Exploring the Use of Quantum Systems to Perform Principal Component Analysis

Conor Bergman, Asher Moldwin, Alexander Brunmayr

May 15, 2019

Abstract

In this paper we investigate a quantum analogue to the eigenfaces machine learning algorithm. We propose a method which performs quantum principal component analysis with an adaptation of the phase estimation circuit. Eigenfaces is a good candidate for a real-world application of such a construction due to its simple structure and its applications in facial recognition and image compression. We reference previous works and provide a summary of the practicality of such an algorithm. This paper assumes the existence of a quantum random access memory and a curated training set of images. We suggest it is possible, in the ideal case, to realize an exponential speedup over the classical algorithm.

1 Introduction

1.1 Motivation

As the first quantum computers have emerged over the past years, with companies such as IBM even offering rudimentary public access to their systems, practical implementations of quantum algorithms have become a hotbed of computer science research. One such area, blooming in parallel with quantum research, is the machine learning sector, which until recently had suffered from a lot of theory without much practical implementation to back it up. The eigenfaces algorithm was first introduced in 1986 by Sirovich and Kirby [6] as a facial recognition algorithm. While the algorithm is simple, it requires a great overhead in computation and memory to store and create the eigenface components.

Taking a step into the algorithm we notice that, like many other machine learning algorithms, eigenfaces is more of a fancy dressing on top of principal component analysis (PCA). We propose to tackle the quantum implementation of principal component anaylsis (qPCA), utilizing the quantum speedup of the Fourier transform to perform qPCA on a quantum system in complexity that goes like $\log(d)$, where d is the dimension of the system.

It is important to note that this implementation is an idealized case, and the limitations of this assumption will be discussed in Section 4 of this paper. We must first assume some access to a quantum RAM (qRAM), which while possible in theory has not yet been physically constructed. We must also define the bounds of the problem. We lose all speedup over the classical algorithm if we try to recover each eigenface component, so we must instead limit ourselves to a well defined training set, such that the first k eigenvectors well cover the basis, and that the eigenvalues of the remaining basis vectors are negligible.

The eigenfaces algorithm is exciting because of its implications. Once we have created our basis set of eigenfaces, we are able to perform image recognition, recreation and compression with very little cost. A weighted linear combination of eigenfaces can be used to replicate entirely foreign images to the dataset.

Suppose we wanted to store one billion images. Instead of storing the raw pixel data every time we want to add a new image to our data set, we can store the weights and components from our prepared basis set that represent it to some threshold, ϵ. In the case of facial recognition, the eigenface algorithm allows us to identify the most significantly contributing image from the training set to a test image - the face from the training data that is most similar to the test.

It is necessary for good results that the training data well samples the desired test-space from which the algorithm seeks to replicate.

1.2 Quantum Machine Learning

Machine learning involves the use of computer algorithms to learn useful patterns in a dataset. Machine learning algorithms can be divided into "super-

vised" and "unsupervised" varieties. Supervised machine learning algorithms use "labeled" data, where every data point has a classification associated with it, and seek to converge on a model which will correctly classify unlabeled data. Unsupervised learning aims to find patterns in unlabeled data by grouping similar data points.

The discovery of fast quantum algorithms for many basic linear algebra operations has led to proposed ways to improve several machine learning algorithms. Both of the supervised and unsupervised varieties of machine learning often rely heavily on matrix operations on large sets of data, making them both areas of interest for researchers seeking to "quantize" machine learning algorithms. The operations that have been theorized to undergo an exponential speedup when implemented using quantum circuits include Fast Fourier Transforms, eigenvalue estimation, and matrix inversion. Generally, these speedups arise from the fact that the data takes exponentially fewer bits to encode in the quantum implementation: Storing a vector containing data of dimension d in the quantum representation only takes $O(\log_2 d)$ qubits compared with $O(d)$ bits to store the data classically, where d is 2^n [9].

