Singular Value Decomposition, Compressed Sensing and the K-SVD Algorithm in Machine Learning

Aaron Shannon[1] and Dr. Nicholas Kevlahan[2]

[1,2]Department of Mathematics, McMaster University, Main Street, Hamilton, L8S 4L8, ON, Canada.

Abstract

In the realm of machine learning and neural networks, the use of Singular Value Decomposition (SVD), Compressed Sensing (CS), and the K-SVD algorithm has created a powerful approach for data representation, feature extraction, and model optimization. SVD plays a pivotal role in decomposing high-dimensional data matrices into their constituent singular values and vectors, enabling dimensionality reduction. Compressed Sensing complements this by enabling sparse signal recovery from undersampled measurements, leveraging the intrinsic sparsity of many real-world datasets. The K-SVD algorithm further refines this framework by iteratively updating dictionaries to enhance signal representation and reconstruction accuracy. Through a literature review the theoretical foundations and practical implementations of these methodologies within the context of machine learning and neural networks will be highlighted. The paper will discuss their applications in neural network optimization, image processing and reconstruction based on both a limited number of samples and an over complete number of samples, showcasing their efficacy in enhancing model robustness and computational efficiency. Finally, this paper presents experimental results that demonstrate the effectiveness of SVD, CS, and K-SVD in various real-world scenarios, and will discuss limitations and areas of future research for extensions of this paper.

Keywords: Compressed Sensing, Singular Value decomposition, K-SVD, Machine Learning, Neural Networks

1 Introduction

With the size of computer and sensor data becoming increasingly larger, to the current size of petabytes [1], the amount of time to read, store and calculate useful information from this data follows the same trend. Even with Moore's Law suggesting that the computational power of new computers doubles every year, a more effective approach to handling big data would be to use efficient mathematical algorithms. Compressed Sensing (CS) is the technique of specifically picking times to collect data from an incoming signal in order to avoid analyzing less important information. Likewise, the Singular Value Decomposition (SVD) algorithm finds the most important information from a matrix to approximate the original data, by taking up less storage space. K-Singular Value Decomposition (K-SVD) takes a large dictionary of data and reconstructs an input based on the smallest number of dictionary elements that still provide a good solution. The goal of this research paper is to find a link between the three topics of Singular Value Decomposition, Compressed Sensing and K-SVD their use within Machine Learning and Neural Networks.

Machine Learning, similar to human learning, tries to create a program to complete a specific task, and update its answers based on new information. An Artificial Neural Network, similar to a human brain's Neural Network and the basis of Machine Learning, takes input data, processes it through interconnected layers of artificial neurons, and produces an output based on the patterns and relationships it learns from the input data during training. The goal is to use these networks to perform tasks such as classification, regression, pattern recognition, and more, by learning from examples and adjusting internal parameters, called weights and biases, to improve performance. The process of creating the networks and applying them to big data sets is significantly faster than what a person or group of people could handle. This is the reason that Machine Learning is significant in the future of computer application.

After exploring the theoretical mathematics behind each algorithm, they will be implemented into a MATLAB or Python code to show its direct relation to Neural Networks, or as the goal of the network to classify or reconstruct training and test data. The programs that will be used to demonstrate the applications of SVD, CS and K-SVD can be found in this GitHub repository.

2 Singular Value Decomposition (SVD)

2.1 Singular Value Decomposition of Square Matrices

Modern computer software stores this large amount of data in vectors and matrices. A large part of the study of numerical linear algebra deals in finding easier ways to represent matrices so that they can be more easily manipulated. One efficient algorithm is the Singular Value Decomposition (SVD) of a matrix [2]. The Singular Value

Decomposition of an m by n matrix A is

$$A = U\Sigma V^T \tag{1}$$

where the matrices on the right hand side are

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 & ... & 0 \\ 0 & \sigma_2 & 0 & ... & 0 \\ 0 & 0 & \sigma_3 & ... & 0 \\ \vdots & \vdots & \vdots & \ddots & \sigma_{min(m,n)} \end{bmatrix}$$

where each σ is a singular value which are the non-zero eigenvalues of the matrix A^TA. Σ has the property that it is diagonal, . The matrix

$$U = \begin{bmatrix} U_1 \ U_2 \ ... \ U_{m-n} \end{bmatrix}$$

where each vector U_i is an eigenvector of the matrix AA^T. They are also known as the left singular values of the matrix A. U has the property that it is unitary, meaning the the transpose of the matrix is equal to its inverse. Finally, the matrix

$$V^T = \begin{bmatrix} V_1 \ V_2 \ ... \ V_{n-m} \end{bmatrix}$$

where each vector V_i is an eigenvector of the matrix A^TA. These are also known as the right singular values of A. The matrix V is also unitary.

