mathematics

IMPACT FACTOR 2.3

CITESCORE 4.0

Article

Quantum K-Nearest Neighbors: Utilizing QRAM and SWAP-Test Techniques for Enhanced Performance

Alberto Maldonado-Romo, J. Yaljá Montiel-Pérez, Victor Onofre, Javier Maldonado-Romo and Juan Humberto Sossa-Azuela

Special Issue

Quantum Computing and Networking

Edited by

Prof. Dr. Jehn-Ruey Jiang and Dr. YungYu Zhang

MDPI

https://doi.org/10.3390/math12121872

mathematics

MDPI

Article

Quantum K-Nearest Neighbors: Utilizing QRAM and SWAP-Test Techniques for Enhanced Performance

Alberto Maldonado-Romo [1,2,*], J. Yaljá Montiel-Pérez [1,*], Victor Onofre [2], Javier Maldonado-Romo [3] and Juan Humberto Sossa-Azuela [1]

[1] Instituto Politécnico Nacional, Centro de Investigación en Computación, Av. Juan de Dios Bátiz S/N, Nueva Industrial Vallejo, Gustavo A. Madero, Mexico City 07738, Mexico
[2] Quantum Open Source Foundation, Toronto, ON M5V 2L1, Canada
[3] Institute of Advanced Materials for Sustainable Manufacturing, Tecnologico de Monterrey, Monterrey 64849, Mexico
* Correspondence: amaldonador2021@cic.ipn.mx (A.M.-R.); yalja@ipn.mx (J.Y.M.-P.)

Abstract: This work introduces a quantum K-Nearest Neighbor (K-NN) classifier algorithm. The algorithm utilizes angle encoding through a Quantum Random Access Memory (QRAM) using n number of qubit addresses with O(log(n)) space complexity. It incorporates Grover's algorithm and the quantum SWAP-Test to identify similar states and determine the nearest neighbors with high probability, achieving $O(\sqrt{m})$ search complexity, where m is the qubit address. We implement a simulation of the algorithm using IBM's Qiskit with GPU support, applying it to the Iris and MNIST datasets with two different angle encodings. The experiments employ multiple QRAM cell sizes (8, 16, 32, 64, 128) and perform ten trials per size. According to the performance, accuracy values in the Iris dataset range from 89.3 ± 5.78% to 94.0 ± 1.56%. The MNIST dataset's mean binary accuracy values range from 79.45 ± 18.84% to 94.00 ± 2.11% for classes 0 and 1. Additionally, a comparison of the results of this proposed approach with different state-of-the-art versions of QK-NN and the classical K-NN using Scikit-learn. This method achieves a 96.4 ± 2.22% accuracy in the Iris dataset. Finally, this proposal contributes an experimental result to the state of the art for the MNIST dataset, achieving an accuracy of 96.55 ± 2.00%. This work presents a new implementation proposal for QK-NN and conducts multiple experiments that yield more robust results than previous implementations. Although our average performance approaches still need to surpass the classic results, an experimental increase in the size of QRAM or the amount of data to encode is not achieved due to limitations. However, our results show promising improvement when considering working with more feature numbers and accommodating more data in the QRAM.

Keywords: QRAM; quantum machine learning; quantum classification; quantum K-Nearest Neighbor

MSC: 68Q12

Citation: Maldonado-Romo, A.; Montiel-Pérez, J.Y.; Onofre, V.; Maldonado-Romo, J.; Sossa-Azuela, J.H. Quantum K-Nearest Neighbors: Utilizing QRAM and SWAP-Test Techniques for Enhanced Performance. *Mathematics* **2024**, *12*, 1872. https://doi.org/10.3390/math12121872

Academic Editors: Jehn-Ruey Jiang and YungYu Zhang

Received: 30 April 2024
Revised: 4 June 2024
Accepted: 13 June 2024
Published: 16 June 2024

1. Introduction

Machine learning (ML) is a set of algorithms and tools used to acquire information or optimize actions based on data [1]. It has witnessed substantial recent growth, primarily due to its versatile applications across diverse industries [2]. ML techniques, encompassing classification [3,4], prediction [5,6], and data generation [7,8], have been instrumental in tasks such as image classification [9,10], forecasting financial systems [11], and generating text through natural language processing [12].

These technological advancements demand substantial computational capacity and optimal performance is expensive. Working with large datasets in deep learning and natural language processing necessitates significant computational resources, thus motivating exploration into different paradigms, such as quantum computing. Quantum algorithms have shown the potential to be exponentially more efficient than classical algorithms for some

problems. Various domains of computer science may benefit from quantum algorithms and their properties. ML stands out due to the potential for performance improvements through reducing the temporal or spatial complexity for important subroutines [13,14]. Classical data may be encoded in quantum states, for example, through qubit rotations. These encodings enable the study of quantum machine learning (QML) algorithms using classical benchmark datasets. Several algorithms have been proposed in this new paradigm.These include quantum principal component analysis [15,16], quantum Support Vector Machines [17], quantum nearest-neighbor classification [18,19], and quantum neural networks [20,21]. The variety of algorithms in QML showcases the richness and potential of this emerging field.

ML consists of three main approaches: supervised learning, unsupervised learning, and reinforcement learning. This study focuses on an example of supervised learning [22,23]. The goal is to train an ML algorithm that employs a labeled dataset to label a data point from outside the training set but sampled from the same underlying distribution. This approach is known as classification, and it can be performed by the K-Nearest Neighbor (K-NN) [24] algorithm based on a similarity or distance metric. The method is one of the best-known classification algorithms in supervised learning [25]. It is also possible to use a K-NN for unsupervised learning tasks. Based on the theory provided in [14], Quantum Nearest Neighbors is almost quadratically more efficient than the classical analog, even with sparse data, which is useful when working with large amounts of data in a short time.

This work describes a novel Quantum K-Nearest Neighbor (QK-NN) algorithm based on data stored in a Quantum Random Access Memory (QRAM) [26–28]. The QRAM incorporates address and data qubits, leveraging quantum superposition. Each data cell represents a leaf of a binary tree, indexed by the address qubits. Our method computes the required distance metric based on the SWAP-Test, which offers a similarity metric for comparing two quantum states. We use Grover's algorithm [26,29–32] to efficiently identify states with high similarity scores by utilizing an oracle. Quantum states from the labeled dataset are analyzed using the SWAP-Test [33,34] via a mid-circuit measurement. This allows our quantum oracle to search for the state with the highest similarity to an unlabeled test example. It is important to note that our results, as is typical with quantum algorithms, are probabilistic. The proposed approach involves storing a sample, or the entire dataset, in QRAM. Properly speaking, our K-NN is not trained; rather, it references the labeled dataset at inference time. In keeping with standard ML parlance, we refer to the data used as "training" data. In the work presented here, we use the Iris dataset (distributed with Scikit-learn [35]) and MNIST [36] as the demonstrators. The experiment format involves creating different sizes of QRAM to store subsets of the dataset until reaching a maximum of 128, which is the feasible capacity for simulation. Also, with the size of 128, different numbers of nearest neighbors are considered, considering the maximum of 13 neighbors. Preprocessing is conducted on the datasets to assess the classification capability of our QK-NN proposal, taking into account information loss and working with two datasets with different types of features and several attributes.

Note, that a noiseless quantum computer is considered in this work, as the circuit depths would be prohibitive on a noisy device. The implementation and development are intended to be added to a model with noise or hardware when the capacity to handle the required depth becomes available. However, dealing with the limitations of NISQ devices is beyond the scope of this work. Given these constraints, the experiments are conducted via classical quantum computer simulation. By studying performance trends with an increasing qubit count, it is possible to speculate about the potential usefulness of this approach as quantum computers push beyond classical regimes. It is important to note that these speculations are based on the current understanding and may change as the field evolves.

The structure of the rest of the paper is as follows: Section 2, describes prior and related work. Section 3 describes the conceptual design and implementation details of our quantum classifier algorithm. Section 4 describes the experimental results from a Qiskit

simulation of a quantum computer. Then, Section 5 discusses our results. Finally, Section 6 presents the conclusions and offers some directions for future research.

2. Related Work

Multiple studies of quantum K-NNs exist in the literature. Here, we review the most recent results, and those contributions closest to this work.

In [26], a recommendation system employing a Quantum K-NN leveraging Grover's algorithm for data processing is proposed, utilizing the Hamming distance instead of the SWAP-Test for a similarity metric. This system includes a classical database constructed using basis encoding and two sampled quantum states for comparison using the quantum algorithm. Additionally, an auxiliary qubit is incorporated to probabilistically amplify recommended elements via Grover's algorithm, thereby enhancing recommendation accuracy.

Another quantum version of the K-NN classifier is proposed in [29]. In this paper, the entire dataset was incorporated into a quantum circuit. The best cases were compared using the Hamming distance and sorted accordingly with a quantum operator.

Reference [37] introduces a QK-NN algorithm based on Mahalanobis distance, which is a multivariate generalization of the square of a standard Z score, i.e., how many standard deviations a point is away from the mean. The authors propose to store the Mahalanobis distance between states in QRAM. They then sample from the collection to obtain the K nearest neighbors using Grover's algorithm.

In [38] the authors introduce a new similarity measure named "polar distance," inspired by the polar coordinate system and quantum properties. This measure considers both angular (phase) and modulus (length) information, with an adjustable weight parameter tailored to specific datasets, they use a classical binary search with the Grover algorithm. The complexity time is $\mathrm{O}\left(\sqrt{m}\right)$.

In [39] the authors present a quantum circuit for K-NN classification that quantizes both neighbor and K value selection processes simultaneously. Their algorithm utilizes least squares loss and sparse regularization to identify the optimal K values and K nearest neighbors. It employs a novel quantum circuit that combines quantum phase estimation, controlled rotation, and inverse phase estimation techniques.

Reference [40] utilizes the SWAP-Test to compute scalar products between feature vectors encoded according to the Mottonen method, representing amplitude encoding in quantum states. This work is the most direct comparison of our proposed QK-NN model due to the similarities between the SWAP-Test and the Grover algorithm because it applies the Quantum Minimization Algorithm (QMA) to determine the K closest neighbors for each test vector. This method also involves employing Grover's algorithm to find the closest neighbor and conducting a specific number of SWAP-Test iterations. Likewise, a similar QMA process is used to enhance accuracy. The time complexity of the Quantum K-NN algorithm is $\mathrm{O}(nm(k + \log(d)))$ where n is the training set size, m is the test set size, k is the number of clusters and d is the space dimension, enabling a potential speed-up for large vector dimensions. Therefore, the number of SWAP-test iterations and QMA iterations influences the efficiency of the QK-NN algorithm.

Finally, Ref. [41] has various K-NN approaches, relying on the inner product and Euclidean distance calculations to identify the nearest neighbor or class centroid. Specifically, the focus is on the utilization of Euclidean distance. While using QRAM and the SWAP-test is mentioned, an intermediate measurement is lacking. Instead, Amplitude Estimation (AE) is employed, similar to the algorithm proposed in reference [38], albeit without QRAM. Their proposal is notable for its scaling of the number of queries, which is proportional to $O\left(\sqrt{M}\log(M)\right)$ rather than M.