This means that if we have many vectors of data, each with dimension d, the number of qbits required to store all of the data in quantum superpositions for each vector is $O(\log_2 d)$. The ability to achieve this compression relies on the use of a Quantum Random Access Memory or qRAM, which will be described in Section 2.

1.3 Classical Algorithm

Classically, the eigenfaces algorithm is implemented in 5 steps.

1. **Assemble training set**

The training set is composed of cropped images of faces, all with the same dimensions. In order to get a good model the training set should have a large variety of people, lighting, and poses. One of the largest limitations of the eigenfaces algorithm is the overhead required to create and prepare the training set, such that all the images are centered and cropped. In our implementation we use the Extended Yalefaces Database B [11][15], shown in figure 1, which provides 16,128 images in 168x192 resolution of 28 different subjects, each with 64 lighting conditions and 9 poses.

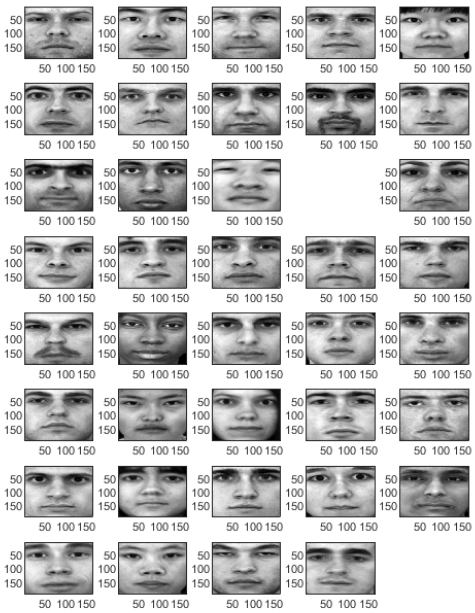


Figure 1: Sample of training images from the Extended Yalefaces Database B. Images are 168x192 pixels in portable grey map (pgm) format, such that each pixel is represented by some one or two byte value between 0 and the maximum greyscale value.

2. **Subtract mean of the training set from each image**

In order to create the covariance matrix, we must first subtract the mean of the training set from each individual image. Note that the images of the Yalefaces dataset are stored in pgm format, so that pixel values are single-valued in greyscale. This makes the mean a simple operation over the pixel space.

3. **Create covariance matrix P**

To create the covariance matrix, we first need to vectorize the images. Abstractly, the covariance matrix tells how each pixel depends on the others. From the covariance matrix we can extract eigenvalues and eigenvectors, which will allow us to form a basis for our eigenfaces model.

4. **Find eigenvalues and eigenvectors of P**

To extract the eigenvalues and eigenvectors of the system classically we use gaussian elimination. This extraction of the eigenvectors from the covariance matrix is principal component analysis. The eigenvalues corresponding to each eigenvector tell how much that vector contributes to the basis as a whole. A sample of the eigenfaces are provided in figure 2.

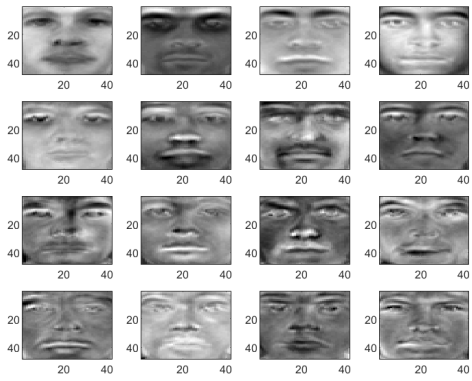


Figure 2: The 16 principal components (eigenvectors corresponding to the most significant eigenvalues) of the model basis set. Note that while each eigenface resembles real faces, none are contained in the training set and are all entirely fictional. We can recreate images by taking a weighted linear combination of some k principal eigenfaces from our model basis.

