Quantum Eigenfaces: Linear Feature Mapping and Nearest Neighbor Classification with Outlier Detection

Armando Bellante[†], William Bonvini[*], Stefano Vanerio[†], Stefano Zanero[†]
[†]Politecnico di Milano, DEIB, Via Ponzio 34/5 Building 20, 20133 Milan, Italy
[*]Independent researcher, Milan, Italy
armando.bellante@polimi.it, william.bonvini@outlook.com, stefano.vanerio@mail.polimi.it, stefano.zanero@polimi.it

Abstract—**We propose a quantum machine learning algorithm for data classification, inspired by the seminal computer vision approach of eigenfaces for face recognition. The algorithm enhances nearest neighbor/centroid classifiers with concepts from principal component analysis, enabling the automatic detection of outliers and finding use in anomaly detection domains beyond face recognition. Assuming classical input data, we formalize how to implement the algorithm using a quantum random access memory and state-of-the-art quantum linear algebra, discussing the complexity of performing the classification algorithm on a fault-tolerant quantum device. The asymptotic time complexity analysis shows that the quantum classification algorithm can be more efficient than its classical counterpart. We showcase an application of this algorithm for face recognition and image classification datasets with anomalies, obtaining promising results for the running time parameters. This work contributes to the growing field of quantum machine learning applications, and the algorithm's simplicity makes it easily adoptable by future quantum machine learning practitioners.**

Index Terms—**quantum computing, eigenfaces, linear feature mapping, principal component analysis, nearest centroid classification, nearest neighbor classification, PCA, machine learning, face recognition, anomaly detection, outlier detection**

I. Introduction

Quantum machine learning (QML) has recently emerged as a promising field, combining the principles of machine learning with the power of quantum computing. Machine learning algorithms learn from examples, and as such, they require a large number of examples to make accurate predictions. Training machine learning algorithms on larger datasets can help reduce the impact of noise and outliers, and prevent overfitting, potentially leading to improved accuracy and robustness. However, increasing the number of training points and features often impacts the classification methods' running time, making the algorithms inefficient on large-scale datasets. Quantum learning algorithms can promise asymptotic time improvements in these parameters over their classical counterparts, enabling faster training and/or classification on datasets with a high number of data points and features. Some famous examples include quantum algorithms for support vector machines [22], nearest neighbor classification [30], and k-means [17].

In this paper, inspired by the seminal work of Turk and Pentland [26, 27], we study a quantum classification algorithm for face recognition with eigenfaces. Trained on a set of faces, the eigenfaces algorithm can classify a new face as either a known face, belonging to a training subject, as a potential face of an unknown subject, or as an outlier image that is not a face. The main idea behind the algorithm is to represent each new face as a linear combination of some basis vectors (known as eigenfaces) and then compare the coefficients of their eigenface representations with the coefficients of the training subjects. Usually, the basis vectors are the training set's principal components, and the comparison method is a nearest neighbor/centroid classifier tuned with threshold parameters to detect the outliers. While modern deep learning algorithms have superseded this classification algorithm in the face recognition task [29], this result served as a milestone in the classical machine learning literature and might find practical applications in the early fault-tolerant stage of the quantum computing era because of its simplicity. Furthermore, some ideas from this algorithm, like observing the amount of norm retained after the projection on the principal components, can be used to make other classifiers more resilient to outliers and/or to perform anomaly detection [28].

More generally, the eigenfaces approach is a linear feature mapping followed by a nearest neighbor/centroid classifier, enhanced with some outlier detection strategies. While quantum algorithms for nearest neighbor/centroid have been studied in the past [30, 20, 23, 14], our paper positions slightly differently. Most importantly, it is focused on the eigenfaces application, covers the specific task of outlier detection, and specifies the implementation of an end-to-end classification system with state-of-the-art routines, from classical data loading to linear feature mapping and classification. We describe the algorithm supposing the availability of a quantum random access memory, a classical computer, and a fault-tolerant quantum computer, describing the steps required for the classification and the asymptotic time complexity of the prediction stage. We strive to make the analysis as general as possible, without making any assumptions about the norms of the data points or the matrices involved in the computation. We corroborate our study with some numerical experiments that show some use cases of these algorithms and how to estimate the running time parameters that dominate the time complexity, which seem already advantageous on small datasets.

Regarding the previously cited related work, the most important comparisons are with Lloyd et al. [20] and Wiebe et al. [30]. On the one hand, the first one presents a nearest centroid algorithm based on inner product distance, assuming a quantum random access memory and mentioning the possibility of applying feature mappings. On the other hand, the second one uses Euclidean distance-based classifiers, which is more similar to our setting, but in a different input model (sparse access), which leads to a dependency on the features' sparsity. Their classifier is based on Euclidean distance estimation, followed by a search for the minimum [9], an approach that found applications in many other algorithms [2, 7, 17], including this work. In particular, this work combines the input model of the first paper (enhanced with up-to-date techniques) with the processing routines of the second one, adding the linear feature mapping and discussing the outlier detection techniques from the eigenfaces approach.

The outline of the manuscript is the following. Section III introduces the classical algorithm for face recognition with eigenfaces and explains the generalization to algorithms involving linear feature mappings and nearest neighbor/centroid classification. Section IV presents quantum input and output primitives and the state-of-the-art linear algebra results that set the ground for our classification procedure. In Section V, we show how to combine these routines to derive the quantum algorithms for the task. Section VI presents numerical experiments for face recognition and image classification with outliers. Finally, Section VII briefly discusses our results.

II. Notation

The notation $[n]$, with $n \in \mathbb{N}$, denotes the set $\{1, \ldots, n\}$. We use capital letters to denote matrices and lowercase letters to denote scalars. Vectors are denoted with an arrow, as in $\vec{x} \in \mathbb{R}^n$. Its entries are x_i for $i \in [n]$. The ℓ_∞ norm of a vector is $\|\vec{x}\|_\infty = \max_{i\in[n]} |x_i|$, while the ℓ_p norm is $\|\vec{x}\|_p = (\sum_{i=1}^n |x_i|^p)^{1/p}$, for $p \in \mathbb{N}$. When $p = 0$, both $\|\vec{x}\|_p$ and $\|\vec{x}\|_p^p$ denote the number of non-zero entries of $\vec{x}$. If not specified otherwise, $\|\vec{x}\|$ is the ℓ_2 norm. We denote the Euclidean distance between two vectors $\vec{x}$ and $\vec{y}$ as $d(\vec{x}, \vec{y}) = \|\vec{x} - \vec{y}\|_2$ and their inner product with $(\vec{x}, \vec{y}) = \vec{x}^T\vec{y}$.

Let $A \in \mathbb{R}^{n\times m}$ be a matrix. We denote its rows as $\vec{a}_i$, for $i \in [n]$, and columns as $\vec{a}_{\cdot,j}$, for $j \in [m]$. Its (i, j) entry is $A_{i,j}$. Its Frobenius norm is defined as $\|A\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^m A_{i,j}^2}$, while its spectral norm (or operator norm) is $\|A\| = \sup_{x\in R^m} \frac{\|A\vec{x}\|}{\|\vec{x}\|}$.

We use the big-O notation for computational complexities. In this work, the notation $\widetilde{O}(\cdot)$ hides polylogarithmic factors in the input size parameters, precision parameters and other parameters appearing inside the brackets.

III. Classical algorithm

Here, we describe the eigenfaces representation and the classification algorithm [26, 27].

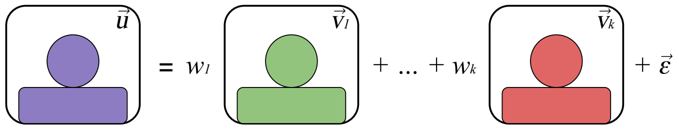


Fig. 1: A face can be expressed as a linear combination of eigenfaces plus an error vector (Eq. 1).

A. The Eigenfaces representation

Images can be thought of as matrices of $w \times h$ pixels or equivalently as vectors of $m = wh$ entries. While the image's resolution might create high dimensional data points (a 128×128 image would generate a vector of size 16384), it is unlikely that images of faces are uniformly distributed over this ample space. The key idea of the eigenfaces classification algorithm is that faces can be expressed as linear combinations of other faces [25]. Via principal component analysis, one can extract the basic faces (eigenfaces) that enable spanning the space where the face images live. Images can then be represented using the linear combination coefficients that enable the best reconstruction. Let us formalize this better.

Let $X \in \mathbb{R}^{n\times m}$ be the matrix containing the faces. Each row $\vec{x}_i \in \mathbb{R}^m$ of X is one face, expressed using m pixels. For simplicity, images are gray-scale, with pixel values in the usual range of $0 - 255$. The first step of the representation consists of computing the "average face" of the dataset $\overline{x} = \frac{1}{n}\sum_{i=1}^n \vec{x}_i$ subtracting it from all the faces to center the data points. This way, we compute a new matrix $U \in \mathbb{R}^{n\times m}$, where each face is $\vec{u}_i = \vec{x}_i - \overline{x}$.

The *eigenfaces* are defined as the eigenvectors of the covariance matrix of U^T, corresponding to the k-greatest eigenvalues. The number k of eigenfaces to retain can be either known from heuristics or set by deciding how much variance the eigenvectors should explain. In practice, one can compute the eigendecomposition of the covariance matrix $U^TU = V\Lambda V^T$ or equivalently perform the singular value decomposition of U and compute the right singular vectors corresponding to the top-k singular values. These vectors $v_1, \ldots, v_k$ are such that $v_i \in \mathbb{R}^m$, so they are of the same dimension as the original images. The principal eigenvectors exhibit facial features when represented as images, hence the name eigenfaces. With a slight abuse of notation, we denote the matrix containing the top-k eigenvectors with V.