Our Contribution

Our results are comparable to recent state-of-the-art quantum algorithms, see, for example, [38] with an accuracy of 95.82% and [29] which found an accuracy of 94.66%. The notable points of this work are as follows:

- Our primary novelty is in the combination of high-performing quantum subroutines along with an empirical study of scaling performance with available qubit count for the QRAM.
- We illustrate how mid-circuit measurements in the SWAP-Test as an oracle support Grover's algorithm, which achieves high accuracy in identifying similar states.
- The algorithm allows for the creation of a quantum circuit for a subset of data with a spatial complexity of $O(\log(n))$ and a temporal complexity of $O(\sqrt{m})$.
- We conducted a series of experiments with ten seeds to demonstrate the benefits of our proposal on the Iris and MNIST datasets, aiming to highlight the limitations of our QK-NN approach.
- We have enhanced the state-of-the-art experimental results using the MNIST dataset, achieving a notable performance of $94.00\% \pm 2.11$.
- We provide repeatable numerical experiments through a structured workflow on the Iris and MNIST datasets. Our code is available online (https://github.com/MaldoAlberto/QKnn, accessed on 21 April 2024).

3. Materials and Methods

This section presents the methods employed in this research. Our QK-NN consists of three primary steps. First, we utilize QRAM for intermediate storage and implement the SWAP-Test for a similarity metric. Second, we employ Grover's algorithm to find the solution with the highest similarity with high probability. Finally, classical post-processing reads the address of the QRAM to determine the identities of the K-NN classes. Each point is described in more detail below.

3.1. Review of the Classical K-NN Algorithm

The classical K-NN classification algorithm is widely used in supervised learning [27,28]. It utilizes two datasets: a training set, where each sample has an associated label and a test set that contains unlabeled samples we wish to classify. For each example from the test set, the algorithm deploys a similarity metric, e.g., a distance metric in feature space (see Section 3.5) to compare the example to each member of the training set. Labels are selected according to a voting rule over K of the nearest neighbors in metric distance space, where the parameter K is an odd hyperparameter so that one class is always more significant than the other according to the pigeonhole principle.

For illustration, Figure 1 describes two classes in our training set, represented as blue and green circles. The method determines which class an unknown example, represented by a red point in Figure 1, belongs to. The distance metric is applied between the unlabeled example and the samples in the training set. With $K = 5$, the nearest neighbor set is observed to contain two green circles and three blue circles. Therefore, by majority vote, the unknown example is classified as blue.

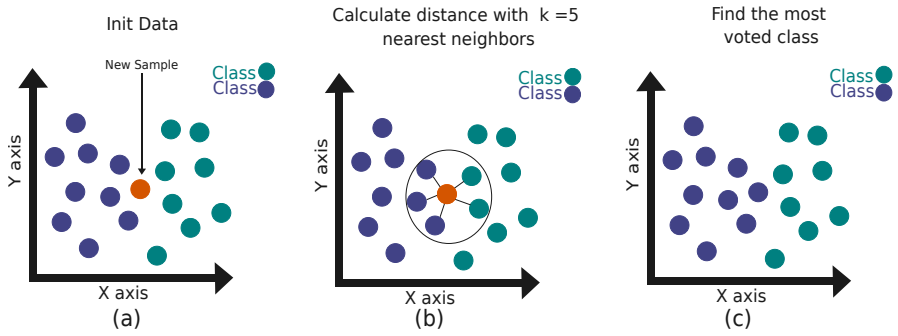


Figure 1. Example of K-NN when $K = 5$: (**a**) An unknown instance (orange circle). (**b**) Find the $K = 5$ nearest neighbors. (**c**) Determine which class predominates by our voting rule (in this case, majority vote).

The steps of a classical K-NN are listed in Algorithm 1.

Algorithm 1 Classical K-NN

1: Designate a labeled training set and an unlabeled test set.
2: **for** each event in the test set **do**
3: **for** each event in the training set **do**
4: Compute the distance between the test event and the training event.
5: **end for**
6: Sort the events in the training set by metric distance.
7: Choose the K nearest neighbors to the test event.
8: The class with highest number of nearest neighbors among the K selected examples is assigned to the test event.
9: **end for**

3.2. Quantum Data Encoding and Preprocessing

In order to utilize a quantum computer to analyze classical data, we must first encode the data as quantum information. To conduct this we apply the transformation $\Psi : \Theta \rightarrow \mathcal{H}$, where $\mathcal{H}$ is the Hilbert space with terms $|\Psi(\Theta)\rangle = |\psi(\theta_0)\rangle, |\psi(\theta_1\rangle, \ldots, |\psi(\theta_n)\rangle$ [42,43]. To represent classical data in a state vector, a set of $|0\rangle^{\otimes n}$ qubits is transformed using a unitary matrix $U(\theta)$ to obtain $|\Psi(\Theta)\rangle$[42]. Algorithms may use select from basis, amplitude, angle, or arbitrary encoding. The algorithm in this paper demonstrates an angle encoding scheme, but in principle, other parameterized circuit methods may also be successful.

3.2.1. Angle Encoding

Angle encoding uses N classical features as rotation angles for n qubits, where $n \leq N \leq 2n$ [42,43] (each qubit may, in principle, support two rotations). A general encoding using single-qubit parameterized gates with one qubit per feature, $\mathrm{U}(\theta)$, may be written as [42,43]:

$$S_{x_j} = \bigotimes_{i=1}^{n=N} \mathrm{U}\left(\theta_j^{(i)}\right), \tag{1}$$

where j indexes examples, θ_j is the corresponding classical data, and i indexes qubits. Our method chooses parameterized rotation gates about Y or Z on the Bloch sphere—$\mathrm{RY}(\theta)$ or $\mathrm{RZ}(\theta)$, respectively, for the data encoding, defined in Equations (2) and (3):

$$\mathrm{RY}(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}, \tag{2}$$

$$\mathrm{RZ}(\theta) = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}. \tag{3}$$

The quantum circuits encoding n features in n qubits are shown in Figure 2. In the case of RZ gates, we must prefix the circuit with a Hadamard gate (H) to transform the basis from $\{|0\rangle, |1\rangle\}$ to $\{|+\rangle, |-\rangle\}$.

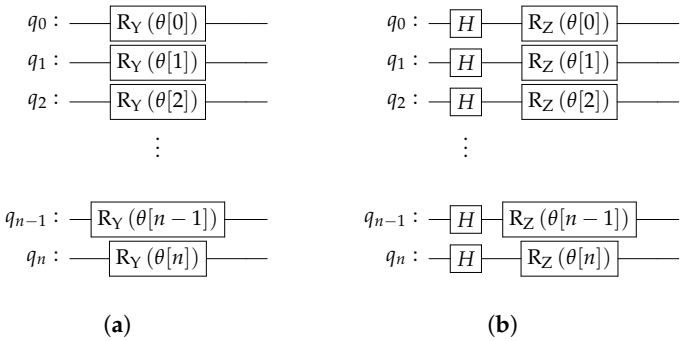


Figure 2. Illustration of the quantum circuits for angle encoding (**a**) using RY gates, and (**b**) using RZ gates (prefixed by a Hadamard gate, H).

3.2.2. Preproccesing

Datasets may contain numerical or categorical values. We convert binary categorical values into π or 2π, generating complete or zero-rotation gates. In the event of multiple classes, it is appropriate to use a "one-hot" encoding method, which requires more qubits. Standard classical machine learning practices such as k-fold cross-validation, leave-one-out, or hold-out data selection may be employed in constructing training and testing datasets. To verify the proposed model's classification accuracy when applied to unlabeled data, k-fold cross-validation, and hold-out samples should be made using random shuffles.

Our initial preprocessing step is to determine the necessary number of features, denoted as N. We choose here to assign one qubit per feature. In other cases, due to limitations on qubit availability in contemporary hardware, dimensionality reduction techniques such as principal component analysis (PCA) [44] may be needed. Angle encoding requires angles between 0 and π when using RY gates, and between 0 and 2π when using RZ gates. Therefore, normalization might be necessary depending on the dataset and quantum rotation gates utilized.

3.3. QRAM

A storage array, an address register, and an output register define classical RAM. Each cell within the array corresponds to a distinct numeric address. Upon initialization, the register holds the address of a memory cell, and subsequently, the content is transferred to the output register. QRAM shares the fundamental components of classical RAM, with the distinction that qubits comprise the address and output registers [27]. Through address superposition, QRAM can access multiple memory cells simultaneously. The address register may effectively store 2^m values, where m is the number of qubits in the register, as shown in Equation (4):

$$\frac{1}{\sqrt{2^m}} \sum_{j=0}^{2^m-1} |j\rangle |0\rangle^{\otimes N} \xrightarrow{\text{QRAM}} \frac{1}{\sqrt{2^m}} \sum_{j=0}^{2^m-1} |j\rangle |\text{ Data }_j\rangle, \tag{4}$$

where j indexes the data in the QRAM and $|\text{Data}_j\rangle$ is the m-qubit content of the j-th memory cell [27]. The address structure may be represented as a binary tree as shown, for example, in Figure 3, which shows an example encoding for the QRAM of three qubits. The state of q_0 is $|0\rangle$, and the states q_1 and q_2 are $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, giving a probability of $\frac{1}{4}$ for each of the states $|000\rangle, |001\rangle, |010\rangle$ and $|011\rangle$. This encodes binary values as values 0, 1, 2, and 3, respectively.

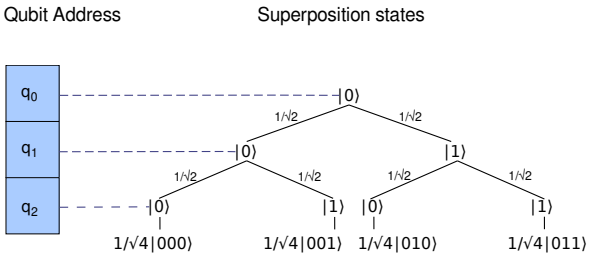


Figure 3. A QRAM example encoding four address values with three qubits.

The QRAM address size is constrained to base-two values such as 2, 4, 8, 16, 32, 64, 128, and so forth. If the classical data indices do not fill the vector, the remaining cells are padded with zero-values.

3.3.1. Encoding the Dataset in QRAM

To encode the training dataset, we begin with m address qubits initialized into $|+\rangle^{\otimes m}$, an ancilla qubit in $|0\rangle$, and $N = n$ data qubits (Here, we assume we have an equal number of data qubits and features. If this is not practical, we may use dimensionality reduction techniques to reduce the required qubit count or different encoding schemes) initialized into $|0\rangle^{\otimes N}$. Note, that given a fixed number of features N, increasing the number of qubits for QRAM increases the address space and the number of training events we may encode. The full QRAM with address qubits, our ancilla (for operations) and the training data qubits is $|+\rangle^{\otimes m}|0\rangle|0\rangle^{\otimes N}$ or $|+\rangle^{\otimes m}|0\rangle^{\otimes N+1}$. The encoding procedure involves preparing all of the qubits, and then executing an operation on the address block to prime the ancilla followed by an operation on the data qubits according to the state of the ancilla in a loop over all of the training data. We will explain these operations individually in the following paragraphs and then explain how the combined circuit is constructed.