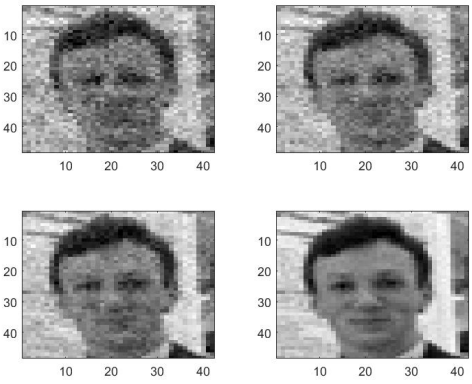


Figure 3: Foreign face recreated using eigenfaces model constructed from training set. Note that each pane corresponds to N = (250, 500, 750, 1000) eigenfaces to recreate the image. It is also important to consider that while the foreign image is a face, it is not well cropped to suit the database, and so performance on face reconstruction is poorer than could be achieved with proper cropping. However, this does display the strength of the algorithm to recreate objects that are not even defined by the training set.

5. **Select k principal eigenvectors as basis set**

We want to only keep the eigenvectors that significantly contribute to a well spanning basis set, which we can determine by their eigenvalues. We only want to keep the eigenvectors with the k largest eigenvalues, such that we can represent images as a weighted linear combination of this basis to some error ϵ. A recreation of the author's headshot is provided for demonstration (3).

1.4 Quantum Algorithm

Our proposed quantum algorithm is very similar to the classical implementation, where we still need to assemble a classical training set, however our principal component analysis is done with a quantum system.

1. **Assemble training set**

Our training set is assembled identically to the classical method.

2. **Load classical data into quantum RAM (qRAM)**

In order to interact with our classical data we need to load it onto a quantum RAM. The architectures for qRAM implementations are discussed further below, however it is important to note that while the proposed architectures are theoretically motivated, they have not been physically realized.

3. **Create density matrix P**

The density matrix is the covariance matrix. In order to perform quantum principal component analysis we will need many copies of P to input to the phase estimation circuit, as explained further later in the paper.

4. **Find eigenvalues and eigenvectors of P**

Instead of gaussian elimination, in the quantum circuit we take advantage of the quantum speedup of the Fourier transform. We can construct an operator of the density matrix and feed the original density matrix through the phase estimation circuit. From this we extract the k largest eigenvalues and corresponding eigenvectors of the system, which form our model basis.

2 Quantum RAM

2.1 Classical to Quantum Data

A quantum RAM allows us to store and access quantum states, such as our image data or our density matrix, while maintaining the quantum nature of the state and its value. If we have some n qubits, each qubit being either $|0\rangle$ or $|1\rangle \in \mathbb{C}^2$ we can represent some 2^n classical bits of data.

$$\left(|0\rangle_0 |1\rangle_1 |1\rangle_2 ... |0\rangle_{n-1}\right) \in \left(\mathbb{C}^2\right)^{\otimes n}$$

And so if we have some classical state of 2^n bits:

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{2^n} \end{bmatrix}$$

We can represent it as a superposition of states i, where i is some n-bit number:

$$|x\rangle = \sum_i x_i |i\rangle$$

Well, how do we load this quantum data onto the qubits? We can consider some idealized case, where using the well known dispersive properties of particle wavefunctions we can represent an entire image with one photon. Seth Lloyd, in his talk to Google on quantum machine learning [8], uses the example of a CD containing some classical data, for example an image. Classically we access this data by shining a laser at a specific position on the CD. If the light is reflected by a mirror back to us we measure that position as a bit 1, if the light is scattered away or absorbed, and so we see no light, we measure that position as a bit 0. We then rotate the CD one position, shoot another photon, and repeat. If we were to replicate this with quantum mechanics, we would use a lens to spread a single photon out over the entire surface area of the CD and what would bounce back to us would be the superposition of being in all the positions where there is a one, and not being in all the positions where there is a zero. We have now mapped a classical source of data to a quantum mechanical representation all the data of the CD, stored in a single photon.

