkopf, S. M. Girvin, and L. Jiang, "Hardware-efficient quantum random access memory with hybrid quantum acoustic systems," *Phys. Rev. Lett.*, vol. 123, no. 25, Dec. 2019, Art. no. 250501.

[17] T. S. Madhulatha, "An overview on clustering methods," 2012, *arXiv:1205.1117*.

[18] G. M. van de Ven and A. S. Tolias, "Three scenarios for continual learning," 2019, *arXiv:1904.07734*.



KOUSTUBH PHALAK (Student Member, IEEE) received the bachelor's degree in electrical and electronics engineering from the Birla Institute of Technology and Science, Pilani, in 2020. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Pennsylvania State University. He works in the field of emerging technologies, especially quantum computing.



JUNDE LI (Student Member, IEEE) received the B.Mgt. degree in logistics management from Qingdao University, in 2015, the M.Sc. degree in engineering management from the City University of Hong Kong, in 2016, and the Ph.D. degree in computer science and engineering from The Pennsylvania State University (Penn State), in December 2022. Before joining Penn State, he was a Process Engineer with ASM Pacific Technology and an Artificial Intelligence Engineer at a local autonomous driving startup in Hong Kong. He was with ApexQubit as a part-time Quantum Machine Learning Engineer, in 2021. He was a Deep Learning Research Intern with the Bosch Center for Artificial Intelligence (BCAI), in 2022. He is currently a Machine Learning Compiler Engineer with Cadence Design Systems. His current research interests include machine learning, computer vision, neural network compression and quantization, and quantum computation and intelligence. He received the Dr. Tse-Yun Feng Graduate Student Award from Penn State.

SWAROOP GHOSH (Senior Member, IEEE) received the B.E. degree (Hons.) from IIT, Roorkee, and the Ph.D. degree from Purdue University.

He is currently an Associate Professor with The Pennsylvania State University. His research interests include quantum computing, emerging memory technologies, and hardware security.

Dr. Ghosh is a Senior Member of the National Academy of Inventors (NAI), an Associate Member of Sigma Xi, and a Distinguished Speaker of the Association for Computing Machinery (ACM). He has also served on the technical program committees for more than 25 ACM/IEEE conferences. He was a recipient of the Intel Technology and Manufacturing Group Excellence Award, the Intel Divisional Award, the two Intel Departmental Awards, the USF Outstanding Research Achievement Award, the College of Engineering Outstanding Research Achievement Award, the DARPA Young Faculty Award (YFA), the ACM SIGDA Outstanding New Faculty Award, the YFA Director's Fellowship, the Monkowsky Career Development Award, the Lutron Spira Teaching Excellence Award, the Dean's Certificate of Excellence, and the Best Paper Award in American Society of Engineering Education (ASEE). He served as the General Chair, the Conference Chair, and the Program Chair for ISQED and DAC Ph.D. Forum and the Track (Co)-Chair for DAC, CICC, ISLPED, GLSVLSI, VLSID, and ISQED. He served as an Associate Editor for the IEEE Transactions on Circuits and Systems—I: Regular Papers and IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems and as a Senior Editorial Board Member for IEEE Journal of Emerging Topics on Circuits and Systems. He served as the Guest Editor for the IEEE Journal of Emerging Topics on Circuits and Systems and IEEE Transactions on Very Large Scale Integration (VLSI) Systems.