Once the eigenfaces are computed, each image can be expressed in terms of $\{v_1, \cdots, v_k\}$. Each image (potentially not present in the original dataset) can then be written as $\vec{x} = \overline{x} + \vec{u}$, where

$$\vec{u} = \sum_{i=1}^{k} w_k\vec{v}_k + \vec{\epsilon} = V^T\vec{w} + \vec{\epsilon}, \tag{1}$$

and $\vec{\epsilon}$ is an error term entirely supported in the null space of V. Figure 1 provides a visual representation of Eq. 1. Images are expressed as the "average face" plus a weighted sum of the eigenfaces and an error that that the eigenfaces cannot capture. In this model, the features are the linear combination

Algorithm 1 Eigenfaces-based classification

Input: $V \in \mathbb{R}^{k\times m}$ matrix of eigenfaces, $\overline{x} \in \mathbb{R}^m$ mean vector of the dataset, $\vec{x} \in \mathbb{R}^m$ sample to classify, $\{\vec{c}_1, \cdots, \vec{c}_p\}$ set of stored weights vectors with $\vec{c}_j \in \mathbb{R}^k$, $\vec{y} \in [P]^p$ label vector, thresholds $\delta_1, \delta_2 \in \mathbb{R}_{>0}$.
Output: an integer indicating the classification outcome.

```
1: Compute the weights vector of $x$:
       $\vec{w} = V(\vec{x} - \overline{x})$
2: Select the minimum distance between $w$ and the stored weights:
       $d = \min_{j\in[p]} \|\vec{w} - \vec{c}_j\|_2^2$
3: Save the index of the closest weights vector:
       $j^* = \arg\min_{j\in[p]} \|\vec{w} - \vec{c}_j\|_2^2$
4: Output: if $d \leq \delta_1$: same class of $\vec{c}_{j^*}$, output $y_{j^*}$;
           if $\delta_1 < d < \delta_2$: similar element, output $-1$.
           if $d \geq \delta_2$: not a similar element, output $-2$.
```

coefficients $\{w_i\}_{i=1}^k$. We can compute the feature vector by reversing Eq. 1:

$$V^T\vec{w} = \vec{u} - \vec{\epsilon} \tag{2}$$

$$VV^T\vec{w} = V(\vec{u} - \vec{\epsilon}) \tag{3}$$

$$\vec{w} = V\vec{u} = V(\vec{x} - \overline{x}) \tag{4}$$

since we have $VV^T = I \in \mathbb{R}^{k\times k}$, $V\vec{\epsilon} = \vec{0}$, and $\vec{u} = \vec{x} - \overline{x}$.

B. Classification and outlier detection

The eigenfaces classification algorithm takes an image $\vec{x} \in \mathbb{R}^m$ and outputs either a natural number denoting the subject the face belongs to, -1 if the image is an unknown face, or -2 if the image is not a face. The training step requires a training dataset $X \in \mathbb{R}^{n\times m}$. Each training image $\vec{x}_i$ is associated with a corresponding subject id $y_i \in \mathbb{N}$. For simplicity assume that the ids cover a contiguous sequence, from 0 to $P < n$.

The main idea is to compute the eigenfaces representation of the image, as discussed in the previous section, and compare it to the ones of the training set using a nearest neighbor or centroid approach. In the nearest neighbor approach, *each* data point of the training set is mapped in the new feature space as $\vec{c}_i = V(\vec{x}_i - \overline{x})$ (Eq. 4), for $i \in [n]$. In the nearest centroid approach, instead, one computes the "average face" of each subject as $\hat{x}_i = \frac{1}{N_i}\sum_{j:y_j=i}\vec{x}_j$ and computes the comparison data points as $\vec{c}_i = V(\hat{x}_i - \overline{x})$, for $i \in [P]$. In any case, from the training set, one computes a set of vectors $\{\vec{c}_1, \cdots, \vec{c}_p\}$, with $p = n$ or $p = P$, and their corresponding labels $\vec{y} \in [P]^p$.

Once the comparison vectors are stored, one can proceed with the classification algorithm. The new image $\vec{x}$ gets mapped in the new feature space as $\vec{w} = V(\vec{x} - \overline{x})$. Then, we compute the Euclidean distance between $\vec{w}$ and the vectors $\vec{c}_j$, for all $j \in [p]$, to select the minimum distance $d = min_{j\in[p]}\|\vec{w} - \vec{c}_j\|_2^2$ and the index $j^* = \arg\min_{j\in[p]}\|\vec{w} - \vec{c}_j\|_2^2$. If the distance d is lower than a threshold $\delta_1 \in \mathbb{R}_{>0}$, the image is classified as the face of the j^* subject. Instead, if the distance is greater than δ_1, but less than $\delta_2 \in \mathbb{R}_{>0}$, the sample is considered an unknown face. Finally, if the distance exceeds δ_2, the image is not considered a face.

The classification procedure is summarized in Algorithm 1. The textbook implementation of this routine leads to a running time of $O(mk + pk)$, where mk is given by matrix multiplication and pk by the search.

1) Norm-based outlier detection: It is possible to introduce an extra check before step 2 to enhance the outlier recognition accuracy with a simple observation. While faces are likely to be supported on the eigenfaces space, different images are not. One can compute how much norm is preserved in the eigenfaces space $\frac{\|V(\vec{x}-\overline{x})\|}{\|(\vec{x}-\overline{x})\|}$ and compare it with a threshold γ: if the percentage is smaller than γ the algorithm outputs -2 without proceeding further.

2) Hyperparameters tuning: The parameters $k, \delta_1, \delta_2, \gamma$ can be learned via numerical optimization on a validation set containing subjects not present in the training set (to tune δ_1) and images that do not represent faces (to tune δ_2). The eigenfaces and the mean vector $\overline{x}$ are computed using exclusively the training set.

3) Generalization remark: Alg. 1 can be seen as a linear feature mapping followed by nearest neighbor/centroid classification. A feature map is a function $\Phi : \mathbb{R}^m \mapsto \mathbb{R}^k$, applied to the features of the data point. The Eigenfaces feature mapping corresponds to $\Phi(\vec{x}) = V\vec{x}$. When discussing our quantum algorithm, we will discuss the complexity of performing the classification after a general linear mapping $\Phi(\vec{x}) = A\vec{x}$ and then focus the analysis on the PCA case, where $A = V$.

IV. Quantum preliminaries

We describe the data access and processing primitives from previous literature that enable designing our quantum algorithm, presenting how to combine them. While we do not report the proof/implementation of every technique, we try to provide the reader with an understanding of the technique and a reference to the relevant work(s).

A. Data access

1) Quantum Random Access Memory: In order to efficiently process classical data, we assume the availability of a quantum random access memory. A quantum random access memory (QRAM) is a device that, analogously to a classical random access memory (RAM), enables efficient retrieval and storage of bitstrings. We can think of a QRAM as a device to store classical bitstrings and retrieve them on quantum registers. Both classical and quantum memories are composed of cells that store bitstrings and the queries are performed by specifying the address of the cell of interest. The main difference is that a QRAM can be queried in superposition. In particular, suppose that the QRAM has $N = 2^n$ cells, each capable of storing a bitstring of length p. Then, the QRAM can be thought as a unitary acting on $n + p$ qubits as

$$U_{\text{QRAM}} : |i\rangle|0\rangle \mapsto |i\rangle|x_i\rangle, \tag{5}$$

for $i \in [N]$ and x_i equal to the bitstring stored in the i^{th} cell.

An important requirement for quantum random access memories is time efficiency. While this device generally requires $\widetilde{O}(Np)$ logical qubits and gates, some architecture proposals suggest that this unitary can be highly parallelized and implemented in *depth* $O(\text{polylog}(N))$ [12]. For sure, the requirement for many gates and logical qubits makes building these devices a hard challenge today, and we do not know whether they will be ever realised in practice. However, recent research suggests that such architectures are highly resistant to generic errors and require only a small amount of error correction, which should not affect their performances [13]. These results preserve hope for the practicality and time efficiency of such devices.

In the next sections, as in many quantum machine learning papers, we will assume the availability of such devices, taking into account the cost of storing data, to perform data access in time $\widetilde{O}(1)$. We want to remark that even if QRAMs turn out impossible to realize, it is still possible to implement our algorithms using alternative data preparation circuits. In that case, the efficiency of the algorithms discussed in this work would be multiplied by the (potentially higher) time complexity of implementing the unitaries needed to provide the desired quantum access to matrices and vectors.

2) Preparing quantum access: Given the premises above, we can proceed by presenting the tools that enable preparing efficient quantum access to matrices and vectors.

Definition IV.1 (Efficient Quantum Access to a Matrix)**.** *We say to have efficient quantum access to a matrix $A \in \mathbb{R}^{n\times m}$ if we can perform the following mappings in time $O(\text{polylog}(nm))$:*

- $U : |i\rangle |0\rangle \to |i\rangle |\vec{a}_i\rangle = |i\rangle \frac{1}{\|\vec{a}_i\|} \sum_j^m A_{i,j} |j\rangle$, *for* $i \in [n]$;
- $V : |0\rangle \to \frac{1}{\|A\|_F} \sum_i^n \|\vec{a}_i\| |i\rangle$.