Once we have this state $|x\rangle$ we need to convert it to quantum digital form so that we can interact with it. This is done with quantum RAM. Typical physical implementations of qRAM have qubits as electrons, which are held in quantum memory using ion traps. One suggestion for quantum image representation, explored by Srivastava et al. [7] involved storing the color values of each pixel in pixel-space as an amplitude of its respective component in an $l \times h$ qubit matrix. An image of N^2 pixels can be stored with $O(\log N)$ qubits.

2.2 Bucket Brigade Architecture

One proposed architecture for the implementation of quantum RAM is the bucket-brigade construction, which allows us to access quantum data more efficiently than a typical binary tree structure, using $O(\log N)$ switches for each memory access [4]. The basic bucket brigade design is shown in Figure 4, where the memory address to be accessed is read one bit at a time. At each junction the bit travels left or right depending on the state of the junction. When it reaches a junction at a wait state, it excites it to either a left state if the bit is a 0, or right if the bit is a 1. As shown in Figure 5, the registers are transferred down the carved path by exciting each junction to some $left'$ or $right'$, which decay back into $left$ or $right$ respectively and emit the incident register bit.

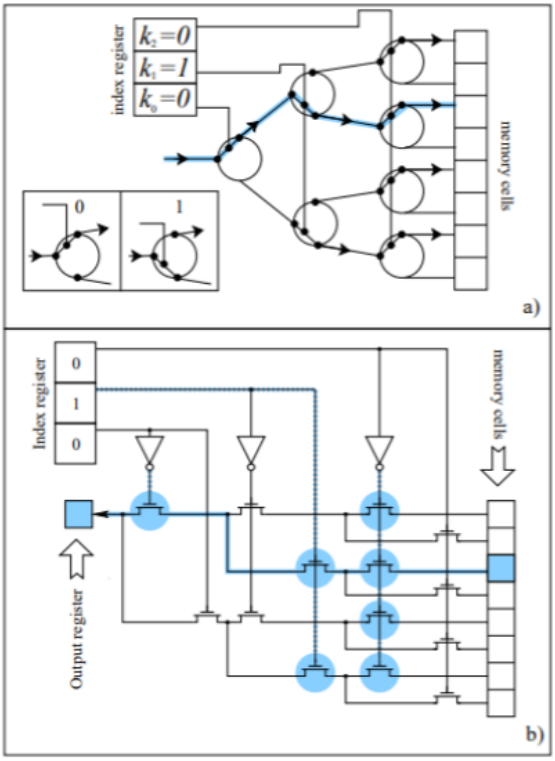


Figure 4: Diagram of Bucket Brigade qRAM architecture[5]. BB-QRAM is very similar to the typical binary tree implementation, but reduces the necessary switches for each memory access from $O(N)$ to $O(log\ N)$, which saves on power and potential error.

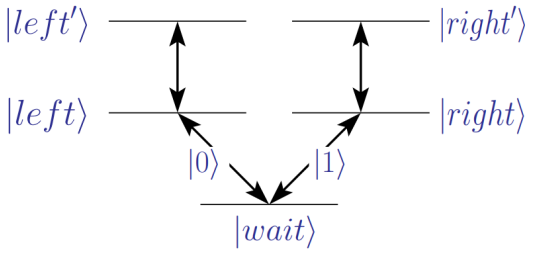


Figure 5: BB-QRAM uses qutrits as junctions, instead of a qubit[4]. These qutrits can take the state left, right, or wait. Wait can be exicted to left or right, which can in turn be excited to left' or right'. The states left' and right' decay back to left and right respectively, and emit the incident bit that excited them. This allows us to carve a path to the memory register.

Once a path has been carved out to the quantum memory register, a bus state is sent down the path which retrieves the quantum state located in memory and resets all the junctions back to the wait state.

This reduction in activated logic gates during each memory call could, in principle, allow for a qRAM implementation without large quantum error correction overhead. The error for each memory call goes like $\log_2 N\epsilon$, where ϵ is the error rate of each switch [5]. A switch error rate of $\epsilon = 0.1\%$ would allow for 2^{100} quantum memory locations with an error of 10%. An effective qRAM must operate with a low enough error rate to be reliable, without slowing down operation or inflating the number of qubits required with error correction.