To build the addresses for the linked data blocks, we employ the Multicontrolled X (MCX) gate, as described in [31]. We will write MCX_m to indicate the number of control qubits m. In this work, m also refers to the number of address qubits available. An example MCX_m is the MCX_2, which is the Toffoli gate (or CCX gate). It involves two control qubits and one target qubit that activates upon meeting the conditions of the control qubits. By default, the control qubits are set to the state $|1\rangle$, and we apply an X gate if we need the state $|0\rangle$. This MCX then prepares the ancillary qubit. The address values are derived from a binary value representing a basis encoding. For example, cell 2 is 0b10, or $|10\rangle$, meaning that the MCX gate will be applied with qubit 0 in the state $|1\rangle$ and qubit 1 in the state $|0\rangle$. We can write this operation generally in Expression (5):

$$MCX_m[H^{\otimes m}|0\rangle \otimes |0\rangle_{ancilla}], \tag{5}$$

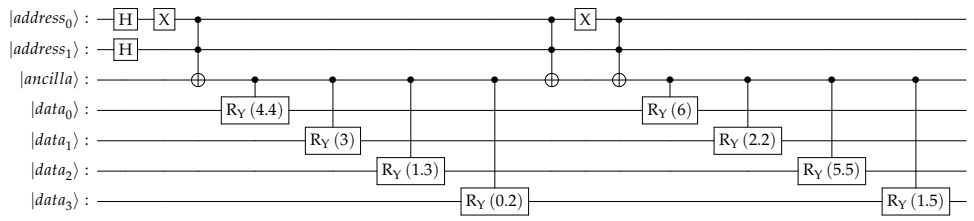
to indicate how the address block and ancilla are prepared.

The prepared ancilla qubit then activates the controlled rotation gates RY or RZ (using one or the other depending on the chosen encoding), to encode the correctly addressed training example, as described in Expression (6):

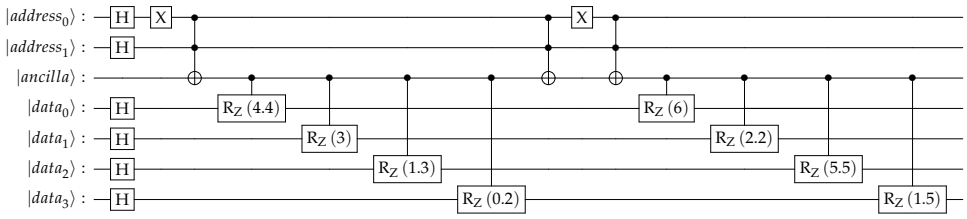
$$\sum_{i=0}^{p}\sum_{j=0}^{N} \text{CRY}(x_{ij})[|\phi\rangle_{\text{ancilla}}|0\rangle_i^{\otimes N}], \text{ or } \sum_{i=0}^{p}\sum_{j=0}^{N} \text{CRZ}(x_{ij})[|\phi\rangle_{\text{ancilla}}H^{\otimes N}|0\rangle_i^{\otimes N}], \tag{6}$$

where p is the size of the training dataset, N is the number of features (and $N = n$ qubits in the data register), x_{ij} provides the value of feature j in example i, CRY is a controlled-RY, and CRZ is a controlled-RZ. If $p < 2^m - 1$, the complex conjugate of the element of the test set to be compared is added to the empty cell address. Otherwise, the MCX and X gates used to generate the binary state are reapplied to avoid affecting other states.

See Figure 4 for an illustration of the address and data encoding combined into a circuit for two data examples.



(**a**)



(**b**)

Figure 4. A sample of QRAM with the angle encoding of the RY and RZ for qubit address $|01\rangle$ and $|11\rangle$. (**a**) QRAM using CRY gates. (**b**) QRAM using CRZ gates.

3.4. Grover's Algorithm

Grover's algorithm aims to search for a value within an unstructured database of size p. Classical algorithms, whether deterministic or probabilistic, require inspecting at least $\frac{p}{2}$ values with a time complexity of $O(p)$ to achieve a probability of finding the value of interest exceeding 50%. In contrast, Grover's algorithm can retrieve the desired value in $O(\sqrt{p})$ steps [45].

Our goal is to find which of the p examples in our training set have the highest similarity to a test event using an unstructured search. To run Grover's quantum algorithm, in general, we begin with initial state $|\psi\rangle = H^{\otimes n}|0\rangle$ and an oracle, U_ω, that operates as shown in Equation (7).

$$U_\omega|\psi\rangle = \begin{cases} |\psi\rangle \text{ if } \psi \neq \omega \\ -|\psi\rangle \text{ if } \psi = \omega \end{cases} \tag{7}$$

where ω is the value being searched for.

We next consider a function f that takes a proposed solution ψ, and returns $f(\psi) = 0$ if $\psi \neq \omega$ and $f(\psi) = 1$ for $\psi = \omega$. Then the oracle is expressed as shown in Equation (8):

$$U_\omega|x\rangle = (-1)^{f(x)}|x\rangle. \tag{8}$$

This transformation means that the amplitude of the $|w\rangle$ state becomes negative over the states, as is described in [26,29–31]. Then, using a reflection operation referred to as a "diffuser" [26,29–32] causes the positive values to move to an amplitude of 0 and the negative values to move to an amplitude of 1, as shown in Equation (9):

$$2|\psi\rangle\langle\psi| - I_N = H^{\otimes n}(2|0\rangle\langle 0| - I_N)H^{\otimes n}, \tag{9}$$

where $|\psi\rangle$ is the uniform superposition of states and I_N is the identity matrix of dimension N. Since $2|\psi\rangle\langle\psi| - I_N$ is a reflection about $|\psi\rangle$, $2|0\rangle\langle 0| - I_N$ is a reflection about $|0\rangle$. The Grover diffuser is implemented in a quantum circuit using a phase-shifting operator that negates the ground state [32].

3.5. SWAP-Test

An important subroutine of the K-NN algorithm is to calculate the distance metric between two examples. The classical algorithm frequently employs the Euclidean distance for this metric, as defined in Equation (10) [34]:

$$|\mathbf{a} - \mathbf{b}| = \sqrt{\sum_{i=1}^{m} (a_i - b_i)^2} \tag{10}$$

where i indexes a sum over features, and $\mathbf{a}$ and $\mathbf{b}$ are two examples.

In quantum computing, a similar metric may be derived using the inner product to find the state overlap, denoted as $\mathrm{F} = |\langle \psi \mid \phi \rangle|^2$. A quantum subroutine known as the SWAP-Test [34] can compute this between two pure states $|\psi\rangle$ and $|\phi\rangle$. For our purposes, $|\psi\rangle$ and $|\phi\rangle$ are the states generated by our encoding algorithm for two examples. The SWAP-Test, illustrated in Figure 5 for a three-qubit example, in general, requires a minimum of $2n + 1$ qubits: one ancilla qubit and n each for the two states being compared: $|\psi\rangle, |\phi\rangle$.

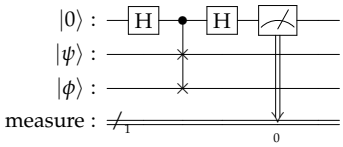


Figure 5. The quantum circuit for the SWAP-Test on two states, each encoded into one qubit.

The quantum circuit for the SWAP-Test consists of an ancilla initialized to the state $|0\rangle$, and qubits to hold the states $|\psi\rangle$ and $|\phi\rangle$. A Hadamard gate is first applied to the ancilla qubit. Subsequently, a controlled SWAP operation is performed, with the ancilla qubit as the control. Finally, another Hadamard gate is applied to the ancilla, and then it is measured. The measurement of the ancilla qubit is repeated s times to obtain the fidelity between the two states. The quantum state after running the gates in the circuit, but before measurement, is expressed in Equation (11):

$$\frac{1}{2}|0\rangle(|\phi\rangle|\psi\rangle + |\psi\rangle|\phi\rangle) + \frac{1}{2}|1\rangle(|\phi\rangle|\psi\rangle - |\psi\rangle|\phi\rangle). \tag{11}$$

Following the measurement, the probability of the first qubit being found in the $|1\rangle$ state is $\frac{1}{2} - \frac{1}{2}|\langle\phi|\psi\rangle|^2$. When the probability of finding $|0\rangle$ is 100%, it indicates that the two states are identical; however, if the probability is 50% $|0\rangle$ and 50% $|1\rangle$, it signifies that the states are orthogonal. Note, that to estimate the error on the overlap to within a factor of ϵ, we require $\mathrm{O}\left(1/\epsilon^2\right)$ copies of the state.

When our states require more than one qubit for encoding the controlled-SWAP follows a specific pair-wise pattern. We maintain the same ancilla for control and create a sequential cascade with different target qubits: one from $|\psi\rangle$ and one from $|\phi\rangle$, as depicted in Figure 6. Each pair of qubits, matched by index, undergoes a controlled SWAP operation, with the ancilla qubit consistently serving as the control. The circuit begins and ends with Hadamard gates applied to the ancilla qubit [33].

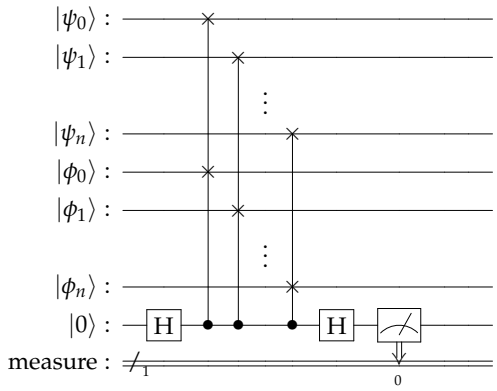


Figure 6. Example of a SWAP-Test with multiple qubit states.

3.6. Quantum K-NN

The QK-NN is based on three critical components: data storage in QRAM, the similarity metric from the SWAP-Test quantum circuit, and utilizing Grover's algorithm to select the most similar states with high probability, as shown in Figure 7, which combines all the elements of our approach into one circuit. Theoretically, we should run Grover's algorithm $\sqrt{p}$ times to search the QRAM for the matching similarity event. However, we empirically found good results with many fewer calls. We found searching $\sqrt{m}$ times to produce a set of nearest neighbors, where m is the number of qubit addresses in the QRAM, to be effective. The result of each run is a probabilistic estimate of the most similar event from the training dataset. It is not guaranteed to be unique and $\sqrt{m}$ calls are not guaranteed to find the K most similar events. We empirically observe in the experiments discussed below that the algorithm samples from the events in a way consistent with the performance of other KNN implementations.