Preparing quantum access to a vector is a special case of preparing quantum access to a matrix. Indeed, one only needs the unitary $U : |0\rangle \to \frac{1}{\|\vec{x}\|} \sum_{i\in[n]} x_i |i\rangle$.

Kerenidis and Prakash [15] proved that there exists a data structure that, if stored in a QRAM, enables efficient quantum access as per Def IV.1.

Theorem IV.2 (Implementing quantum operators using an efficient data structure [15, Theorem 10])**.** *Let $A \in \mathbb{R}^{n\times m}$. There exists a data structure to store the matrix A with the following properties:*

1) *The size of the data structure is $O(nnz(A)\log^2(nm))$.*
2) *The time to store a new entry $(i, j, A_{i,j})$ is $O(\log(nm))$*[1].
3) *Provided coherent quantum access to this structure (as discussed in Sec. IV-A1) there exists quantum algorithms that implement U and V as per Def. IV.1 in time $O(\text{polylog}(nm))$.*

[1]The original proof, which can be found in the appendix of the referenced paper, considers time $O(\log^2(nm))$ because it considers that the entries are encoded in $\log(nm)$ bits. Similarly to Chakraborty et al. [6, Theorem 4], we do not consider this overhead, as one might want to tune the number of bits to the required precision. Note that we generally omit logarithmic overheads due to the precision of binary encodings and hardware limitations.

Given a vector $\vec{x} \in R^n$, stored in this data structure, we can create access to $|\vec{x}\rangle = \frac{1}{\|\vec{x}\|}\sum_{i=1}^n x_i |i\rangle$ in $O(\text{polylog}(n))$ time.

Quantum algorithms built using this definition of data access (Def. IV.1) usually incur in running time overheads that depend on the matrix normalization factor $\|A\|_F$. In a subsequent work, Kerenidis and Prakash [16] showed that it is possible to preprocess A, during the storage step, to change the dependency on $\|A\|_F$ to $\mu_p(A)$, defined as follows.

Definition IV.3 (Parameter $\mu_p(A)$)**.** *Let $A \in R^{n\times m}$. Then,*

$$\mu_p(A) = \sqrt{s_{2p}(A)s_{2(1-p)}(A^T)}, \tag{6}$$

where $s_p(A) = \max_{i\in[n]} \|\vec{a}_i\|_p^p$.

This alternative parameter reduces the running time of quantum algorithms requiring access to matrices with particular row/column norms properties. In practice, without any promise on the input matrix, in the storage step, one can search for the best $\mu_p(A)$ over a finite set of $p \in \mathcal{P} \subset [0, 1]$, with $|\mathcal{P}| \in O(1)$ and compare it with $\|A\|_F$ to choose the smallest. This procedure still takes time and space $\widetilde{O}(nm)$, while enabling for overhead

$$\mu(A) = \min(\|A\|_F, \mu_p(A)). \tag{7}$$

While many quantum linear algebra operations can be implemented using this data access and phase estimation [15, 16], the most efficient and up-to-date techniques rely on block-encoding the matrix in a unitary operator. Informally, this means embedding the matrix A, scaled by factor α and encoded with precision ϵ, into the top-left corner of a unitary U_A which possibly acts on extra qubits:

$$U_A = \begin{pmatrix} A/\alpha & \cdot \\ \cdot & \cdot \end{pmatrix}. \tag{8}$$

Without loss of generality, we can assume that A is an operator acting on s qubits, eventually padded with zeros. More formally, the definition of block-encoding is the following.

Definition IV.4 (Block-encoding [6])**.** *Suppose that A is an s-qubit operator, $\alpha, \epsilon \in \mathbb{R}^+$ and $q \in N$. We say that the $(s + q)$-qubit unitary U_A is an (α, q, ϵ) block-encoding of A, if*

$$\left\| A - \alpha(\langle 0|^{\otimes q} \otimes \mathbb{I})U_A(|0\rangle^{\otimes q} \otimes \mathbb{I})\right\| \le \epsilon,$$

where $\|\cdot\|$ is the operator norm.

When a matrix A is stored in a quantum accessible data strucure, it is possible to implement a block-encoding of its symmetric embedding $\overline{A} = \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix}$ in time $\widetilde{O}(1)$.

Theorem IV.5 (Implementing block-encodings from quantum data structures [6, Theorem 4])**.** *Let $A \in R^{n\times m}$.*

1) *Fix $p \in [0, 1]$. If $A^{(p)}$, and $(A^{(1-p)})^T$ are stored in quantum accessible data structures, then there exist unitaries U_R and U_L that can be implemented in time $O(\text{polylog}(nm/\epsilon))$ such that $U_R^\dagger U_L$ is a $(\mu_p(A), \lceil log(n + m + 1)\rceil, \epsilon)$-block-encoding of $\overline{A}$.*

2) *On the other hand, if A is stored in quantum accessible data structure, then there exist unitaries U_R and U_L that can be implemented in time $O(\mathrm{polylog}(nm/\epsilon))$ such that $U_R^\dagger U_L$ is a $(\|A\|_F, \lceil log(n+m+1)\rceil, \epsilon)$-block-encoding of $\overline{A}$.*

In Appendix A, we show that, for our purposes, having a block-encoding of $\overline{A}$ is equivalent to having a block-encoding of A. To sum up, we can store A in a quantum accessible data structure in time and space $O(nm\log^2(nm))$ to have efficient quantum access (as per Def. IV.1) or to implement a block-encoding (as per Def. IV.4) in time $\widetilde{O}(1)$, with $\alpha = \mu(A)$, as per Eq. 7. The structures are different if $\mu(A) \neq \|A\|_F$.

B. Data processing

Once defined the data access methods, we can focus on the three main processing techniques: matrix-vector product, distance estimation, and finding the minimum.

Using the block-encoding framework it is possible to perform matrix-vector multiplication efficiently. More precisely, here the task is to create a quantum state $|\vec{z}\rangle = \frac{A\vec{x}}{\|A\vec{x}\|}$, given a matrix A and a vector $\vec{x}$. At the same time, it is possible to efficiently estimate the the ratio between the norm of the resulting vector and the original one $\frac{\|A\vec{x}\|}{\|\vec{x}\|}$, which will be handy later for our classification algorithm. The algorithm applies the block-encoding to the desired quantum state and performs amplitude amplification or estimation [4, 32]. While the results of Theorem IV.6 and Corollary IV.7 are inspired by commonly used techniques, we taylor them for our specific needs. We state the result in the main text and invite the interested reader to consult the proofs in Appendix A.

Theorem IV.6 (Matrix-vector multiplication). *Let U_A be a (α, q, ϵ_0)-block-encoding of a matrix $A \in \mathbb{R}^{n\times n}$, implementable in time T_A. Let there be quantum access to a vector $\vec{x} \in \mathbb{R}^n$ in time T_x. Let $\epsilon > 0$. There exist quantum algorithms that output:*

1) *A classical estimate $\overline{t}$ of $t = \frac{\|A\vec{x}\|}{\|\vec{x}\|}$ such that $\left|t - \overline{t}\right| \leq \epsilon$ with high probability in time $O\left((T_A + T_U)\frac{\alpha}{\epsilon}\right)$, provided $\epsilon_0 \leq \frac{\epsilon}{c}$, for some constant c.*
2) *A classical estimate t of $\|A\vec{x}\|$ such that $|\|A\vec{x}\| - t| \leq \eta\|A\vec{x}\|$ with high probability in expected time $\widetilde{O}\left((T_A + T_X)\frac{\alpha}{\epsilon}\frac{\|\vec{x}\|}{\|A\vec{x}\|}\right)$, provided $\epsilon_0 \leq \frac{\epsilon}{c}$, for some known constant c.*
3) *A quantum state $|\vec{z}\rangle$ such that $\left\||\vec{z}\rangle - \frac{A\vec{x}}{\|A\vec{x}\|}\right\| \leq \epsilon$ in time $\widetilde{O}\left((T_A + T_X)\frac{\alpha}{\gamma}\right)$, provided that we know some $\gamma \leq \frac{\|A\vec{x}\|}{\|\vec{x}\|}$ and that $\epsilon_0 \leq \frac{\epsilon\gamma}{2}$.*
4) *A quantum state $|\vec{z}\rangle$ such that $\left\||\vec{z}\rangle - \frac{A\vec{x}}{\|A\vec{x}\|}\right\| \leq \epsilon$ in expected time $\widetilde{O}\left((T_A + T_X)\alpha\frac{\|\vec{x}\|}{\|A\vec{x}\|}\right)$ if $\frac{\|A\vec{x}\|}{\|\vec{x}\|} \neq 0$ and otherwise runs forever. The correctness requires $\epsilon_0 \leq \frac{\epsilon\|A\vec{x}\|}{2\|\vec{x}\|}$.*

It is possible to extend this result to when A and $\vec{x}$ are stored in quantum accessible data structures, assuming they are stored with enough bit precision.

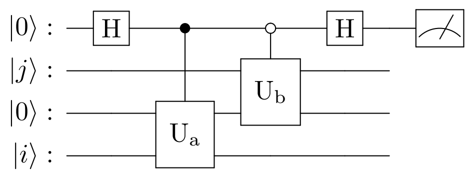


Fig. 2: Quantum circuit for estimating $\langle\vec{a}_i|\vec{b}_i\rangle$. Assume $U_a|i\rangle|0\rangle = |i\rangle|\vec{a}_i\rangle$ and $U_b|j\rangle|0\rangle = |j\rangle|\vec{b}_j\rangle$. The probability of measuring the auxiliary qubit in the state $|1\rangle$ at the end of the circuit is $P = \frac{1-\langle\vec{a}_i|\vec{b}_j\rangle}{2}$.