3 Quantum Implementation of the PCA Algorithm

Our goal is to perform principal component analysis on the covariance matrix of the images in the training set in order to identify the most important eigenvectors to be used later for eigenfaces. In our quantum implementation of PCA, we load multiple copies of the classically computed covariance matrix onto the qRAM.

3.1 Density Matrices and Mixed States

In its qRAM representation the covariance matrix is a density operator ρ on the Hilbert space. As stated in Chapter 3 of KLM[16], a density operator is also a representation of a mixed state or ensemble of states

$$\left\{(|\psi_1\rangle, p_1), (|\psi_2\rangle, p_2), ..., (|\psi_n\rangle, p_k)\right\}$$

If we have this mixed state, then with probability p_i we are in the pure state $|\psi_i\rangle$, $(1 < i < n)$. The mixture of pure states $|\psi_i\rangle$ can be represented more succinctly using the density operator notation

$$\rho = \textstyle\sum_{i=1}^{k} p_i |\psi_i\rangle \langle\psi_i|$$

If we think of ρ as a matrix, the $|\psi_i\rangle$'s are its eigenvectors and the corresponding eigenvalues are p_i.

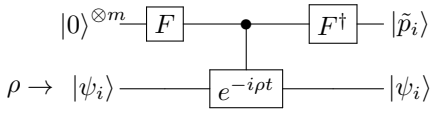
3.2 Matrix Exponentiation

Since the density or covariance matrix is symmetric with respect to its diagonal, it is a Hermitian matrix. However, if we want to use the density operator in a quantum circuit, we need it to be unitary. Fortunately, there is a way to create a unitary operator out of a Hermitian matrix: if $H = H^\dagger$ is Hermitian, then e^{-iH} and e^{iH} are unitary (and the inverse of each other). This identity can be easily verified by doing the Taylor expansions of e^{-iH} and e^{iH} (and noting that $e^{-iH}e^{iH} = I$).

We now want to create the unitary operator $e^{-i\rho}$ to use in a quantum circuit. There are fast algorithms to exponentiate matrices if they are sparse. However, the density matrix is generally not sparse. Lloyd, Mohseni and Rebentrost explain in their 2013 paper that such a matrix exponentiation can still be performed effectively if ρ is low-rank. They show that "multiple copies of the state ρ can be used to implement the unitary operator $e^{-i\rho t}$ " and that exponentiating non-sparse matrices can be done "in time $O(\log d)$, an exponential speed-up over existing algorithms" [1].

3.3 qPCA: Phase Estimation on $e^{-i\rho t}$

The goal of the principal component analysis of the covariance matrix ρ is to find the eigenvectors with the highest associated eigenvalues so we can select the top k eigenvectors of ρ with the largest eigenvalues to use them as our basis set for the eigenfaces. We have access to the eigenvectors of ρ simply by sampling multiple copies of ρ and then being in the pure state ψ_i with probability p_i. Finding each eigenvector's eigenvalue, however, requires a more elaborate algorithm. Lloyd et al. propose a quantum circuit that performs a phase estimation of the unitary operator $e^{-i\rho t}$ with the mixed state ρ as input.



This is an adapted version of the phase estimation circuit as represented in the KLM textbook. Here, F is the quantum Fourier transform (QFT) and $F^\dagger$ is the inverse quantum Fourier transform (QFT^{-1}).

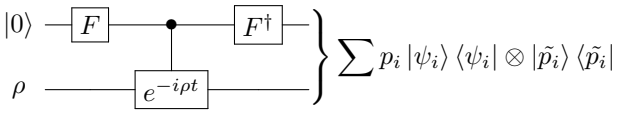
Since $e^{-i\rho t}|\psi_i\rangle = e^{-ip_i t}|\psi_i\rangle$, the output of this circuit is the input eigenstate $|\psi_i\rangle$ itself and the state $|\tilde{p_i}\rangle$ containing the estimate of the eigenvalue p_i. We can then measure $|\tilde{p_i}\rangle$ in the computational basis to have an approximate value for the number $2^m p_i/2\pi$ in binary form, where m is the number of ancillary qubits used to perform the phase estimation.