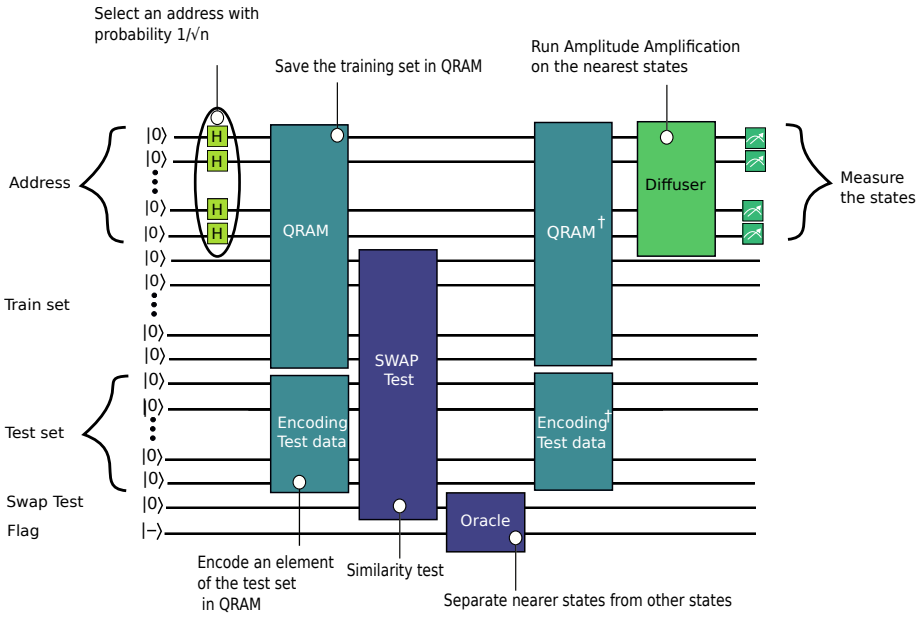


Figure 7. Overview of the quantum circuit for the K-nearest neighbors algorithm.

The performance of the proposed classifier is assessed through a mean accuracy metric computed from R runs. We propagate statistical uncertainties from shot noise through the average to the end results.

The QK-NN algorithm is presented step-by-step in Algorithm 2.

Algorithm 2 Quantum K-NN

Designate a labeled training set and an unlabeled test set.
Apply angle encoding to the features of both datasets.
for each event in the test set **do**
    for 1 ... $\sqrt{m}$ (Grover iterations) **do**
        for S shots **do**
            Load the training set into QRAM.
            Initialize the oracle qubit to the state $|-\rangle$.
            Compute the SWAP-Test between the test event and training data encoded as a superposed state in the data qubits portion of the QRAM.
            Apply an intermediate Measure in the SWAP-Test qubit.
            Apply the X-gate to the oracle qubit when the intermediate measurement of the SWAP-Test is 0.
            Apply the transposed complex-conjugate of the QRAM and the diffuser to the address qubits.
            Measure the QRAM address qubits.
        end for
        The states measured in the address qubits of the QRAM should be converted from binary to decimal to compute the index of the example in the dataset.
        Record the index of most similar state after S shots of the circuit. Note, that if we find an empty address we throw the result out and repeat the iteration.
    end for
    Sort and select the K nearest neighbor candidates with the highest similarity scores as compared to the test event.
    Vote over the K candidates to find the class of the current test event.
end for

3.7. Utilization Procedure Summary

To summarize this section, for our first step, we preprocess the dataset classically and divide it into training and test sets. This preprocessing step involves any necessary dimensionality reduction and/or normalization of variables for encoding purposes. Subsequently, the data are transferred to a QRAM and subjected to the SWAP-Test with an intermediate measurement in the SWAP-Test qubit. Following this, Grover's algorithm is utilized on the SWAP-Test value to ascertain the indices of the samples with the highest probability of overlap with the test sample, thereby identifying the K nearest neighbors. This method is depicted in Figure 8, where the five basic steps are as follows:

1. Define the dataset: A dataset is selected and divided into (labeled) train and (unlabeled) test sets.
2. Execute required preprocessing: Any required classical preprocessing to utilize the features of the events is applied at this stage.
3. Run the Quantum K-NN: The proposed quantum nearest neighbor algorithm, generated by a test on QRAM, with high overlap events selected using Grover's algorithm, is employed next.
4. Perform postprocessing: This step consists of measuring the set of address qubits of the QRAM memory and selecting the nearest neighbors.
5. Compute the accuracy: After each example is labeled we can compute the accuracy of the algorithm.

It is important to highlight that when obtaining states with higher probability, only the top-k states activated with a 1 are considered of interest. Subsequently, these states

are sorted in descending order until reaching k neighbors. This process entails a time complexity of $O(\sqrt{m})$, where m represents the number of address qubits used in Grover iterations, and k denotes the number of iterations required for the selection process. In this selection process, only half of the results are utilized, from which half of the states are chosen as the k neighbors.

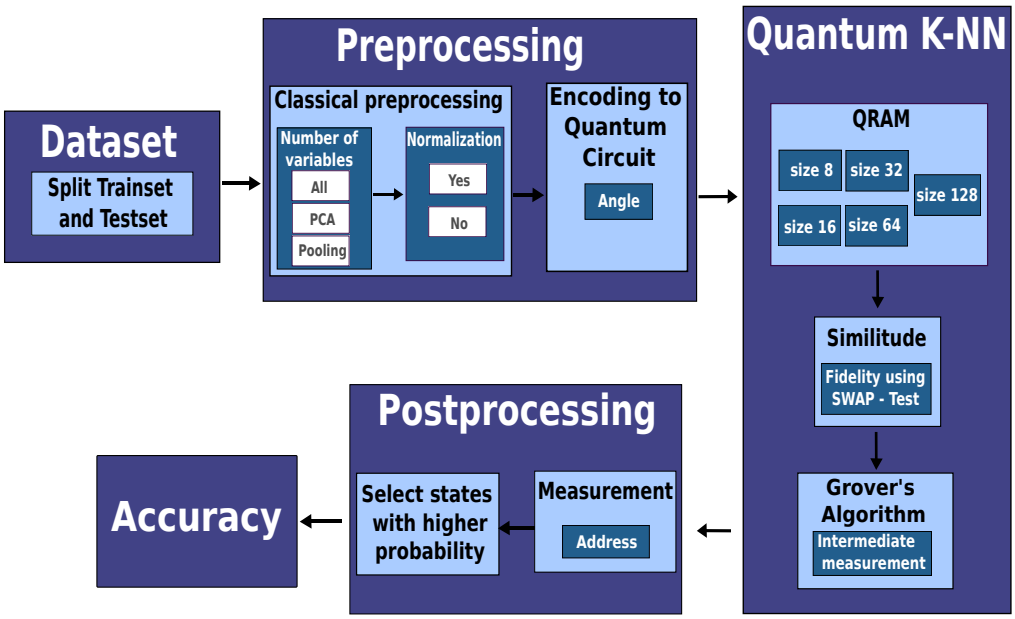


Figure 8. The proposal described in five blocks, labeled: Dataset, Preprocessing, Quantum K-NN, Postprocessing and Accuracy.

4. Experimental Results

We performed a series of simulated experiments (all experiments are conducted via classical simulation of a noiseless quantum computer) using the IBM Qiskit [46] framework on the Iris Dataset [47] and MNIST Dataset [36]. The Iris dataset with 150 events distributed over three classes (iris Setosa, iris Virginica, and iris Versicolor, each with 50 samples) and four features per example. In the case of the MNIST Dataset, there are 183 events of two classes (0 and 1) with two features, for example.

The QRAM for event index and feature encoding was built with either the CRY or the CRZ angle encoding, depending on the experiment, as described in Section 3.3.1, where each rotation represents a feature of each event.

We used a hold-out 70–30% validation method to obtain a training set of 105 samples, with 45 samples for the test set for the Iris Dataset and 128 train set samples, with 55 samples for the test set for the MNIST Dataset. Each experiment was run with state vector simulation and then sampled with 10,000 shots. We repeated each experiment for ten trials or runs. The set of training events was re-drawn with new random seeds for each trial. The same set of events was used across all algorithms for each given limit set by the QRAM size under consideration in any given trial run, i.e., we choose a trial dataset, repeat all the different configurations of the classifier, and then choose a new trial dataset for a total of ten trials. The Qiskit Aer employs the GPU-based simulator among other features too. These variables are illustrated in Figure 9, identifying the variables and actions performed in each experiment.

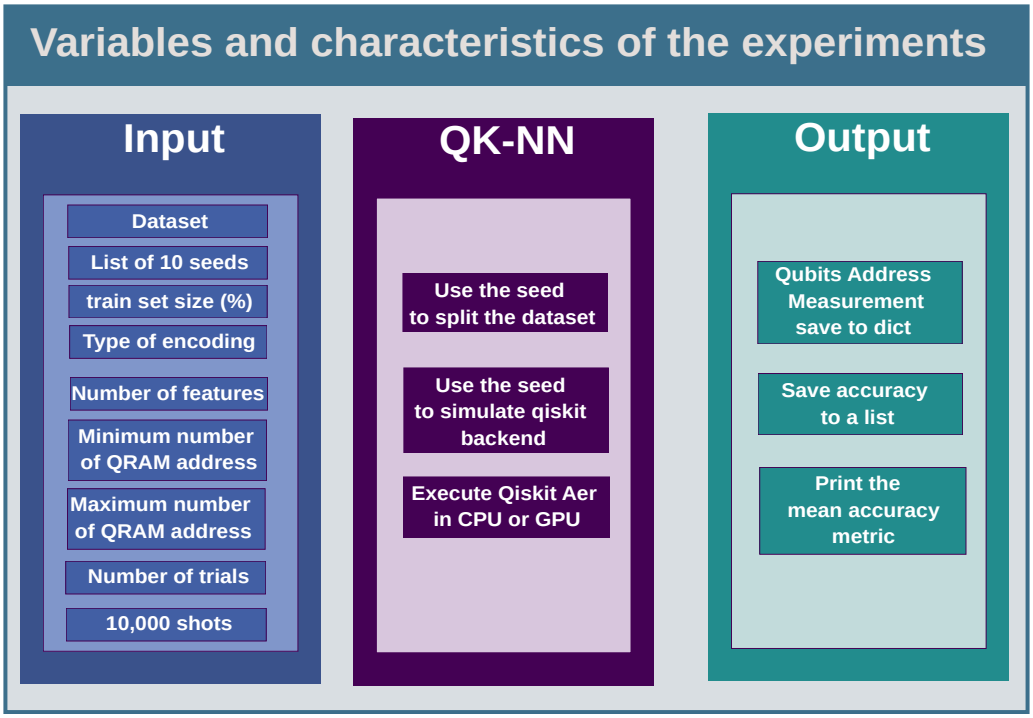


Figure 9. The list of variables and characteristics of the experiments conducted in the input, during the experiment, and post-processing for its output.

4.1. Experiments

We created two groups of experiments to characterize the performance of the algorithm. The first group varied QRAM size—which, in turn, varied the size of the training dataset—and compared several different encoding strategies for both the classical and quantum K-NNs. Note, that the classical K-NNs used the same restricted-size dataset to match the events loadable into the QRAM. The results from the first group may be found in Tables 1 and 2 for Iris Dataset and Table 3 for MNIST Dataset. The second group varied the K-value for the nearest neighbors algorithm with QRAM fixed and studied different classical and quantum encoding strategies. The results from the second group may be found in Table 4.

When the addressable size of the QRAM exceeded the number of events in the total training sample, in the case of the Iris dataset, as a post-processing step, we ignored experiments that returned an index for the highest similarity between 105 and 127 and simply repeated the experiment. We used the same PCA transform on the data in all cases when it was applied. Note, that the label "PCA = 1" on any column in the tables means the results in that column are from the dataset compressed to one feature by the PCA algorithm (shown in Figure 10).