Corollary IV.7 (Matrix-vector multiplication with quantum data structures). *Let $A \in \mathbb{R}^{n\times m}$ and $\vec{x} \in \mathbb{R}^m$ stored in a quantum data structure. There exist quantum algorithms that output:*

1) *A classical estimate $\overline{t}$ of $t = \frac{\|A\vec{x}\|}{\|\vec{x}\|}$ such that $\left|t - \overline{t}\right| \leq \epsilon$ with high probability in time $\widetilde{O}\left(\frac{\mu(A)}{\epsilon}\right)$.*
2) *A classical estimate t of $\|A\vec{x}\|$ such that $|\|A\vec{x}\| - t| \leq \eta\|A\vec{x}\|$ with high probability in expected time $\widetilde{O}\left(\frac{\mu(A)}{\epsilon}\frac{\|\vec{x}\|}{\|A\vec{x}\|}\right)$.*
3) *A quantum state $|\vec{z}\rangle$ such that $\left\||\vec{z}\rangle - \frac{A\vec{x}}{\|A\vec{x}\|}\right\| \leq \epsilon$ in time $\widetilde{O}\left(\frac{\mu(A)}{\gamma}\right)$, provided that we know some $\gamma \leq \frac{\|A\vec{x}\|}{\|\vec{x}\|}$.*
4) *A quantum state $|\vec{z}\rangle$ such that $\left\||\vec{z}\rangle - \frac{A\vec{x}}{\|A\vec{x}\|}\right\| \leq \epsilon$ in expected time $\widetilde{O}\left(\mu(A)\frac{\|\vec{x}\|}{\|A\vec{x}\|}\right)$ if $\frac{\|A\vec{x}\|}{\|\vec{x}\|} \neq 0$ and otherwise runs forever.*

The second ingredient of our algorithm is a routine for estimating inner products and Euclidean distances.

Theorem IV.8. *[Distance and Inner Products Estimation [17]] Assume for a matrix $A \in \mathbb{R}^{n\times d}$ and a matrix $B \in \mathbb{R}^{k\times d}$ that the following unitaries $|i\rangle|0\rangle \mapsto |i\rangle|\vec{a}_i\rangle$, and $|j\rangle|0\rangle \mapsto |j\rangle|\vec{b}_j\rangle$ can be controlled and executed in times T_1 and T_2 respectively, and the norms of the vectors are known. For any $\Delta > 0$ and $\epsilon > 0$, there exists a quantum algorithm that computes:*

- *$|i\rangle|j\rangle|0\rangle \mapsto |i\rangle|j\rangle\left|\overline{d^2(\vec{a}_i,\vec{b}_j)}\right\rangle$ where $|\overline{d^2(\vec{a}_i,\vec{b}_j)} - d^2(\vec{a}_i,\vec{b}_j)| \leqslant \epsilon$ w.p. $\geq 1-\Delta$*
- *$|i\rangle|j\rangle|0\rangle \mapsto |i\rangle|j\rangle\left|\overline{(\vec{a}_i,\vec{b}_j)}\right\rangle$ where $|\overline{(\vec{a}_i,\vec{b}_j)} - (\vec{a}_i,\vec{b}_j)| \leqslant \epsilon$ w.p. $\geq 1-\Delta$*

in time $\widetilde{O}\left(\frac{\|\vec{a}_i\|\|\vec{b}_j\|}{\epsilon}(T_1 + T_2)\log(1/\Delta)\right)$.

The building block of the algorithm discussed in this result is shown in Fig. 2. One can obtain an additive ϵ estimate of $\langle\vec{a}_i|\vec{b}_j\rangle = \frac{(\vec{a}_i,\vec{b}_j)}{\|\vec{a}_i\|\|\vec{b}_j\|}$ in time $O((T_1 + T_2)\frac{1}{\epsilon})$ by executing amplitude estimation on that circuit, on the auxiliary qubit being in the state $|1\rangle$. If we know the norm of the vectors, we can run the estimation with precision $\frac{\epsilon}{\|\vec{a}_i\|\|\vec{b}_j\|}$ to estimate $(\vec{a}_i,\vec{b}_j)$ to additive precision ϵ. Knowing the inner product, we can compute the squared Euclidean distance as

$$\|\vec{a}_i - \vec{b}_j\|^2 = \|\vec{a}_i\|^2 + \|\vec{b}_j\|^2 - 2(\vec{a}_i, \vec{b}_j). \quad (9)$$

Finally, the last ingredient is an algorithm to find the minimum value of a vector and its index. Durr and Hoyer [9] propose a routine that uses several iterations of amplitude amplification, each with different oracles, and achieve a quadratic speedup on its classical counterpart if the search is unstructured.

Theorem IV.9 (Finding the minimum [9])**.** *Let there be quantum access to a vector $\vec{u} \in [0,1]^N$ via the operation $|j\rangle|0\rangle \rightarrow |j\rangle|u_j\rangle$ in time T. Then, we can find the minimum value $u_{\min} = \min_{j\in[N]} |u_j|$ and its index $j_{\min} = \arg\min_{j\in[N]} u_j$ w.p. greater than $1-\Delta$ in time $O\left(T\sqrt{N}\log\left(\frac{1}{\Delta}\right)\right)$.*

Variations of this routine allow retrieving the K minima values and indexes with probability $\geq 1-\Delta$ in time $O(T\sqrt{KN}\log(1/\Delta))$ [10, 21]. These algorithms can be used to extend our study to K-Nearest-Neighbor classifiers.

Before proceeding with the quantum algorithm, there is an important caveat that we want to point out. The routine for finding the minimum requires access to the vector's state preparation oracle and its inverse. However, we often build this oracle by combining quantum routines that are not "exact". As a result, we often have access to an *approximate* state preparation oracle, and therefore a more careful correctness and complexity analysis is needed. In our case, the approximation will be given by the matrix-vector routine's error and the distance estimation's probability of failure. Some examples of similar scenarios can be found in Wiebe et al. [30], Chen and de Wolf [7]. Similarly to the above-mentioned analysis, we can ignore these approximation terms at the cost of some polylogarithmic running time overhead.

V. Quantum algorithm

With the tools laid out, we are ready to discuss the quantum algorithms. We focus on the classification steps and consider that the training has been performed on a classical computer; i.e., the quantum computer is only used in the prediction stage, whose time efficiency is usually more relevant in industrial applications. However, the interested reader can consult Bellante et al. [3, Theorems 10, 12] for a method to extract the top-k principal components and the total retained amount of variance from a matrix stored in a quantum accessible data structure.

A. Data loading and preprocessing

To run the classification algorithm, we need classical access to the threshold parameters $\delta_1, \delta_2, \gamma$, and quantum access to the linear feature mapping matrix $A \in \mathbb{R}^{k\times m}$, the neighbors/centroid matrix $C \in \mathbb{R}^{p\times k}$, and the mean-centered sample $\vec{u} \in \mathbb{R}^m$. More precisely, we will need: 1. efficient quantum access to the rows of C via $U_C|j\rangle|0\rangle = |j\rangle|\vec{c}_j\rangle$, for $j \in [p]$ (Def. IV.1, only U); 2. a block-encoding U_A of A; 3. efficient quantum access to $\vec{u}$ via a unitary $U_u|0\rangle = |\vec{u}\rangle$. Each of these unitaries needs to be controllable. It might be that we get quantum access to this data in a natural way, as a result of some quantum processes or algorithms. In that case, we assume that the costs of implementing these controlled unitaries are T_C, T_A, T_u, respectively.

Algorithm 2 Quantum Eigenfaces-based classification

```
Input: Quantum access, as per Sec. V-A, to the linear mapping $A \in \mathbb{R}^{k\times m}$ (eigenfaces), to the set $\{\vec{c}_1, \cdots, \vec{c}_p\}$ of stored weights vectors, with $\vec{c}_j \in \mathbb{R}^k$, and to the mean-shifted target sample $\vec{u} \in \mathbb{R}^m$. Classical access to the label vector $\vec{y} \in [P]^p$ and to the thresholds $\delta_1, \delta_2 \in \mathbb{R}_{>0}$.
Output: an integer indicating the classification outcome.

1: Map $\vec{u}$ in the new feature space using using matrix-vector multiplication:
       $|\Phi(\vec{u})\rangle = \frac{A\vec{u}}{\|A\vec{u}\|}$
2: Perform Euclidean distance estimation in superposition with $|\Phi(\vec{u})\rangle$ and the centroids unitary, to produce an approximation of:
       $|\varphi\rangle = \sum_{j\in[p]} |j\rangle \left|d^2(\Phi(\vec{u}), \vec{c}_j)\right\rangle$
3: Execute finding the minimum on $|\varphi\rangle$ to obtain the minimum distance estimate $\overline{d}$ and the index $j^*$ of the corresponding weight vector
4: Output: if $\overline{d} \leq \delta_1$: same class of $\vec{c}_{j^*}$, output $y_{j^*}$;
           if $\delta_1 < \overline{d} < \delta_2$: similar element, output $-1$.
           if $\overline{d} \geq \delta_2$: not a similar element, output $-2$.
```

However, when the data comes from classical problems, we create quantum access with the following procedures. To build efficient quantum access to the rows of C, $\{\vec{c}_j\}_{j=1}^p$ for $\vec{c}_j \in \mathbb{R}^k$, we can use the data structure in Theorem IV.2. Initializing it costs a preprocessing time $\widetilde{O}(nk)$. Similarly, we can efficiently implement a block-encoding of A with $\alpha = \mu(A)$ using the data structure of Lemma IV.5, in time $\widetilde{O}(km)$. Alternatively, it might be possible to compile a circuit for the block-encoding of A using some classical processing [5]. For some specific matrices A it might be possible to achieve $\alpha = \|A\|$ and polylog circuit depth/size, even without a QRAM. Finally, we can compute the mean vector $\overline{x}$ from the training set $X \in \mathbb{R}^{n\times m}$ in time $O(nm)$ and store it in a classical device. None of these costs affects the time analysis of classification, as the data structures do not need to change during the algorithm and are not a function of the target sample. On the other hand, when we receive the target sample $\vec{x} \in \mathbb{R}^m$, we need to subtract the mean and store the resulting vector $\vec{u} = \vec{x} - \overline{x}$ in a quantum accessible data structure, as per Theorem IV.2. This operation can be considered part of the classification steps and has a time complexity $\widetilde{O}(m)$. While it should not be forgotten, we will omit it in the running times, as it is comparable to the cost of acquiring the classical data in a classical memory.