If we now consider the mixed state ρ to be the subject of the unitary operation $e^{-i\rho t}$, we can write

$$e^{-i\rho t}\rho = e^{-i\rho t}\sum_{i=1}^{k} p_i \left|\psi_i\right\rangle \left\langle\psi_i\right| = \sum_{i=1}^{k} p_i e^{-ip_i t} \left|\psi_i\right\rangle \left\langle\psi_i\right|$$

Therefore, the qPCA algorithm can be written simply as

$$\text{qPCA: } \rho \left|0\right\rangle^{\otimes m} \mapsto \sum_{i=1}^{k} p_i \left|\psi_i\right\rangle \left\langle\psi_i\right| \otimes \left|\tilde{p_i}\right\rangle \left\langle\tilde{p_i}\right|$$



If we measure the state $\left|\tilde{p_i}\right\rangle$ (the upper register in the circuit above) in the computational basis, we know the eigenvalue to the eigenstate in the lower register. Running this circuit multiple times, we can find the highest-probability eigenpairs and select the top k eigenvectors with the largest eigenvalues to form our basis for our eigenfaces model.

We can then measure the few selected eigenstates to load them on a classical memory. Accurately measuring these eigenstates is expensive, but since we only need to measure k of them, this method of principal component analysis still seems to be worthwhile.

4 Considerations of Efficiency

One constraint of using the eigenvalue estimation algorithm is that in order to extract a large number of eigenvalues with high confidence, many copies of the density matrix ρ will be needed. First, it will take m copies of rho to create the matrix exponential $e^{-i\rho}$ which is used in the eigenvalue estimation circuit. This matrix exponential then needs to be reconstructed n times for each time the eigenvalue estimation circuit is run, meaning we need nm copies of ρ just to find the amplitudes of the principal components in our data. The probability of seeing a given eigenstate ψ after running the eigenvalue estimation circuit n times is $p_i n$, meaning that we should see the very largest principal components with high probability after relatively few iterations but would have to run it many times to have a high probability of seeing the smaller-amplitude states.

4.1 Analysis of Complexity

The general motivation for theorizing that the quantum PCA algorithm can provide an exponential speedup comes from the fact that it employs the quantum Fourier transform in the eigenvalue estimation circuit. In their paper Lloyd et al. [1]concluded that the quantum implementation of PCA has complexity which is polynomial in the logarithm of the dimension of the data, while the classical PCA algorithm would take time polynomial in the dimension of the data – seemingly signifying an exponential advantage for the quantum version.

4.2 Caveats and Practical Constraints

In a 2015 article, Aaronson [2] listed several reasons for skepticism when weighing claims of exponential speedups in quantum machine-learning algorithms, citing reasons to believe that some theoretical speedups would only occur in limited cases and would not necessarily lead to a more efficient algorithm when used for a real-world application. These caveats included a possible storage constraint which says that fast storage of data into qRAM might only be possible with relatively uniform data, i.e. we would need our data not to contain some values which are much larger than the rest. Another possible constraint, which affects the HHL algorithm[17] for solving systems of equations, would require that the data must have a condition number which does not increase polynomially with the size of the data. Aaronson notes that once all of the constraints are considered, it is possible that a once-promising quantum machine learning algorithm might be reduced to working only in simple cases where a classical algorithm would also have performed well.

We summarize the primary constraints as follows:

1. **There must exist a quantum RAM**

While we have introduced an architecture for qRAM, and many other variations exist, the scope of creating a useful qRAM is not known. It may not be physically possible or practical to store large amounts quantum data in an electronically accessible format while maintaining both correctness and speed.