For the MNIST dataset used in our proposal, we use only classes 0 and 1, following a series of preprocessing steps illustrated in Figure 11. Firstly, we normalize the dataset from 0 to 255 to a range of 0 to 1 and apply filter data if the pixel value must be greater than 0.75; otherwise, it is 0. Next, image resizing using the bilinear method transforms the original 28×28 image to 16×16. Subsequently, an average pooling of size 6×6 is applied, resulting in a 2×2 image.

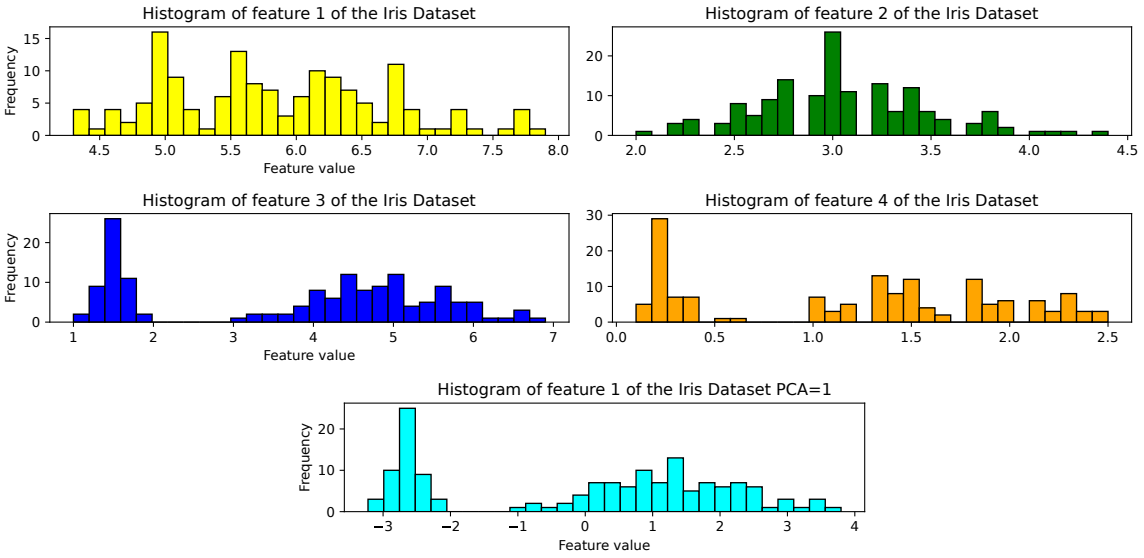


Figure 10. Robustness distribution of the 4 pixels resulting from the preprocessing of classes 0 and 1.

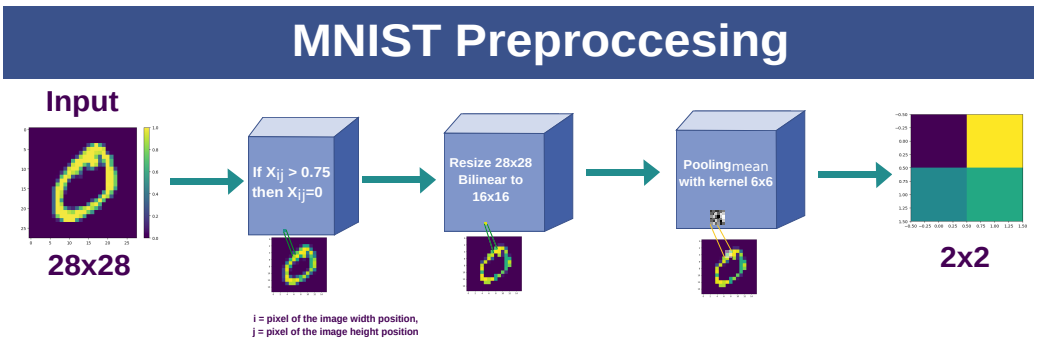


Figure 11. Preprocessing for MNIST dataset.

An analysis reveals that pixels 0 and 1 and 2 and 3 tend to have a similar data distribution. Therefore, we only use the data from pixels 1 and 2, as depicted in Figure 12.

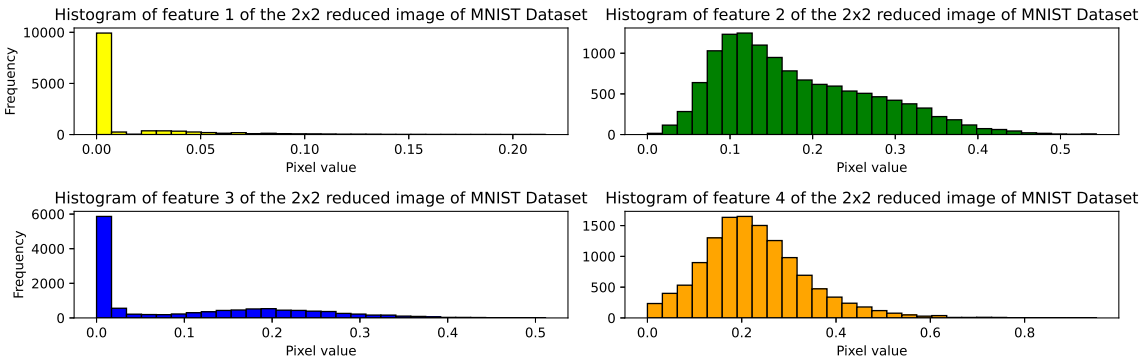


Figure 12. Robustness distribution of the 4 pixels resulting from the preprocessing of classes 0 and 1.

4.1.1. Experiments Group 1

The first group of experiments is performed using a QRAM memory of sizes 8, 16, 32, 64, and 128, for the case of the Iris dataset with non-zero occupied cells up to 105 examples. For all experiments in group one, we set K = 1. The number of qubits required for each experiment depends on the size of the QRAM and may be expressed by Equation (12):

$$3 + m + 2n, \tag{12}$$

where three is a constant value for all the required ancilla qubits, m is the number of address qubits, encoding 2^m different example addresses in QRAM, and n is the number of features per sample in both the train and test sets. Therefore, for Iris dataset experiments, the single-feature (PCA-reduced) circuit used 12 qubits, while 18 qubits were required to use four features. The MNIST dataset experiments have 183 events of two classes, 0 and 1, of two features that require 14 qubits.

Table 1 displays the mean accuracy achieved over 10 runs with various QRAM sizes with the Iris dataset. It shows the QKNN with RY encoding and two classical counterparts: one using the Euclidean distance on raw features and one using PCA to reduce the set of features to a single feature. The classical and quantum datasets were matched and the sampling was fixed for each trial at each QRAM size. The quoted uncertainties include statistical uncertainty from dataset sampling, and the quantum result includes state vector sampling, propagated through the calculation.

Table 1. Here, we show the results for experiment group one, part one of the Iris Dataset. The QRAM size controls the number of examples in the training set for both the classical and quantum algorithms, and sample draws were shared across algorithms for each experimental trial. For all experiments in group one, we set K = 1. All values are in percent.

	Mean Accuracy (%) for K = 1 NN's Using the Iris Dataset		
QRAM Size	**Classical**	**Classical** $_{\text{PCA = 1}}$	**RY**
8	87.6 ± 7.20	84.4 ± 6.67	89.3 ± 5.78
16	91.3 ± 4.62	88.7 ± 4.27	89.8 ± 6.04
32	95.3 ± 3.42	87.1 ± 5.33	92.2 ± 5.33
64	95.8 ± 2.93	90.4 ± 2.58	93.8 ± 3.38
128	96.4 ± 2.22	90.2 ± 2.22	94.0 ± 1.56

Table 2 expands the results by including the RZ encoding, as well as considering the cases where we used PCA to reduce the data to a single feature.

Table 2. Here, we show the results for experiment group one, part two of the Iris Dataset. The QRAM size controls the number of examples in the training set for both the classical and quantum algorithms and sample draws were shared across algorithms for each experimental trial. For all experiments in group one, we set $K = 1$. All values are in percent.

	Mean Accuracy (%) for K = 1 NN's Using the Iris Dataset		
QRAM Size	**RZ**	**RY** $_{\text{PCA=1}}$	**RZ** $_{\text{PCA = 1}}$
8	89.1 ± 6.0	68.7 ± 10.93	69.1 ± 10.4
16	90.7 ± 6.04	71.8 ± 12.09	73.8 ± 5.69
32	93.8 ± 2.67	77.3 ± 4.09	77.8 ± 5.33
64	93.6 ± 2.44	79.8 ± 4.18	77.6 ± 3.42
128	92.9 ± 2.04	78.4 ± 3.6	79.3 ± 4.36

Table 3 presents the results of the MNIST dataset using the same process and methodology explained in Table 1. In this case, only the classical version and the RY and RZ rotation encodings are employed, as they operate with two features.

Table 3. Here, we show the results for experiment group one of the MNIST Dataset. The QRAM size controls the number of examples in the training set for both the classical and quantum algorithms, and sample draws were shared across algorithms for each experimental trial. For all experiments in group one we set $K = 1$. All values are in percent.

	Mean Accuracy (%) for K = 1 NN's Using the MNIST Dataset		
QRAM Size	**Classical**	**RY**	**RZ**
8	87.09 ± 14.11	79.6 ± 18.25	79.45 ± 18.84
16	96.73 ± 1.96	87.27 ± 8.36	90.18 ± 7.13
32	97.09 ± 2.18	93.82 ± 4.29	96.73 ± 1.53
64	95.82 ± 2.11	93.64 ± 3.45	95.27 ± 2.33
128	96.55 ± 2.00	92.91 ± 2.04	94.00 ± 2.11

Table 4. Here, we show the results for experiment group two for the Iris dataset. All values are in percent.

	Mean Accuracy (%) for the K = 1, 3, 5, 7, 9, 11-NN's Using the Iris Dataset					
K Value	**Classical**	**Classical** $_{\text{PCA=1}}$	**RY**	**RZ**	**RY** $_{\text{PCA=1}}$	**RZ** $_{\text{PCA=1}}$
1	96.4 ± 2.22	90.2 ± 2.22	94.0 ± 1.56	92.9 ± 2.04	78.4 ± 3.6	79.3 ± 4.36
3	97.3 ± 1.42	93.1 ± 2.49	94.2 ± 2.22	94.4 ± 2.44	83.3 ± 4.22	83.3 ± 4.44
5	97.3 ± 1.60	94.7 ± 1.96	94.0 ± 1.8	96.0 ± 2.22	84.4 ± 2.22	83.8 ± 2.89
7	97.8 ± 0.89	94.4 ± 2.22	95.7 ± 2.04	96.0 ± 1.77	84.9 ± 3.11	85.1 ± 3.64
9	98.0 ± 0.80	94.7 ± 2.40	95.7 ± 2.04	95.8 ± 1.24	89.0 ± 2.22	86.2 ± 2.4
11	97.3 ± 1.9	95.3 ± 2.09	95.6 ± 1.33	96.4 ± 1.51	84.0 ± 3.11	84.7 ± 2.93

4.1.2. Experiments Group 2

The second group used a fixed QRAM memory with 128 cells (occupied with 105 train events in the Iris dataset) and varied the K value for nearest-neighbor voting. The results, which use 1 to 11 nearest neighbors, are presented in Table 4 for the Iris data set and Table 5 for the MNIST dataset. Changing K does not change the number of qubits required for the quantum algorithms, instead, it changes the number of times the circuit is run. For $K > 1$, we run the circuit K times and vote over the returned values in a classical post-processing step. Note, that the oracle is not guaranteed to return a unique result on each call.