For a standard linear system

$$Ax = b$$

It is possible that the inverse of the matrix A does not exist, therefore another way to solve the system is by first left multiplying by the Transpose.

$$A^TAx = A^Tb$$

The solution to the linear equation is then

$$x = (A^TA)^{-1}A^Tb$$

In order to simplify the notation, the substitution

$$A^+ = (A^TA)^{-1}A^T$$

is made. This is known as the Moore-Penrose psuedo-inverse. While the pseudo-inverse appears to have more computational cost as there are multiple matrix multiplications, the properties of the SVD cause this relation to simplify further into

$$A^+ = V\Sigma^+U^T$$

This new matrix Σ^+ is represented as

$$\Sigma^+ = \begin{bmatrix} \frac{1}{\sigma_1} & 0 & 0 & ... & 0 \\ 0 & \frac{1}{\sigma_2} & 0 & ... & 0 \\ 0 & 0 & \frac{1}{\sigma_3} & ... & 0 \\ \vdots & \vdots & \vdots & \ddots & \frac{1}{\sigma_{min(m,n)}} \end{bmatrix}$$

When the original σ value is considered small, or in the language of numerical methods, smaller than the machine precision in a specific coding language, the new value is turned into zero.

2.2 Singular Value Decomposition of Under Determined Matrices

There is an alternative method for when the matrix A is rank deficient, or there are more unknowns than equations to solve. This changes the SVD equation to

$$A = \begin{bmatrix} U_1 \; 1_2 \; ... \; U_{m-n} \end{bmatrix} \begin{bmatrix} \Sigma_1 \\ 0 \end{bmatrix} \begin{bmatrix} V^T \end{bmatrix}$$

While the properties of the matrices remain; U and V unitary and Σ diagonal, the extra zero vector that occurs below Σ is a result of the lack of eigenvalues for those dimensions. While an exact solution cannot be found, a least squared solution to the linear system with this matrix can still be approximated by

$$Ax = b$$

By a similar logic to the square case, it is possible that the inverse cannot be directly calculated and so the system will first be right multiplied by the transpose. This results in the solution to the least square solution being

$$x = A^T(AA^T)^{-1}b$$

or using the pseudo-inverse

$$A^+ = (AA^T)^{-1}$$

the solution becomes

$$x = A^+b$$

Since the solution is an approximate to the actual solution, the accuracy of the approximation can be changed through the best rank k approximation. The best rank k approximation is when only k of the total number of singular values are included in the solution. From equation (1)

$$A = U\Sigma V^T = \sum_{i=1}^{r} = \sigma_1 U_1 V_1^T + \sigma_2 U_2 V_2^T + ... + \sigma_r U_r V_r^T$$

From the property that the singular values are ordered from largest to smallest, most of the contribution to the sum comes from the first values. By leaving out the smaller singular values of the sum the best rank k approximation is

$$A_k = \sum_{i=1}^{k\leq r} = \sigma_1 U_1 V_1^T + \sigma_2 U_2 V_2^T + ... + \sigma_k U_k V_k^T$$

2.2.1 Computational Complexity

In numerical linear algebra the computational complexity measures how many calculations a method or algorithm requires to complete or the amount of data that needs to be stored in a computer. For the SVD algorithm, the size of the matrix A is $m \times n$ giving the total number of values needed to be stored mn.

The purpose of the best rank k approximation is for the application of size reduction. By leaving out small values, the total number of values needed to be stored decreases. The size of U reduces to $m \times k$, V to $k \times n$ and Σ to $k \times k$. This causes the total number of values needed to be stored $k(m + n + 1)$.

2.2.2 Size Reduction

When comparing the size of the number of values of both the complete matrix and rank k matrix, the rank k approximation needs significantly less values for small values of k. An expression can be made for the total compression size between the methods.

$$Total\ Compression\ Size\ = \frac{mn}{k(m+n+1)}$$

A modern camera such as one found in a modern smartphone can have enough photo-receptors to take around a 50 Megapixel image. This equates to the size of the image matrix on the order of $m = n = 10^3$.

2.2.3 Condition Numbers

In order to determine if a matrix is well-behaved enough to run computer code and simulations, is through a condition number. For a matrix the condition number is defined as