2. **The data must be "nice"**

The HHL algorithm assumes that our density matrix ρ is robustly invertible, such that the values of our matrix are nearly uniform. The time complexity of the HHL algorithm goes linearly with some factor χ, where $\chi = |\rho_{max}/\rho_{min}|$[2]. Because we take our input to be some pixel matrix, we can assume our

data will be well behaved, and as such χ will grow exponentially.

3. **We cannot recover exactly all eigenvectors**

Even if we are able to satisfy the first two constraints sufficiently to receive an exponential speedup over the classical algorithm, we are left with eigenvalues and eigenvectors stored as quantum states. We need to classically recover the data, and classically recovering data is expensive. In order to recover a state of n qubits using HHL we must re-run the algorithm $N = n^2$ times.

Wiebe et. al. provide an alternative method for state approximation using quantum least-squares fit [12]. This algorithm allows us to estimate a state in $\approx O(\log N \frac{1}{\epsilon^3})$, where ϵ is the fault tolerance. Because we only need to measure the k principal components, and we assume a training set such that the eigenvalues decay sufficiently fast, we are still able to achieve an exponential speedup over the classical algorithm.

We can also consider the case of "zero-one" problems, such as classification, where we don't need to know exactly any of the states, just one bit of information. If we are able to prepare some state $|cat\rangle$, which corresponds to images showing a cat, and its opposite state $|dog\rangle$, we can measure some test image in the $\{|cat\rangle, |dog\rangle\}$ basis without having to directly measure and write out the full $|cat\rangle$ or $|dog\rangle$ state.

4.3 Further Areas of Study

It will be difficult to know with certainty whether the quantum PCA algorithm described by Lloyd et al. will be useful until it can be tested on a real-life quantum computer with access to a robust qRAM. It is possible that the constraints on its usefulness will also lead to modified or more restricted forms of the PCA algorithm. For example, the problem might be sufficiently simplified in the case of image classification if there are only two classes, obviating the need to find all the smaller-amplitude eigenpairs. It is also possible that this algorithm will prove to be useful when used in conjunction with other quantum algorithms: For example, for problems that would require extracting the basis vectors and not just an estimate for the eigenvalues (such as image compression), it is possible that we could use the quantum least-squares method described by Wiebe et al. in their 2012 paper on quantum data fitting [12].

Another concern mentioned by Aaronson, is that some quantum algorithms may only seem to have a speedup because the classical algorithms that they are compared with are not truly analogous, and that by creating the quantum version of the algorithm we might be simplifying the problem in some way that would affect the efficiency of our algorithm. This is the topic of several papers by Tang [13][14], who sought to formulate a method to "dequantize" quantum algorithms in a way which would ensure that the problem being solved by the quantum algorithm is the same as the one solved by the classical one.

One of these papers describes classical algorithms for PCA and clustering which were inspired by the quantum algorithms [14]. Using these classical analogues of quantum algorithms can yield insight as to whether using real quantum algorithms can significantly improve efficiency over the analogous classical implementation. If the classical version of the same algorithm proves to have relatively similar efficiency to the real quantum algorithm, this indicates that the problem being considered is not improved exponentially by using the quantum algorithm.

To create the comparable classical version of a quantum algorithm, Tang posits that the sampling conditions in the classical algorithm must match the state-preparation conditions of the quantum algorithm being emulated. This condition is detailed in Tang's earlier paper focusing on dequantizing another quantum machine learning algorithm, the quantum algorithm for recommendation systems. Tang notes that algorithms that are often easiest to dequantize are those that do not fall into the category of BQP-complete algorithms, and algorithms outside this category often do not perform significantly better when using the quantum version. She then references the quantum PCA algorithm proposed by Lloyd et al. and proposes its classical equivalent and suggests that no exponential speedup is achieved by using the quantum version.