Table 5. Here, we show the results for experiment group two for the MNIST dataset. All values are in percent.

Mean Accuracy (%) for the K = 1, 3, 5, 7, 9, 11-NN's Using the MNIST Dataset			
K Value	**Classical**	**RY**	**RZ**
1	96.55 ± 2.00	92.91 ± 2.04	94.00 ± 2.11
3	98.00 ± 0.98	95.09 ± 2.73	94.18 ± 1.89
5	98.55 ± 0.87	95.27 ± 1.82	95.64 ± 1.45
7	98.36 ± 0.98	95.82 ± 1.38	96.36 ± 1.45
9	98.00 ± 0.98	96.18 ± 1.02	96.36 ± 1.09
11	97.82 ± 0.87	96.55 ± 0.98	96.36 ± 1.09

4.1.3. Experiments Limitation

The experiments conducted on our proposal are subject to Qiskit 1.0.x and Qiskit Aer version 0.14.0.1, which use GPU support—considering an NVIDIA GeForce RTX 4090 with an AMD Ryzen 9 7950X 16-core processor and 128 GB of RAM—to identify the times and capacities, a Grover iteration is made by varying the QRAM memory values ranging from 3 qubit addresses to 8, starting from 1 feature up to 9. Furthermore, in qiskit implementation, use the method transpile with the following characteristics optimization_level equal to 3, and the basis_gates=[“cx,” “rz,” “x,” “sx”], where we use the same basis gates that use

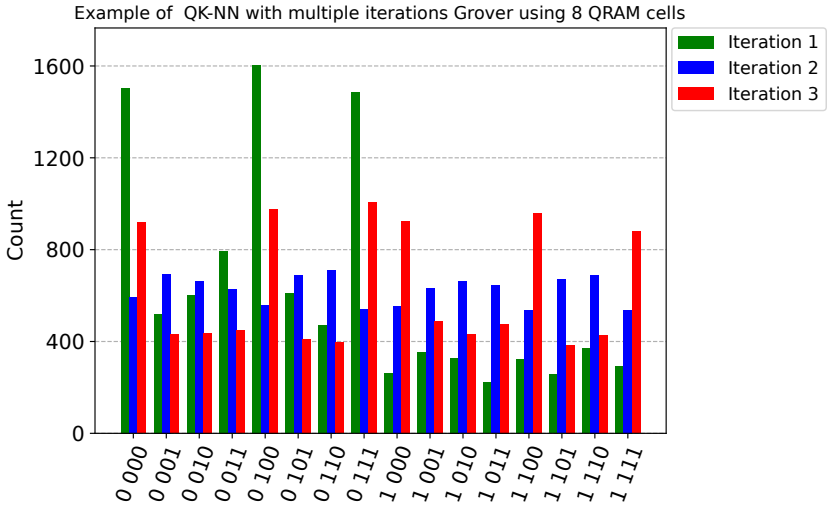
the IBM Quantum Hardware in its platform. Those characteristics consider the number of 2 qubit gates and the depth of the QK-NN circuits to identify the limitations in processing capabilities through an iteration of our proposed circuit with an example of the MNIST dataset, repeating each combination ten times as shown in Tables 6 and 7. Experimental evidence in Figures 13 and 14 demonstrates that as the number of cells increases, the number of iterations increases by one for the SWAP-Test with higher probability. In other words, our algorithm shows that the states increase their probability even with more memory cells by increasing the number of iterations of $\sqrt{m}$ addresses qubits.

Table 6. Here, we show part one of the results for experiments about the time considering different numbers of qubit addresses and features.

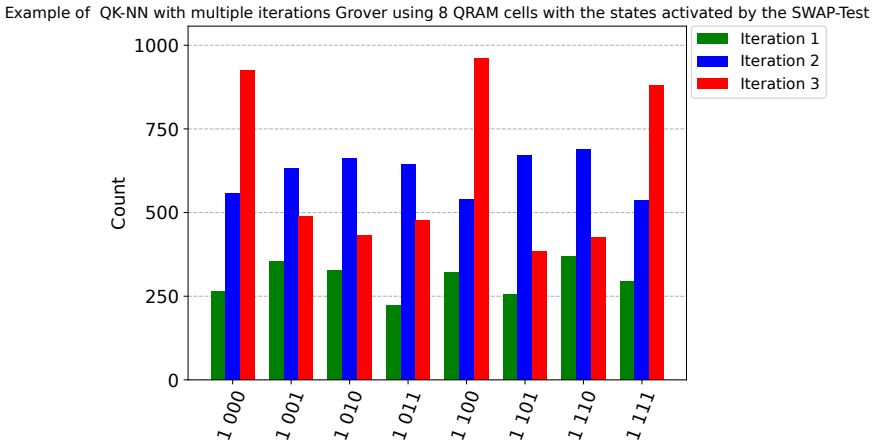
Mean Execution Time (s) for 3 to 6 Qubit Address, and 1 to 8 Features Samples.				
Number of Qubits / Number of Features	**3**	**4**	**5**	**6**
1	0.04 ± 0.04	0.06 ± 0.00	0.21 ± 0.00	0.90 ± 0.01
2	0.05 ± 0.07	0.07 ± 0.00	0.24 ± 0.01	0.87 ± 0.03
3	0.05 ± 0.06	0.07 ± 0.00	0.24 ± 0.00	0.87 ± 0.00
4	0.05 ± 0.06	0.08 ± 0.00	0.54 ± 0.00	1.97 ± 0.01
5	0.07 ± 0.06	0.17 ± 0.00	0.56 ± 0.00	2.07 ± 0.01
6	0.08 ± 0.06	0.18 ± 0.00	0.61 ± 0.00	2.57 ± 0.01
7	0.09 ± 0.06	0.20 ± 0.00	0.81 ± 0.00	4.53 ± 0.01
8	0.09 ± 0.02	0.28 ± 0.00	1.61 ± 0.01	12.44 ± 0.02

Table 7. Here, we show part two of the results for experiments about the time considering different numbers of qubit addresses and features.

Mean Execution Time (s) for 7 to 9 Qubit Address, and 1 to 8 Features Samples.			
Number of Qubits / Number of Features	**7**	**8**	**9**
1	3.70 ± 0.03	16.07 ± 2.15	64.82 ± 11.98
2	3.83 ± 0.03	15.15 ± 0.15	133.18 ± 9.65
3	8.04 ± 0.03	31.98 ± 0.10	129.41 ± 0.63
4	8.02 ± 0.03	33.52 ± 0.17	146.42 ± 0.70
5	9.60 ± 0.83	42.36 ± 0.22	226.21 ± 20.13
6	13.41 ± 0.06	83.98 ± 15.58	526.88 ± 12.55
7	30.81 ± 0.04	225.30 ± 3.87	-
8	12.44 ± 0.02	-	-

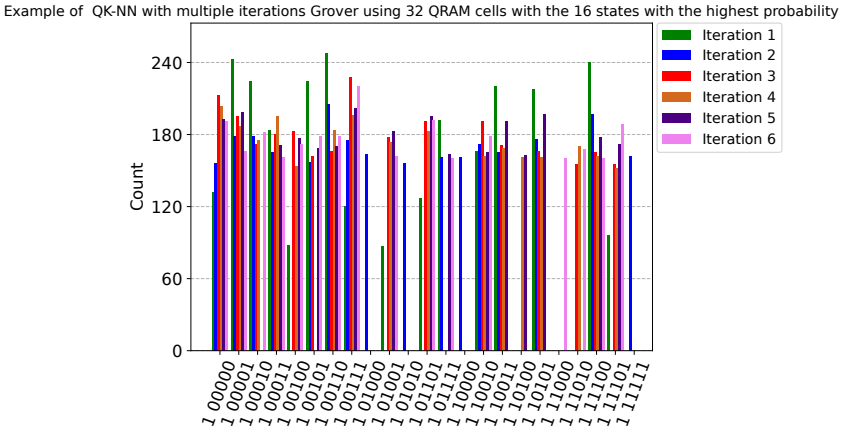


(**a**)



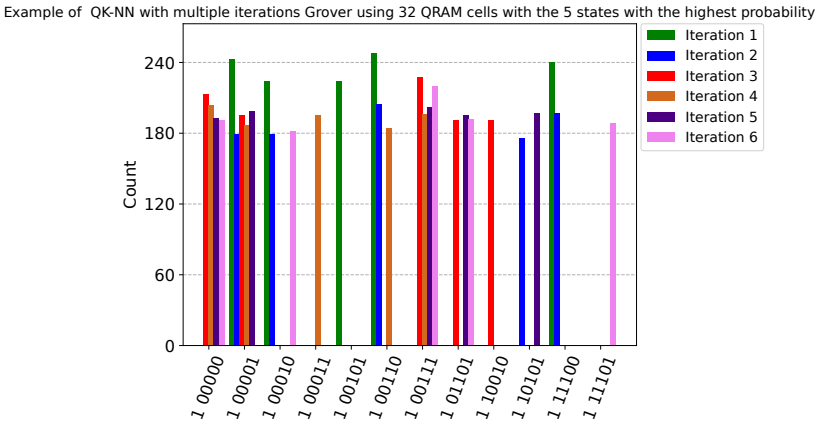
(**b**)

Figure 13. Probability histograms of QK-NN experiments with eight memory cells with Grover's iterations from 1 to 3. (**a**) Consider all the states. (**b**) Consider the probability of states when the SWAP-Test qubit equals one.



(**a**)

Figure 14. *Cont.*



(**b**)

Figure 14. Probability histograms of QK-NN experiments with eight memory cells with Grover's iterations from 1 to 6. (**a**) Consider all the higher 16 states when the SWAP-Test qubit equals one. (**b**) Consider the probability of the higher five states when the SWAP-Test qubit equals one.

5. Discussion

This paper introduces a QKNN algorithm that utilizes Grover's algorithm to accelerate event comparison. The dataset is encoded using a QRAM, and the Grover oracle is constructed via a SWAP-Test intermediate measurement. The algorithm evaluation is presented in Tables 1–5 where we find competitive performance across various QRAM (encoded dataset) sizes and encoding strategies.

The experiments show that the behavior of the proposed circuit with different preprocessing for encoding offers performance similar to the classical algorithm using the Euclidean distance implemented by the Scikit-learn library [35]. The quantum results are within statistical uncertainty of the classical results for any given experiment, but consistently slightly lower across experiments. Both quantum encoding approaches performed roughly equally well across experiments.

When considering different QRAM sizes, if the total encodable training set was small, all classifiers had poor accuracy. However, when training set sizes approached a quarter of the available training data, we found good performance, as displayed in Tables 1–3. When considering different numbers of neighbors, we find optimal performance by $K = 3$ or 5, with a precise determination obscured by statistical uncertainties, as presented in Tables 4 and 5. In general, we find performance increasing with the qubit counts, both as available to QRAM and as available for encoding features; the PCA-reduced as the Pooling experiments underperform the exact feature encoding.