B. Classification and outlier detection

Let us start by assuming quantum access, as defined in the previous section, to the rows of C, the feature mapping A, and the sample $\vec{u} = \vec{x} - \overline{x}$, in times T_C, T_A, T_u, respectively. We consider the cost of a generic linear mapping $\Phi(\vec{u}) = A\vec{u}$ and then discuss the specific case for $A = V$, as in the eigenfaces setting. The high level procedure is summarized in Alg. 2.

The first step consists in performing the linear feature mapping $A\vec{u}$. To this aim, we can use Theorem IV.6/Corollary IV.7. The procedure succeeds in expected time (point 4):

$$\widetilde{O}\left(\alpha\frac{\|\vec{u}\|}{\|\Phi(\vec{u})\|}(T_A + T_u)\right). \tag{10}$$

This step prepares an approximation of the state $|\Phi(\vec{u})\rangle = \frac{A\vec{u}}{\|A\vec{u}\|} = \frac{1}{\|\vec{w}\|}\sum_{i=1}^{k} w_i |i\rangle$. Using a controlled version of this algorithm, and the unitary $U_C|j\rangle|0\rangle = |j\rangle|\vec{c}_j\rangle$, we can use the Euclidean distance estimation routine of Theorem IV.8. The combination of the two routines runs in expected[2] time

$$\widetilde{O}\left(\frac{\max_{j\in[p]}(\|\vec{c}_j\|)\|\Phi(\vec{u})\|}{\epsilon}\left(T_C + \alpha\frac{\|\vec{u}\|}{\|\Phi(\vec{u})\|}(T_A + T_u)\right)\right), \tag{11}$$

where we use $\max_{j\in[p]}(\|\vec{c}_j\|)$ to bound the error of all the estimated distances $d^2(\vec{c}_j, \Phi(\vec{u}))$ in superposition. This step prepares an approximation of $|\vec{\varphi}\rangle = \frac{1}{\sqrt{n}}\sum_{j=1}^{n}|j\rangle\left|d^2(\Phi(\vec{u}), \vec{c}_j)\right\rangle$. The approximation is given both by the distance estimation routine and the matrix-vector product, and it is both in the amplitudes and in the bits that encode the distance in the second register. The block encoding error ϵ_0 on A, induces an error on the distances of $|d^2(\Phi(\vec{u}), \vec{c}_i)) - \widetilde{d}^2(\Phi(\vec{u}), \vec{c}_i)| \leq 2\epsilon_0\|\vec{u}\|\max_{i\in[n]}(\|\vec{c}_i\|)$ (see Eq. 9). Running the distance estimation routine on these approximated distances, with parameters ϵ_1 and Δ, produces a state $|\vec{\varphi}\rangle = \frac{1}{\sqrt{n}}\sum_{j=1}^{n}|j\rangle|\hat{d}^2(\Phi(\vec{u}), \vec{c}_j)\rangle$, where each $|\hat{d}^2(\Phi(\vec{u}), \vec{c}_j)\rangle$ is a superposition of some computational basis states $|\overline{d}^2(\Phi(\vec{u}), \vec{c}_j)\rangle$ that are ϵ_1 far from $|\widetilde{d}^2(\Phi(\vec{u}), \vec{c}_j)\rangle$ and show up with probability $1 - \Delta$, and other states that are more than ϵ_1 apart, according to the theorem statement. Reducing the parameter Δ in the distance estimation mitigates the error on the amplitudes. On the other hand, the overall error on the distances is $|d^2(\Phi(\vec{u}), \vec{c}_i)) - \overline{d}^2(\Phi(\vec{u}), \vec{c}_i))| \leq \epsilon_1 + 2\epsilon_0\|\vec{u}\|\max_{i\in[n]}(\|\vec{c}_i\|)$. Dividing the error equally among the two terms, we obtain $\epsilon_1 \leq \epsilon/2$, which justifies Eq. 11, and $\epsilon_0 \leq \frac{\epsilon}{4\max_{i\in[p]}(\|\vec{c}_i\|)\|\vec{u}\|}$, which translates in a requirement on the block-encoding error and scales polylogarithmically if the matrix is stored in a data structure.

Once we have approximate access to the quantum state $|\vec{\varphi}\rangle$, we can execute the procedure for minimum finding (see below Theorem IV.9 for references on minimum finding with approximate amplitudes). In particular, we can extract $\min_{j\in[n]}(\overline{d}^2(\Phi(\vec{u}), \vec{c}_j))$ and the corresponding index j^*, in expected[2] time

$$\widetilde{O}\left(\sqrt{p}\frac{\max_{j\in[n]}(\|\vec{c}_j\|)\|\Phi(\vec{u})\|}{\epsilon}\left(T_C + \alpha\frac{\|\vec{u}\|}{\|\Phi(\vec{u})\|}(T_A + T_u)\right)\right) \tag{12}$$

[2]Even though the Euclidean distance estimation and minimum finding routines run in worst case bounded time, the procedure of Eq. 10 makes the overall time expected. We address the issue in Sec. V-B3.

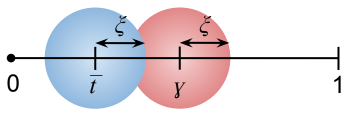


Fig. 3: Visualization of the norm-based outlier detection error.

After extracting the minimum and its index, one can conclude the algorithm as in the classical case, comparing the distance to the thresholds δ_1, δ_2 and outputting either j^*, -1, or -2.

When A, C, and $\vec{u}$ are stored in quantum accessible data structures, the access costs become $\widetilde{O}(1)$ and the expected running time becomes

$$\widetilde{O}\left(\sqrt{p}\mu(A)\frac{\max_{i\in[p]}(\|\vec{c}_i\|)\|\vec{u}\|}{\epsilon}\right). \tag{13}$$

In the eigenfaces classification algorithm, and in many others, the linear mapping of choice is the PCA dimensionality reduction. In this case, the matrix $A = V \in \mathbb{R}^{k\times m}$, with $k \leq m$, is a projector and has singular values equal to 1 or 0. Using this fact, we can bound $\mu(A) \leq \|A\|_F = \sqrt{k}$. The expected running time can be bounded as

$$\widetilde{O}\left(\sqrt{pk}\frac{\max_{i\in[p]}(\|\vec{c}_i\|)\|\vec{u}\|}{\epsilon}\right), \tag{14}$$

providing a quasi-quadratic speed-up over the textbook classical algorithm $O(mk + pk)$.

This advantage would be even more marked if we could avoid using a QRAM and design an efficient block encoding of V with $\alpha = \|V\| = 1$, eliminating the term $\sqrt{k}$. While this might happen for some structured matrices V, in the general case there might be a tradeoff between the optimal α, the number of auxiliary qubits and the circuit depth.

1) Norm-based outlier detection: As discussed in Sec.III-B1, we can enhance the outlier detection with a norm-based classification, exploiting the fact that outliers will be scarcely supported on the principal component space. For this, we are given a threshold $\gamma \in [0, 1]$. For each sample, we estimate the ratio of preserved norm $\frac{\|\Phi(\vec{u})\|}{\|\vec{u}\|}$ and compare it to γ: if it is lower, we label the sample as an outlier (output -2 and do not proceed further with the classification).

Using Theorem IV.6 or Corollary IV.7 (with data structures), we can estimate the ratio $\frac{\|\Phi(\vec{u})\|}{\|\vec{u}\|}$ to additive accuracy ξ in respective times

$$\widetilde{O}\left(\frac{\alpha}{\xi}(T_A + T_u)\right) \text{ or } \widetilde{O}\left(\frac{\mu(A)}{\xi}\right), \tag{15}$$

where we can use again $\mu(A) \leq \sqrt{k}$ for PCA. The classical textbook algorithm for this step would have complexity $O(mk)$, dominated by matrix-vector multiplication. In practice, as we see experimentally, one can think of γ as a quantity substantially greater than 0.2 and set a constant error $\xi = 0.01$, achieving considerable speed-ups on high-dimensional data.