5 Conclusion

Instead of focusing solely on the theoretical details of existing quantum algorithms, we chose to look at a specific case of a powerful classical algorithm in the field of image processing, and explore how it might be improved if it could use a quantum method from our class textbook. The reason for choosing this was to combine the basic theory that we learned with a more complicated, real-world algorithm and gain an understanding of how the advent of quantum computing might contribute not just to solving small standalone math problems but could even be useful in solving problems in image processing, a field which is, on the surface, as far removed as it gets from the oft-cited application of quantum algorithms for breaking codes

and quantum cryptography.

One issue which will need to be addressed when the quantum version of Eigenfaces is implemented, is whether the entire algorithm will be delegated to the quantum computer or just some specific subroutines. From our exploration of this topic, it seems that the eigenvalue estimation is the most important part of the algorithm to execute on a quantum computer. However, as we mentioned, other quantum algorithms such as the least squares fitting method might be needed to extract useful data from this. In addition, some methods have been proposed for storing entire images in qRAM [7], possibly meaning that even more of the algorithm could be computed in the quantum system.

References

[1] S. Lloyd, M. Mohseni, P. Rebentrost, *Quantum Principal Component Analysis*, 2013.

[2] S. Aaronson, *Read the Fine Print*, 2015.

[3] D. Park, F. Petruccione, J.K. Rhee, *Circuit-Based Quantum Random Access Memory for Classical Data*, 2019.

[4] V. Giovannetti, S. Lloyd, L. Maccone, *Quantum random access memory*, 2008.

[5] V. Giovannetti, S. Lloyd, L. Maccone, *Architectures for a quantum random access memory*, 2008.

[6] L. Sirovich, M. Kirby, *Low-dimensional procedure for the characterization of human faces*, 1986.

[7] M. Srivastava, S. Roy-Moulick, P. Panigrahi, *Quantum Image Representation Through Two-Dimensional Quantum States and Normalized Amplitude*, 2015.

[8] S. Lloyd, Google Quantum AI Lab Tech Talk, *Quantum Machine Learning*, 2014.

[9] S. Lloyd, M. Mohseni, P. Rebentrost, *Quantum algorithms for supervised and unsupervised machine learning*, 2013.

[10] Extended Yalefaces Database B.

[11] Kuang-Chih Lee, Jeffrey Ho, David Kriegman, *Acquiring Linear Subspaces for Face Recognition under Variable Lighting*, 2005.

[12] N. Wiebe, D. Braun, S. Lloyd, *Quantum Data-Fitting*, 2012.

[13] E. Tang, *A quantum-inspired classical algorithm for reccomendation systems*, 2019.

[14] E. Tang, *Quantum-inspired classical algorithms for principal component analysis and supervised clustering*, 2018.

[15] A. Georghiades, P. Belhumeur, D. Kriegman, *From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose*, 2001.

[16] P. Kaye, R. LaFlamme, M. Mosca, *An Introduction to Quantum Computing*, 2007.

[17] A. Harrow, A. Hassidim, S. Lloyd, *Quantum algorithm for solving linear systems of equations*, 2009

A Contributions

Alexander Brunmayr: Helped other group members with technical understanding of how the eigenvalue estimation would be used with the density matrix exponential and collaborated on writing the report.

Asher Moldwin: Investigated complexity considerations and general discussion of quantum machine learning possibilities and caveats, collaborating with teammates to explore whether these constraints might prevent qPCA from being useful.

Conor Bergman: Introduced team to topic of PCA and eigenfaces. Explained concept of qRAM to teammates and how it is essential for implementing qPCA. Classical implementation of algorithm.

B Code and Data

The source code for classical implementation of the eigenfaces algorithm, which we used with minor adaptations, can be found at https://en.wikipedia.org/wiki/Eigenface.

Data for the training set was taken from the Extended Yalefaces Database B

C Acknowledgments

The authors would like to thank Professor Childs for his helpful discussions and feedback.
We also wish to acknowledge the creators of the Yale Face Database B, which was used in this paper for demonstration.
Note Figures 4 and 5 are from Giovannetti et. al[5][4].