It is not surprising to find approximate parity on accuracy given the simplicity of the dataset. It is difficult to measure any performance advantage at this stage in terms of wall-clock timing, but the dataset encoding has logarithmic scaling in the quantum case versus linear in the classical, and a polynomial advantage in best-match search time. The quantum algorithm performance likely has large constant factors to contend with, and we are not considering the cost of loading the dataset into QRAM which is daunting. However, it is reasonable to hypothesize the scaling performance advantages inherent in the technique could manifest some problems on future quantum computers.

Our results may be compared to two recent implementations of a QKNN using the Iris dataset. First, in [38], the algorithm uses a polar distance metric, and instead of a QRAM with Grover's algorithm, it employs amplitude estimation. It achieves an accuracy of around 95%, roughly independent of K for K between one and 13. The authors performed 30 ten-fold cross-validation experiments for a training set size of 135 (re-drawn 10 times). The second approach, detailed in [29], is based on the Hamming distance and a sorting algorithm. It achieves an accuracy of 94.66% for $K = 100$. The authors used a "leave

one out" training strategy, creating an effective training set size of 149, and repeated their experiments 50 times. The accuracy comparison between our algorithm and these recent state-of-the-art results is summarized in Table 8.

Table 8. Here, we present an accuracy comparison across the different methods discussed in this paper with Iris dataset. The QK-NN polar distance result is from [38] and the QK-NN sorting result is from [29]. The training data prescription is different in those works, making an exact comparison into a subtle task. They also do not present uncertainties. All values are in percent.

Metric	Classical K-NN	QK-NN Polar Distance	QK-NN Sorting	QK-NN QRAM Size 128 with RY (This Proposal)	QK-NN QRAM Size 128 with RZ (This Proposal)
Accuracy	96.4 ± 2.22	95.82	94.66	94.0 ±1.56	92.9 ±2.04

Our work presents a time complexity determined by the number of iterations in the Grover algorithm and the reading of probability states subject to the Grover algorithm, giving us a complexity of $O(\sqrt{m})$. In the state of the art, it is observed that method [38] has a complexity similar to ours with $O\left(\sqrt{mk}\right)$, implying a similar process of operations in terms of efficiency. However, our work highlights the ability to achieve satisfactory results with just one iteration, as evidenced in Figures 13 and 14. Additionally, our approach is based on angle encoding using the QRAM, unlike amplitude encoding using heaps, which is used in other methods. Consequently, the complexity of our work is determined by the number of gates to be encoded, expressed as $2 * log_2 n + log_2 M + t$, where n represents the number of qubits to be encoded, M denotes the number of parameters, and t is the value to be compared, and ours is $3 + 2 * n + m$. Also, [29] $O(knmp)$ is more complex when working with a four-variable system, where k is the number of the classes, n and m are the requirements to design the operator and sorting takes p time.

In the MNIST dataset, [41] presents a study that leverages the concept of QRAM memory, assuming the provision of oracles, to minimize the number of oracle queries required. They propose that they can achieve the accuracy level α using neural networks (NN) with a scaling number of oracle queries, proportionate to $O(\sqrt{M}) = O\left((1-\alpha)^{-5/4} \log\left((1-\alpha)^{-1}\right)\right)$ for constant success probability. In contrast, the classical nearest-neighbor algorithm scales queries as $O\left((1-\alpha)^{-5/2}\right)$. In Table 9, we experimentally demonstrate the type of experiment where we achieve the classification of this dataset and compare it with the classical part, obtaining only a 2.55% difference in RZ encoding.

Figures 15 and 16 demonstrate that our circuit proposal maintains a consistent ratio between the depth and the number of gates for two qubits. As we scale up the QK-NN to 128, it becomes inefficient, experiencing an exponential increase, supported by Table 7, where even with a size of 2^8 or 256 cells, having more than six features becomes impossible. Therefore, dealing with a large dataset in the Qiskit 1.0.x version with GPU could be more efficient. However, we conducted experiments with the MNIST dataset that support and enrich the results of [40], which consider oracles and QRAM as a given.

We can define to what extent it is viable to perform a QK-NN algorithm through simulation. Consider it a future work to address this process in real hardware, such as using IBM Quantum processors. These processors would work with the same number of gates if there is interaction between the qubits. Otherwise, the experiments will use extra ancillary qubits and additional CX gates. We also consider it essential to implement these results, taking into account the use of quantum hardware in robust projects, as in [48–50].

Table 9. Here, we present an accuracy comparison across the different methods discussed in this paper with the MNIST dataset. All values are in percent.

Metric	Classical K-NN	QK-NN QRAM Size 128 with RY (This Proposal)	QK-NN QRAM Size 128 with RZ (This Proposal)
Accuracy	96.55 ± 2.00	92.91 ± 2.04	94.00 ± 2.11

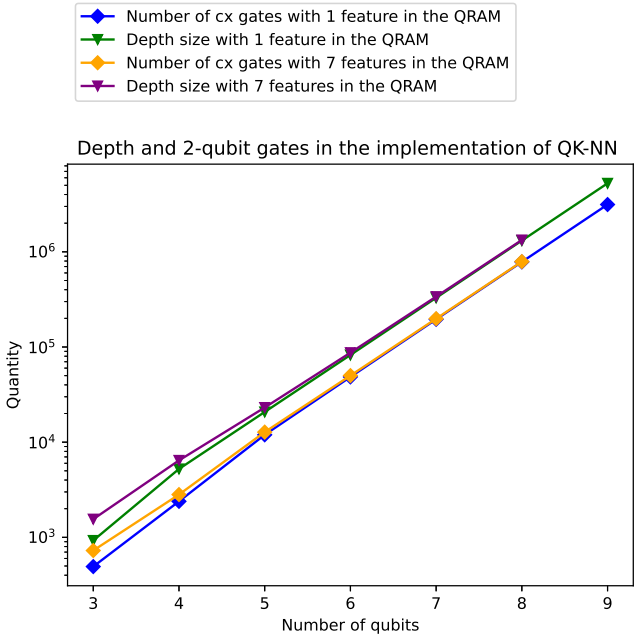


Figure 15. Depth and number of 2-qubit gates in different features and qubit address implementation of the QK-NN.

Considering the work's open-source nature, the code provided by this study can be incorporated into the noisy simulation module using the Qiskit 1.0.X format to explore the method's application on real quantum hardware. It supports CPU and GPU, utilizing CUDA 12.4, although this is not the study's primary purpose.

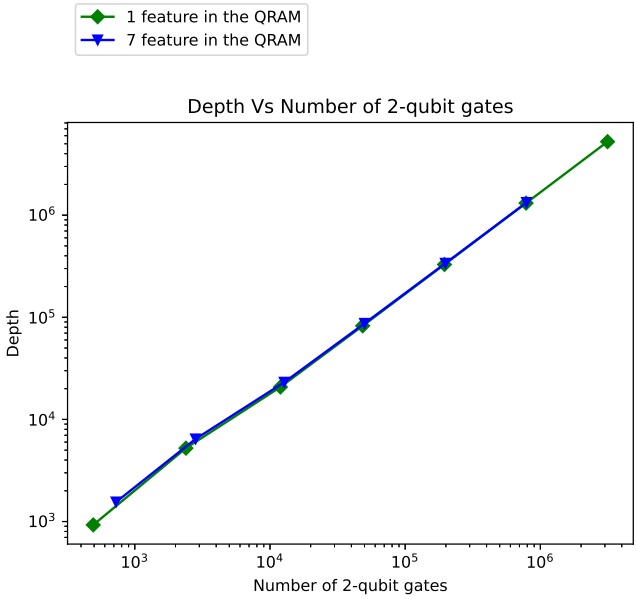


Figure 16. Depth VS number of 2-qubit gates

6. Conclusions

The notable findings of this study in the Iris dataset include an accuracy of approximately $94 \pm 1.6\%$, placing the performance of the proposed algorithm on par with the classical K-NN version, which achieves an accuracy of approximately $96 \pm 2.2\%$ obtained using the scikit-learn implementation. Additionally, the accuracy values are roughly equivalent to the performance of recent quantum implementations such as one approach using the polar distance metric with 95.82%, and another implementing a sorting algorithm with 94.66%.

Regarding the MNIST dataset, classes 0 and 1 demonstrate an accuracy of $94.00 \pm 2.11\%$, consistent with the classical result obtained using scikit-learn, registering $96.55 \pm 2.00\%$. This outcome underscores critical advancements in the state-of-the-art, presenting an experimental milestone for this dataset. It leverages a QRAM memory and an oracle integrated within a framework grounded in the quantum computing paradigm, such as Qiskit.

Our work builds on the theoretical foundations presented in [14,16,17], and provides a versatile framework capable of accommodating different training dataset sizes and different encoding schemes. Furthermore, in frameworks such as Qiskit, using intermediate measurements similar to dynamic circuits [51] while leaving other qubits unmeasured, enables circuit processing and the application of Grover's algorithm to identify states with higher probabilities than those with identical fidelity.

Our method offers potential extensions by integrating the Hamming distance with the fidelity metric. This would allow for the analysis of datasets with different data types. Leveraging the ability of QRAM to accommodate different cell types, such as those represented by RY and RZ, which interpret binary values as $|0\rangle$ and $|1\rangle$, opens avenues for developing two different oracles. For example, one oracle could be based on the Hamming distance, while the other could be based solely on the SWAP-Test, assigning specific qubits from the training set and test set features to each distance metric.

The main limitations in the application of our approach on modern hardware are to be found in available qubit count and noise levels. Here, we have considered noiseless circuit simulation as a proxy for an anticipated blend of quantum error mitigation and error correction on a small number of qubits. Scaling the exact classical simulation of this algorithm to high qubit counts and circuit depths faces severe constraints. It is an open question as to whether approximate circuit methods, such as tensor networks, could produce equivalent results.

The natural next step is to consider more complicated problems with larger datasets and more complex feature sets. We are releasing code that may be easily adapted to such studies. It will be important to eventually understand if quantum machine learning offers a provable performance advantage. This could be demonstrated in the future by demonstrating strong scaling performance in qubit count for complex problems, or by measuring significant complexity or wall-clock savings. The latter demonstration must likely wait for better hardware performance and at least partial quantum error correction.

Author Contributions: Conceptualization, A.M.-R.; methodology, A.M.-R. and J.Y.M.-P.; software, A.M.-R. and V.O.; validation, A.M.-R.; formal analysis, A.M.-R. and V.O. ; resources, J.Y.M.-P.; writing—original draft preparation, A.M.-R.; writing—review and editing, J.Y.M.-P., V.O., J.M.-R. and J.H.S.-A.; visualization, A.M.-R. and V.O.; supervision, J.Y.M.-P.; funding acquisition, J.H.S.-A. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by Instituto Politécnico Nacional (IPN) and Consejo Nacional de Humanidades Ciencia y Tecnologías (CONAHCYT).

Data Availability Statement: This study on performance testing can be fully replicated, including the data, by executing the code in the repository https://github.com/MaldoAlberto/QKnn (accessed on 21 April 2024). The Iris dataset is distributed with Scikit-learn [35].