2) Uncertainty areas: There is a major difference between the classical algorithms and their quantum versions: the latter introduces some error on the distance and the norm estimation. These errors ϵ and ξ can affect the classifier performances and must be chosen by evaluating a trade-off between the running time and the metrics of interest (e.g., accuracy, recall). Moreover, they create some uncertainties during the classification, defining some grey area around the thresholds $\delta_1, \delta_2, \gamma$. Fig. 3 offers a visual representation of the problem for the norm-based outlier detection: if the norm ratio estimate $\bar{t}$ falls in $[\gamma - \xi, \gamma + \xi]$ (the *red* area) we cannot reliably label the data point, as the true t (somewhere in the *blue* area) can be either greater or smaller than γ. The problem occurs similarly for the comparison with δ_1 and δ_2. We propose two ways to handle this uncertainty: 1. one can continue repeating the algorithm by halving the error until the uncertainty is resolved. The algorithm will terminate with expected time that scales with the inverse of the gap between the threshold and the true estimate (e.g., $\widetilde{O}(1/|t - \gamma|)$ in Fig. 3). 2. live with the uncertainty. Decide between always rejecting, always accepting, or any random combination of the two, and choose errors that work empirically.

3) Expected to worst time and failure probability: Both the quantum classification algorithm of Sec. V-B and the halving routine described above run in *expected* time. While their expected time is particularly efficient, the algorithms could run forever in some particularly unlucky cases, which is not desirable when deployed for automated detection in industrial applications. In particular, the issue appears in the halving problem because we do not know lower bounds for $|\gamma - t|$, and in Eq. 10 because we do not know lower bounds for the ratio $\frac{\|\Phi(\vec{u})\|}{\|\vec{u}\|}$. If we knew lower bounds, we could run the expected time version of the algorithm and stop them when the time exceeds the lower bounds, knowing whether they failed or not (hence moving the uncertainty from the time to the success of the routine). In the first case, we can define an arbitrary lower bound ξ', and give up identifying norm ratios more than ξ close to γ, setting an effective timeout at $\widetilde{O}(1/\xi')$. In the second case, when the norm-based outlier detection anticipates the classification, we can use that $\gamma \leq \frac{\|\Phi(\vec{u})\|}{\|\vec{u}\|}$ and set the timeout to $\widetilde{O}\left(\sqrt{p}\mu(A)\frac{\max_{i\in[p]}(\|\vec{c}_i\|)\|\Phi(\vec{u})\|}{\gamma\epsilon}\right)$. The algorithms will then shift the uncertainty from the running time to the probability of success. While an accurate analysis of the failure probability is not in the scope of this manuscript, we remind the reader that if the algorithm succeeds probability $> 1/2$, one can run the algorithm $O(\log(1/\delta))$ times and output the prediction by majority vote, increasing the probability of success to an arbitrary $1 - \delta$, by Chernoff bound.

VI. Numerical experiments

We conducted numerical experiments[3] on two datasets for face recognition and one for image classification. First, we assessed the utility of the norm-based outlier detection step in the classical scenario. Secondly, we studied the performance of the quantum algorithm at increasing values of error ϵ. Lastly, we compared the quantum running times for each value of ϵ (as defined in Eq. 13) with the classical counterpart $O(mk + pk)$.

[3]Code available at https://github.com/WilliamBonvini/quantum-eigenfaces

TABLE I: Dataset Split Information

Dataset	m	Outliers	k	#Training	#Validation	#Test
ORL	10304	No	70	288	36	36
	10304	Yes	70	288	54	54
YALE	32256	No	80	322	69	70
	32256	Yes	80	322	149	134
MNIST	784	No	60	49000	10500	10500
	784	Yes	60	49000	14000	14000

A. Datasets

1) Face Recognition: We experimented with two well-known datasets for face recognition: the Olivetti Research Lab Dataset (*ORL*) [24] and the Yale Face Extended B Cropped Dataset [11, 19] (*Yale*). *ORL* includes 400 gray-scale images of 40 unique individuals, each with 10 images. All images are sized at 112×92 pixels. *Yale* comprises 2414 frontal-face images with size 192×168 over 38 subjects and about 64 images per subject. The images in both datasets were captured under different lighting conditions and facial expressions. Due to numerous images within *Yale* characterized by challenging lighting conditions, we constrained the analysis to images in which subjects are more discernible.

2) Image Classification: We relied on the popular *MNIST* [18] dataset, which consists of $70,000$ gray-scale images of handwritten digits from 0 to 9. We introduced outliers by sampling data from *Fashion MNIST* [31]: a dataset of $70,000$ gray-scale images of clothing items. Both datasets have a resolution of 28×28 pixels. Clothing items were labeled with an outlier label (e.g., -2).

B. Methodology

We split each dataset into a training, a validation, and a test set (see Table I for details). The validation set was used to tune the hyper-parameters k and δ: k is the number of principal components that determine the dimensionality of the weight vectors. We chose it to preserve at least 85% of the training set's variance. Depending on the experiment, δ is synonymous with δ_1 or δ_2 (as defined in Sec. III-B). We chose not to rely on both the distance thresholds simultaneously for our experiments. The *face recognition* experiments detect the correct subject given an image in a dataset that contains only faces (no images from other distributions). In this case, δ's goal was to discern known subjects from unknown subjects; therefore, $\delta = \delta_1$. In this scenario, with a slight abuse of terminology, we use the term *outliers* to refer to face images from unknown subjects. On the other hand, the *image classification* task consists of correctly identifying digits from *MNIST* and detecting outliers, defined as images of clothing items. In this case, δ aimed to discern digits from actual outliers; therefore, $\delta = \delta_2$. In selecting δ, we tried to find a compromise between classification and outlier detection metrics. An example is shown in Fig. 4. The norm threshold γ was set to $\max_{\vec{x}\in X}\frac{\|V(\vec{x}-\overline{x})\|}{\|(\vec{x}-\overline{x})\|}$, where X is the training set.

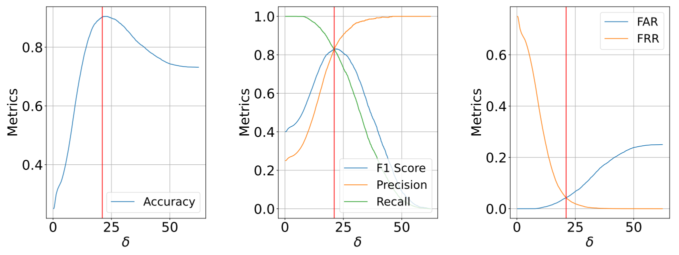


Fig. 4: Selection of δ for the image classification task.

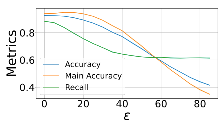
TABLE II: Numerical Experiments Results

Dataset	δ	Outliers	γ	ϵ	Run. time	Acc.	Main Acc.	Recall
ORL	∞	No	-	0	$7.41 \cdot 10^5$	-	0.944	-
	74.38	Yes	-	0	$7.41 \cdot 10^5$	0.870	0.833	0.944
	74.38	Yes	0.75	0	$7.41 \cdot 10^5$	0.870	0.833	0.944
	74.38	Yes	0.75	15	$3.12 \cdot 10^3$	0.849	0.829	0.888
YALE	∞	No	-	0	$2.61 \cdot 10^6$	-	0.986	-
	232.0	Yes	-	0	$2.61 \cdot 10^6$	0.888	0.900	0.875
	232.0	Yes	0.94	0	$2.61 \cdot 10^6$	0.940	0.900	0.984
	232.0	Yes	0.94	100	$2.84 \cdot 10^3$	0.910	0.866	0.959
MNIST	∞	No	-	0	$2.99 \cdot 10^6$	-	0.975	-
	23.52	Yes	-	0	$2.99 \cdot 10^6$	0.906	0.940	0.803
	22.34	Yes	0.75	0	$2.99 \cdot 10^6$	0.927	0.940	0.885
	22.34	Yes	0.75	15	$6.66 \cdot 10^3$	0.913	0.949	0.804

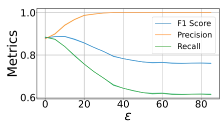
We examine four situations for each dataset. The first one entails searching for the nearest neighbor in an outlier-free configuration. The remaining three consider a setting with outliers: without γ thresholding, with γ thresholding, and a simulation of the quantum algorithm that, besides relying on γ, accounts for the presence of errors in the distance and norm ratio estimation. In particular, we artificially introduced uniform error within $[-\epsilon, \epsilon]$ in the distance estimation for various ϵ, and uniform error $\xi \in [-0.01, 0.01]$ in the norm-based outlier detection. When the perturbed quantities became negative, we truncated them to 0. We repeated our experiments 100 times for each value of ϵ and computed the average performance. While the choice of a uniform error is pessimistic (compared to the amplitude estimation noise distribution), we still obtained promising results.

1) Metrics: Performances were assessed with the following metrics. *Accuracy*: the number of correct predictions over all the predictions. *Main Accuracy*: the number of samples from the classes in the training set correctly predicted over their total. *False Acceptance Rate (FAR)*: the number of outliers classified as inliers over the total test samples. *False Rejection Rate (FRR)*: the number of inliers misclassified as outliers over the total number of test samples. Precision, Recall, and F1-Score are framed with respect to outliers. *Precision*: the number of outliers correctly detected over the total number of samples predicted as outliers. *Recall*: the number of outliers correctly detected over the totality of outliers. *F1 Score*: defined as $2 \cdot Precision \cdot Recall/(Precision + Recall)$.

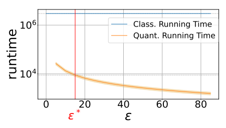
Intuitively, *Main accuracy* helps us assess the accuracy of the original classification task while *Recall* the number of outliers correctly detected. *Accuracy* combines the two.



(a) Classification



(b) Outlier Detection



(c) Running time

Fig. 5: Performance on varying values of ϵ for the image classification task with outliers and norm-based detection.

C. Performance and quantum running time

Table II presents the results of the classical (first three lines per dataset) and quantum (in green) performance and running times. We see how introducing δ (2nd line) causes some performance decrease but enables detecting many outliers. On *Yale* and *MNIST*, introducing the threshold γ (3rd line) enables the detection of more outliers. In all three datasets, the value for γ, which can be used to bound the worse case quantum time, is reasonably close to one. The quantum algorithm's performance shows that the distance estimation error ϵ negatively influences the classification results. On the other hand, we notice a substantial running time improvement over the classical counterparts, for all datasets. In particular, we measure a $\simeq 10^2$ improvement for *ORL* and a $\simeq 10^3$ improvement for *Yale* and *MNIST*. We report a more detailed study of the performance on the *Image Classification* task at increasing ϵ values in Fig. 5. We plot average and standard deviation performance and running times. Introducing the error ϵ leads to outliers finding min distances closer to the threshold δ, causing a decrease in Recall and an increase in Precision. At high values of ϵ, the Recall stabilizes to the quantity detected by the ξ-corrupted norm-based recognition.

VII. Conclusion

In this paper, we studied a quantum version of the eigenfaces classification algorithm for face recognition, a milestone result for classical machine learning. We describe the classification procedure in an end-to-end way, assuming a quantum memory, and by means of two algorithms: a linear feature mapping + nearest neighbor/centroid classifier and a norm-based outlier detector. The two main algorithms are of independent interest and go beyond the eigenfaces face recognition task. The norm-based outlier detection routine can be used as a support to any algorithm employing PCA and dealing with outliers. Similarly, the quantum algorithm of Sec. V-B can be used for any linear feature mapping followed by a nearest neighbor/centroid classifier. When working with classical data, we obtain asymptotic complexities $\widetilde{O}(\mu(A)/\xi)$ and $\widetilde{O}\left(\sqrt{p}\mu(A)\frac{\max_{i\in[p]}(\|\vec{c}_i\|)\|\vec{u}\|}{\epsilon}\right)$, as opposed to the classical counterparts $O(mk)$ and $O(mk + pk)$. Our experiments show that the algorithms effectively recognize the outliers while maintaining good classification accuracy. Moreover, the running time parameters are reasonable and might enable for some advantage even on small datasets. However, while the experimental results are encouraging, we obtain them while disregarding polylogarithmic terms and current hardware limitations. As the theoretical advantage of the presented algorithms is roughly quadratic, these algorithms are unlikely to be advantageous in an early fault-tolerant era, where error correction will introduce a considerable overhead [1]. We stress, though, that the simplicity of the procedures and the relevance of their classical counterparts make them suitable for implementations by future quantum machine learning practitioners, and we cannot exclude they might provide relevant practical advantages in a well-developed fault-tolerant era. We leave for future work further experiments with more realistic error models and large-scale datasets, as well as the study of more efficient block-encodings for projectors V.

Acknowledgment

A.B. and S.Z. would like to thank professors Donatella Sciuto and Ferruccio Resta for their support. A.B. would like to thank Patrick Rebentrost, for hosting him at CQT at the time of writing, and João Doriguello, Alessandro Luongo, and Jinge Bao for useful discussions.

Appendix

A. Matrix-vector multiplication

In this section, we prove Theorem IV.6 and Corollary IV.7. The first proof works in the general block-encoding framework, while the second assumes the matrix is stored in a quantum accessible data structure (Lemma IV.5).

1) Theorem IV.6:

Proof. Assume a (α, q, ϵ_0)-block-encoding U_A of A implemented in T_A and a unitary U_x that prepares $|\vec{x}\rangle$ in time T_U. From the definition of block-encoding, we have

$$\left\| A - \alpha(\langle 0|^{\otimes q} \otimes \mathbb{I})U_A(|0\rangle^{\otimes q} \otimes \mathbb{I})\right\| \leq \epsilon_0, \tag{16}$$

Let us define $A' = (\langle 0|^{\otimes q} \otimes \mathbb{I})U_A(|0\rangle^{\otimes q} \otimes \mathbb{I})$ as the matrix on the top-left corner of U_A, such that $\left\|\frac{A}{\alpha} - A'\right\| \leq \frac{\epsilon_0}{\alpha}$. Then,

$$\begin{aligned} U_A(I^{\otimes q} \otimes U_x)\,|0\rangle &= U_A\,|0\cdots 0\rangle\,|\vec{x}\rangle && (17)\\ &= \begin{pmatrix} A' & \cdot \\ \cdot & \cdot \end{pmatrix}\begin{pmatrix} \vec{x} \\ 0 \end{pmatrix} && (18)\\ &= |0\rangle^q A'\,|\vec{x}\rangle + \left|0^{\perp}\right\rangle, && (19)\end{aligned}$$

where $\left|0^{\perp}\right\rangle$ is un-normalized. The probability of measuring the first q qubits in the state $|0\rangle^q$ is $P(|0\rangle^q) = \|A'\,|\vec{x}\rangle\|^2$.

1) Using amplitude estimation, we can obtain an estimate h such that $|h - \|A'\,|\vec{x}\rangle\|| \leq \epsilon_1$ with high probability in time $O((T_A + T_X)1/\epsilon_1)$. From the reverse triangular inequality

$$\left| \|A'\,|\vec{x}\rangle\| - \left\|\frac{A}{\alpha}\,|\vec{x}\rangle\right\| \right| \leq \left\| A'\,|\vec{x}\rangle - \frac{A}{\alpha}\,|\vec{x}\rangle \right\| \leq \frac{\epsilon_0}{\alpha}, \tag{20}$$

which implies $\left|h - \left\|\frac{A}{\alpha}\,|\vec{x}\rangle\right\|\right| \leq \epsilon_1 + \frac{\epsilon_0}{\alpha}$. We set the output to $\bar{t} = \alpha h$, obtaining $\left|\bar{t} - \frac{\|A\vec{x}\|}{\|\vec{x}\|}\right| \leq \alpha\epsilon_1 + \epsilon_0$. Since $\epsilon_0 \leq \frac{\epsilon}{c}$, for some constant c, we can choose $\epsilon_1 = \frac{\epsilon}{q\alpha}$, for a suitable constant q, to obtain $\left|\bar{t} - \frac{\|A\vec{x}\|}{\|\vec{x}\|}\right| \leq \epsilon$ with high probability in time $O\left((T_A + T_X)\frac{\alpha}{\epsilon}\right)$.

2) Using the algorithm above as an oracle, we can build an algorithm that outputs an ϵ-multiplicative approximation of $\frac{\|A\vec{x}\|}{\|\vec{x}\|}$ in expected time $\widetilde{O}\left((T_A + T_X)\frac{\alpha}{\epsilon}\frac{\|\vec{x}\|}{\|A\vec{x}\|}\right)$. The routine and its proof are explained in Chowdhury et al. [8, Appendix D]. Note that while the algorithm is extremely likely to terminate in the expected time, the worst case of this algorithm is not bounded, and in unlucky cases, the routine runs forever.

3) Let $\gamma \leq \frac{\|A\vec{x}\|}{\|\vec{x}\|}$ be a lower bound. The algorithm consists in running amplitude amplification instead of estimation. The proof is the same as in Chakraborty et al. [6, Lemma 4, arxiv version]. However, using a fixed point version of amplitude amplification [32], we can bound the worst case time complexity of the algorithm, rather than the expected. The algorithm runs in time $\widetilde{O}\left((T_A + T_X)\frac{\alpha}{\gamma}\right)$ and requires $\epsilon_0 \leq \frac{\epsilon\gamma}{2}$.

4) The proof is similar, but without a lower bound γ. Using the QSearch version of amplitude amplification [4, Theorem 3], we succeed in expected time $\widetilde{O}\left((T_A + T_X)\alpha\frac{\|\vec{x}\|}{\|A\vec{x}\|}\right)$ if $\frac{\|A\vec{x}\|}{\|\vec{x}\|} \neq 0$ and otherwise execute forever. $\square$

2) Corollary IV.7:

Proof. The result follows by creating a block encoding of A from a quantum data structure and applying Theorem IV.6. It is convenient to think of A as a square matrix with size some power of 2, padded with zeroes if necessary. Similarly, x can be thought as a vector with size the same power of 2, padded with zeroes if necessary. Using Lemma IV.5 we can provide a block encoding of $\overline{A}$, rather than A. However, this is not a problem for our applications.