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ML	Machine Learning
QML	Quantum Machine Learning
QRAM	Quantum Random Access Memory
K-NN	K Nearest Neighbor
QK-NN	Quantum K Nearest Neighbor
PCA	Principal Component Analysis
MCX	Multi Controlled X

References

1. Mendonça, M.O.; Netto, S.L.; Diniz, P.S.; Theodoridis, S. Chapter 13—Machine learning: Review and trends. In *Signal Processing and Machine Learning Theory*; Diniz, P.S.R., Ed.; Academic Press: New York, NY, USA, 2024; pp. 869–959. ISBN 9780323917728. [CrossRef]
2. LeCun, Y.; Bengio, Y.; Hinton, G. Deep Learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
3. Voulodimos, A.; Doulamis, N.; Doulamis, A.; Protopapadakis, E. Deep Learning for Computer Vision: A Brief Review. *Comput. Intell. Neurosci.* **2018**, *2018*, 7068349. [CrossRef] [PubMed]
4. Omar, A.; Abd El-Hafeez, T. Quantum computing and machine learning for Arabic language sentiment classification in social media. *Sci. Rep.* **2023**, *13*, 17305. [CrossRef] [PubMed]
5. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to Sequence Learning with Neural Networks. *Adv. Neural Inf. Process. Syst.* **2014**, *4*.
6. Flurin, E.; Martin, L.S.; Hacohen-Gourgy, S.; Siddiqi, I. Using a Recurrent Neural Network to Reconstruct Quantum Dynamics of a Superconducting Qubit from Physical Observations. *Phys. Rev. X* **2020**, *10*, 011006. [CrossRef]
7. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Networks. In *Advances in Neural Information Processing Systems 27*; Ghahramani, Z., Ed.; Curran Associates, Inc.: New York, NY, USA, 2014.
8. Maldonado-Romo, J.; Maldonado-Romo, A.; Aldape-Pérez, M. Path Generator with Unpaired Samples Employing Generative Adversarial Networks. *Sensors* **2022**, *22*, 9411. [CrossRef]
9. Litjens, G.; Kooi, T.; Bejnordi, B.E.; Setio, A.A.; Ciompi, F.; Ghafoorian, M.; Van Der Laak, J.A.; Van Ginneken, B.; Sánchez, C.I. A Survey on Deep Learning in Medical Image Analysis. *Med. Image Anal.* **2017**, *42*, 60–88. [CrossRef] [PubMed]
10. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. *Neural Inf. Process. Syst.* **2012**, *25*. [CrossRef]
11. Hewamalage, H.; Bergmeir, C.; Bandara, K. Recurrent Neural Networks for Time Series Forecasting: Current Status and Future Directions. *Int. J. Forecast.* **2019**, *37*, 388–427. [CrossRef]
12. Ray, P.P. ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. *Internet Things -Cyber-Phys. Syst.* **2023**, *3*, 121–154. [CrossRef]
13. Schuld, M.; Sinayskiy, I.; Petruccione, F. An introduction to quantum machine learning. *Contemp. Phys.* **2014**, *56*, 172–185. [CrossRef]
14. Wittek, P. *Quantum Machine Learning: What Quantum Computing Means to Data Mining*; Academic Press: San Diego, CA, USA, 2014.
15. Lloyd S, Mohseni M, Rebentrost P. Quantum principal component analysis. *Nat. Phys.* **2013**, *10*, 631–633. [CrossRef]
16. Ruan, Y.; Chen, H.W.; Liu, Z.H.; Zhang, J.; Zhu, W. Quantum principal component analysis algorithm. *Jisuanji Xuebao/Chin. J. Comput.* **2014**, *37*, 666–676. [CrossRef]
17. Heredge, J.; Hill, C.; Hollenberg, L.; Sevior, M. Quantum Support Vector Machines for Continuum Suppression in B Meson Decays. *Comput. Softw. Big Sci.* **2021**, *5*, 27. [CrossRef]
18. Ruan, Y.; Xue, X.; Liu, H.; Tan, J.; Li, X. Quantum Algorithm for K-Nearest Neighbors Classification Based on the Metric of Hamming Distance. *Int. J. Theor. Phys.* **2017**, *56*, 3496–3507. [CrossRef]
19. Wiebe, N.; Kapoor, A.; Svore, K. Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning. *Quantum Inf. Comput.* **2015**, *15*, 318–358. [CrossRef]
20. Gupta, S.; Zia, R.K. Quantum Neural Networks. *J. Comput. Syst. Sci.* **2002**, *63*, 355–383. [CrossRef]
21. Bonnell, G.; Papini, G. Quantum neural network. *Int. J. Theor. Phys.* **1997**, *36*, 2855–2875. [CrossRef]
22. Nasteski, V. An overview of the supervised machine learning methods. *Horizons B* **2017**, *4*, 51–62. [CrossRef]
23. Liu, Q.; Wu, Y. Supervised Learning. In *Encyclopedia of the Sciences of Learning*; Springer: Boston, MA, USA, 2012._451. [CrossRef]
24. Zhang, S.; Li, X.; Zong, M.; Zhu, X.; Cheng, D. Learning k for kNN Classification. *ACM Trans. Intell. Syst. Technol.* **2017**, *8*, 1–19. [CrossRef]
25. Taunk, K.; De, S.; Verma, S.; Swetapadma, A. A Brief Review of Nearest Neighbor Algorithm for Learning and Classification. In Proceedings of the 2019 International Conference on Intelligent Computing and Control Systems (ICCS), Madurai, India, 15–17 May 2019; pp. 1255–1260. [CrossRef]
26. Sawerwain, M.; Wróblewski, M. Recommendation systems with the quantum k-NN and Grover algorithms for data processing. *Int. J. Appl. Math. Comput. Sci.* **2019**, *29*, 139–150. [CrossRef]

27. Giovannetti, V.; Lloyd, S.; Maccone, L. Quantum Random Access Memory. *Phys. Rev. Lett.* **2008**, *100*, 160501. [CrossRef] [PubMed]
28. Hong, F.Y.; Xiang, Y.; Zhu, Z.Y.; Jiang, L.Z.; Wu, L.N. A Robust Quantum Random Access Memory. *Phys. Rev. A* **2012**, *86*, 010306. [CrossRef]
29. Quezada, L.F.; Sun, G.H.; Dong, S.H. Quantum Version of the k-NN Classifier Based on a Quantum Sorting Algorithm. *Ann. Der Phys.* **2022**, *534*, 2100449. [CrossRef]
30. Sakhi, Z.; Tragha, A.; Kabil, R.; Bennai, M. Grover Algorithm Applied to Four Qubits System. *Comput. Inf. Sci.* **2011**, *4*, 125–130. [CrossRef]
31. Nielsen, M.A.; Chuang, I.L. *Quantum Computation and Quantum Information: 10th Anniversary Edition*; Cambridge University Press: Cambridge, UK, 2011; ISBN 9781107002173.
32. Mandviwalla, A.; Ohshiro, K.; Ji, B. Implementing Grover's Algorithm on the IBM Quantum Computers. In Proceedings of the 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 10–13 December 2018; pp. 2531–2537. [CrossRef]
33. Garcia-Escartin, J.C.; Chamorro-Posada, P. SWAP test and Hong-Ou-Mandel effect are equivalent. *Phys. Rev. A* **2013**, *87*, 052330. [CrossRef]
34. Seetharaman, R. TEQIP -III Sponsored First International Conference on Innovations and Challenges in Computing, Analytics and Security. 2020. Available online: https://books.aijr.org/index.php/press/catalog/book/90 (accessed on 21 April 2024)
35. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
36. LeCun, Y.; Cortes, C.; Burges, C.J.C. *The MNIST Database of Handwritten Digits*; MNIST: New York, USA, 1998. Available online: http://yann.lecun.com/exdb/mnist/ (accessed on 29 April 2024).
37. Gao, L.Z.; Lu, C.Y.; Guo, G.D.; Zhang, X.; Lin, S. Quantum K-nearest neighbors classification algorithm based on Mahalanobis distance. *Front. Phys.* **2022**, *10*, 1047466. [CrossRef]
38. Feng, C.; Zhao, B.; Zhou, X.; Ding, X.; Shan, Z. An Enhanced Quantum K-Nearest Neighbor Classification Algorithm Based on Polar Distance. *Entropy* **2023**, *25*, 127. [CrossRef]
39. Li, J.; Zhang, J.; Zhang, J.; Zhang, S.Quantum KNN Classification With K Value Selection and Neighbor Selection. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2023**, *43*, 1332–1345. [CrossRef]
40. Fastovets, D.V.; Bogdanov, Y.I.; Bantysh, B.I.; Lukichev, V.F. Machine learning methods in quantum computing theory. *Int. Conf. Micro Nano-Electron.* **2019**, *11022*, 752–761.
41. Wiebe, N.; Kapoor, A.; Svore, K.M. Quantum Nearest-Neighbor Algorithms for Machine Learning. Rinton Press. *Quantum Inf. Comput.* **2015**, *15*, 318–358. Available online: https://www.microsoft.com/en-us/research/publication/quantum-nearest-neighbor-algorithms-for-machine-learning/ (accessed on 19 April 2024).
42. Schuld, M.; Petruccione, F. *Supervised Learning with Quantum Computers*; Springer: Berlin, Germany, 2018. [CrossRef]
43. LaRose, R.; Coyle, B. Robust data encodings for quantum classifiers. *Phys. Rev. A* **2020**, *102*, 032420. [CrossRef]
44. Greenacre, M.; Groenen, P.J.; Hastie, T.; d'Enza, A.I.; Markos, A.; Tuzhilina, E. Principal component analysis. *Nat. Rev. Methods Prim.* **2022**, *2*, 100. [CrossRef]
45. Grover, L.K. A Fast Quantum Mechanical Algorithm for Database Search. In Proceedings of the Annual ACM Symposium on Theory of Computing, Philadelphia, PA, USA, 22–24 May 1996; pp. 212–219.
46. Qiskit, Qiskit Contributors. Qiskit: An Open-source Framework for Quantum Computing. 2023. Available online: https://zenodo.org/records/8190968 (accessed on 19 April 2024).
47. Fisher, R.A. Iris. UCI Machine Learning Repository. 1988. Available online: https://archive.ics.uci.edu/dataset/53/iris (accessed on 19 April 2024).
48. Trochatos, T.; Xu, C.; Deshpande, S.; Lu, Y.; Ding, Y.; Szefer, J. A Quantum Computer Trusted Execution Environment. *IEEE Comput. Archit. Lett.* **2023**, *22*, 177–180. [CrossRef]
49. Itoko, T.; Raymond, R.; Imamichi, T.; Matsuo, A.; Cross, A.W. Quantum circuit compilers using gate commutation rules. In Proceedings of the 24th Asia and South Pacific Design Automation, Tokyo, Japan, 21–24 January 2019; pp. 191–196. [CrossRef]
50. Scholten, T.L.; Perry, D., II; Washington, J.; Glick, J.R.; Ward, T. A Model for Circuit Execution Runtime And Its Implications for Quantum Kernels At Practical Data Set Sizes. *arXiv* **2023**, arXiv:2307.04980.
51. Fang, K.; Zhang, M.; Shi, R.; Li, Y. Dynamic quantum circuit compilation. *arXiv* **2023**, arXiv:2310.11021.