From the definition of block-encoding, we have

$$\left\| \overline{A} - \alpha(\langle 0|^{\otimes q} \otimes \mathbb{I})U_A(|0\rangle^{\otimes q} \otimes \mathbb{I})\right\| \leq \epsilon_0, \tag{21}$$

Let us define $\overline{A}' = (\langle 0|^{\otimes q} \otimes \mathbb{I})U_A(|0\rangle^{\otimes q} \otimes \mathbb{I})$ as the matrix on the top-left corner of U_A, such that $\left\|\overline{A} - \alpha\overline{A}'\right\| \leq \epsilon_0$. We have $\overline{A} = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}$ and $\overline{A}' = \begin{bmatrix} P & Q \\ R & S \end{bmatrix}$ for some matrices P, Q, R, S of same size as A. We can show that having a block-encoding of $\overline{A}$ is somehow equivalent to having a block encoding of A, in terms of α and ϵ_0. Let us define two projectors Π_1, Π_2 such that $\Pi_1\overline{A}\Pi_2 = A$ and $\Pi_1\overline{A}'\Pi_2 = Q$. We have

$$\begin{aligned} \|A - \alpha Q\| &= \left\|\Pi_1\overline{A}\Pi_2 - \Pi_1\alpha\overline{A}'\Pi_2\right\| && (22)\\ &= \left\|\Pi_1(\overline{A} - \alpha\overline{A}')\Pi_2\right\| \leq \epsilon_0. && (23)\end{aligned}$$

The next step is to apply the block encoding to the state, analogously to Eq. 19. The proofs proceed as in Theorem IV.6.

$$\begin{aligned} U_A(I^{\otimes a} \otimes X \otimes U_x)\,|0\rangle &= U_A\,|0\cdots 01\rangle\,|\vec{x}\rangle && (24)\\ &= \begin{pmatrix} P & Q & \cdot \\ R & S & \cdot \\ \cdot & \cdot & \cdot \end{pmatrix}\begin{pmatrix} 0 \\ \vec{x} \\ 0 \end{pmatrix} = \begin{pmatrix} Q\vec{x} \\ S\vec{x} \\ \cdot \end{pmatrix} && (25)\\ &= |0\rangle^a Q\,|\vec{x}\rangle + \left|0^{\perp}\right\rangle. && (26)\end{aligned}$$

The last remark is that assuming quantum access implies $\alpha = \mu(A)$, which motivates the formulation in the main text. $\square$

References

[1] Ryan Babbush, Jarrod R. McClean, Michael Newman, Craig Gidney, Sergio Boixo, and Hartmut Neven. Focus beyond quadratic speedups for error-corrected quantum advantage. *PRX Quantum*, 2:010103, Mar 2021. doi: 10.1103/PRXQuantum.2.010103. URL https://link.aps.org/doi/10.1103/PRXQuantum.2.010103.

[2] Armando Bellante and Stefano Zanero. Quantum matching pursuit: A quantum algorithm for sparse representations. *Physical Review A*, 105(2):022414, 2022.

[3] Armando Bellante, Alessandro Luongo, and Stefano Zanero. Quantum algorithms for svd-based data representation and analysis. *Quantum Mach. Intell.*, 4(2): 1–23, 2022. doi: 10.1007/s42484-022-00076-y. URL https://doi.org/10.1007/s42484-022-00076-y.

[4] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.

[5] Daan Camps and Roel Van Beeumen. FABLE: fast approximate quantum circuits for block-encodings. In *IEEE International Conference on Quantum Computing and Engineering, QCE 2022, Broomfield, CO, USA, September 18-23, 2022*, pages 104–113. IEEE, 2022. doi: 10.1109/QCE53715.2022.00029. URL https://doi.org/10.1109/QCE53715.2022.00029.

[6] Shantanav Chakraborty, András Gilyén, and Stacey Jeffery. The power of block-encoded matrix powers: Improved regression techniques via faster hamiltonian simulation. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 33:1–33:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi: 10.4230/LIPIcs.ICALP.2019.33. URL https://doi.org/10.4230/LIPIcs.ICALP.2019.33.

[7] Yanlin Chen and Ronald de Wolf. Quantum algorithms and lower bounds for linear regression with norm constraints, 2022.

[8] Anirban N. Chowdhury, Rolando D. Somma, and Yi ğit Subaş ı. Computing partition functions in the one-clean-qubit model. *Phys. Rev. A*, 103:032422, Mar 2021. doi: 10.1103/PhysRevA.103.032422. URL https://link.aps.org/doi/10.1103/PhysRevA.103.032422.

[9] Christoph Durr and Peter Hoyer. A quantum algorithm for finding the minimum, 1999.

[10] Christoph Dürr, Mark Heiligman, Peter Høyer, and Mehdi Mhalla. Quantum query complexity of some graph problems. *SIAM J. Comput.*, 35(6):1310–1328, 2006. doi: 10.1137/050644719. URL https://doi.org/10.1137/050644719.

[11] Athinodoros S. Georghiades, Peter N. Belhumeur, and David J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23 (6):643–660, 2001. doi: 10.1109/34.927464. URL https://doi.org/10.1109/34.927464.

[12] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Architectures for a quantum random access memory. *Phys. Rev. A*, 78:052310, Nov 2008. doi: 10.1103/PhysRevA.78.052310. URL https://link.aps.org/doi/10.1103/PhysRevA.78.052310.

[13] Connor T. Hann, Gideon Lee, S.M. Girvin, and Liang Jiang. Resilience of quantum random access memory to generic noise. *PRX Quantum*, 2:020311, Apr 2021. doi: 10.1103/PRXQuantum.2.020311. URL https://link.aps.org/doi/10.1103/PRXQuantum.2.020311.

[14] Sonika Johri, Shantanu Debnath, Avinash Mocherla, Alexandros SINGK, Anupam Prakash, Jungsang Kim, and Iordanis Kerenidis. Nearest centroid classification on a trapped ion quantum computer. *npj Quantum Information*, 7(1):122, Aug 2021. ISSN 2056-6387. doi: 10.1038/s41534-021-00456-5. URL https://doi.org/10.1038/s41534-021-00456-5.

[15] Iordanis Kerenidis and Anupam Prakash. Quantum recommendation systems. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, volume 67 of *LIPIcs*, pages 49:1–49:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi: 10.4230/LIPIcs.ITCS.2017.49. URL https://doi.org/10.4230/LIPIcs.ITCS.2017.49.

[16] Iordanis Kerenidis and Anupam Prakash. Quantum gradient descent for linear systems and least squares. *Phys. Rev. A*, 101:022316, Feb 2020. doi: 10.1103/PhysRevA.101.022316. URL https://link.aps.org/doi/10.1103/PhysRevA.101.022316.

[17] Iordanis Kerenidis, Jonas Landman, Alessandro Luongo, and Anupam Prakash. q-means: A quantum algorithm for unsupervised machine learning. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 4136–4146, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/16026d60ff9b54410b3435b403afd226-Abstract.html.

[18] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791. URL https://doi.org/10.1109/5.726791.

[19] Kuang-Chih Lee, Jeffrey Ho, and David J. Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27 (5):684–698, 2005. doi: 10.1109/TPAMI.2005.92. URL https://doi.org/10.1109/TPAMI.2005.92.

[20] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum algorithms for supervised and unsupervised machine learning, 2013.

[21] Kohei Miyamoto, Masakazu Iwamura, and Koichi Kise. A quantum algorithm for finding k-minima, 2019.

[22] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Phys. Rev. Lett.*, 113:130503, Sep 2014. doi: 10.1103/PhysRevLett.113.130503. URL https://link.aps.org/doi/10.1103/PhysRevLett.113.130503.

[23] Yue Ruan, Xiling Xue, Heng Liu, Jianing Tan, and Xi Li. Quantum algorithm for k-nearest neighbors classification based on the metric of hamming distance. *International Journal of Theoretical Physics*, 56(11): 3496–3507, Nov 2017. ISSN 1572-9575. doi: 10.1007/s10773-017-3514-4. URL https://doi.org/10.1007/s10773-017-3514-4.

[24] Ferdinand Samaria and Andy Harter. Parameterisation of a stochastic model for human face identification. In *Proceedings of Second IEEE Workshop on Applications of Computer Vision, WACV 1994, Sarasota, FL, USA, December 5-7, 1994*, pages 138–142. IEEE, 1994. doi: 10.1109/ACV.1994.341300. URL https://doi.org/10.1109/ACV.1994.341300.

[25] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *J. Opt. Soc. Am. A*, 4(3):519–524, Mar 1987. doi: 10.1364/JOSAA.4.000519. URL https://opg.optica.org/josaa/abstract.cfm?URI=josaa-4-3-519.

[26] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991.

[27] Matthew A. Turk and Alex Pentland. Face recognition using eigenfaces. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 1991, 3-6 June, 1991, Lahaina, Maui, Hawaii, USA*, pages 586–591. IEEE, 1991. doi: 10.1109/CVPR.1991.139758. URL https://doi.org/10.1109/CVPR.1991.139758.

[28] Miel Verkerken, Laurens D'hooge, Tim Wauters, Bruno Volckaert, and Filip De Turck. Unsupervised machine learning techniques for network intrusion detection on modern data. In Romain Laborde, Nadjib Aitsaadi, Solange Ghernaouti, Abdelmalek Benzekri, and Joaquín García-Alfaro, editors, *4th Cyber Security in Networking Conference, CSNet 2020, Lausanne, Switzerland, October 21-23, 2020*, pages 1–8. IEEE, 2020. doi: 10.1109/CSNet50428.2020.9265461. URL https://doi.org/10.1109/CSNet50428.2020.9265461.

[29] Mei Wang and Weihong Deng. Deep face recognition: A survey. *Neurocomputing*, 429:215–244, 2021. doi: 10.1016/j.neucom.2020.10.081. URL https://doi.org/10.1016/j.neucom.2020.10.081.

[30] Nathan Wiebe, Ashish Kapoor, and Krysta M. Svore. Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning. *Quantum Inf. Comput.*, 15(3&4):316–356, 2015. doi: 10.26421/QIC15.3-4-7. URL https://doi.org/10.26421/QIC15.3-4-7.

[31] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[32] Theodore J. Yoder, Guang Hao Low, and Isaac L. Chuang. Fixed-point quantum search with an optimal number of queries. *Phys. Rev. Lett.*, 113:210501, Nov 2014. doi: 10.1103/PhysRevLett.113.210501. URL https://link.aps.org/doi/10.1103/PhysRevLett.113.210501.