$$cond(A) = \frac{\sigma_{max}}{\sigma_{min}}$$

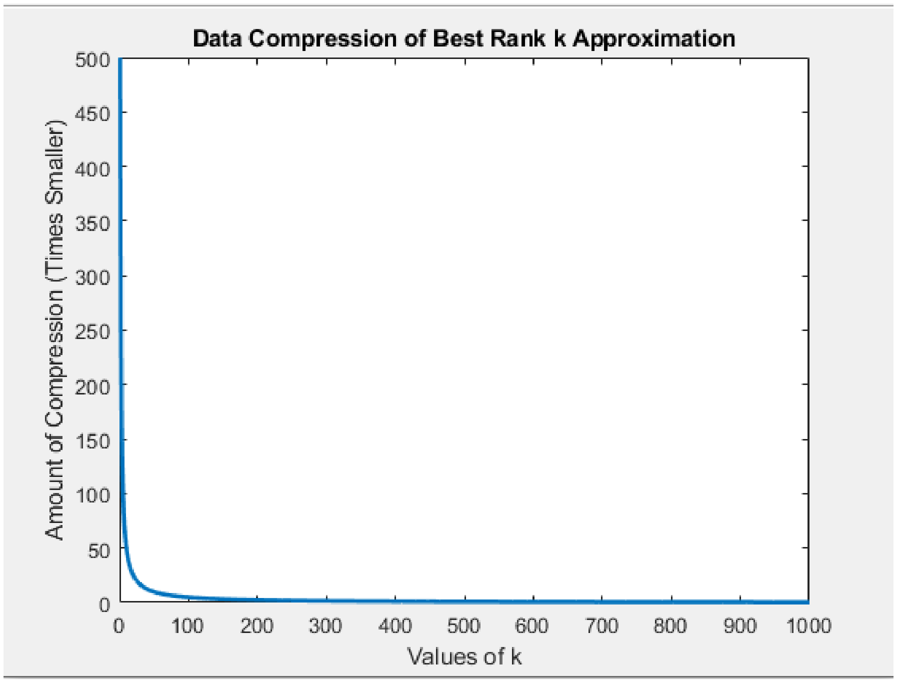


Fig. 1 Shows the relationship between the values of k in the best rank k approximation and the how many times smaller the compression is for that value.

When using a Best Rank k Approximation, the condition number changes since the smallest eigenvalues are removed.

$$cond(A) = \frac{\sigma_{max}}{\sigma_k}$$

3 Compressed Sensing (CS)

3.1 Norms

In the study of numerical mathematics, a norm takes a number, vector or matrix and transforms it into a positive real number as an output. The physical meaning of this value changes based on the type of norm that is being used. The general case is the p-norm, l_p, where p is any real integer larger than 3. For any value of p, the formula is defined as

$$||x||_p = (\Sigma_{i=1}^{n} x_i^p)^p$$

Some of the more interesting and applicable norms for the cases of compressed sensing are the zero, one and two norm.

Starting with l_0, and a vector x, the output is simply the number of non-zero elements of x. For example, taking a row vector $x = [0, 1, 0, 2, 0, 3]$, the zero norm would be

$$||x||_0 = 3$$

Now looking at l_1 and a vector x, the output is the sum of all the elements of x. This norm is also called the taxi-cab or Manhattan norm. For example, taking the same row vector $x = [0, 1, 0, 2, 0, 3]$, the one norm would be

$$||x||_1 = 0 + 1 + 0 + 2 + 0 + 3$$

$$||x||_1 = 6$$

Finally, taking l_2 and a vector x, the output is the sum of the squares of all the elements of x. This norm is also known as the Euclidean norm, and in the special case of x having only two elements is equivalent to the Pythagorean Theorem. For example, taking the same row vector $x = [0, 1, 0, 2, 0, 3]$, the two norm would be

$$||x||_2 = \sqrt{0^2 + 1^2 + 0^2 + 2^2 + 0^2 + 3^2}$$

$$||x||_2 = 14$$

While all of these methods seem simple to calculate for a human by hand, on large scales thee norms become much harder to compute. Efficient computation of the l_0 norm has particular challenges compared to the l_1 and l_2 norms. In particular, optimization in the zero norm is likely to produce a non-convex function. This means that the function has local minima or saddle points where the optimization algorithm can be stuck.

Because optimization in the l_0 norm produces non-convex functions, it is preferable to approximate the l_0 norm using the l_1 norm. Optimization using the l_1 norm results in a convex function, which can be robustly solved using standard optimization methods.

Even though optimization using the l_1 norm does not minimize the number of non-zero elements, it reduces the number of non-zero elements, which is the goal of the using the l_0 norm. Finally, the l_0 norm is unable to reduce the noise associated with the input data. In contrast, the l_1 norm effectively de-noises the input data, since the it minimizes the total magnitude of all elements.

Since the l_1 norm approximates the relevant properties of the l_0, but produces a convex function to be minimized, it is norm used for compressed sensing.

3.2 Sparsity

Similar to the definition of Sparse within the English language, a natural signal is sparse when it contains a small number of non-zero coefficients. The number of coefficients which contain most of the information can be counted and is equal to S. For example, if a signal has 100 coefficients which contained an acceptable amount of information, it would be said that the signal is 100 sparse. The sparsity of a signal is of great importance as it determines how efficiently it can be determined.

3.3 Incoherence

Coherence is defined as the largest correlation between any elements of two matrices or vectors. For the case of Compressed Sensing coherence is used between the sensing (Φ)and representation (Ψ)matrices or basis. The inner product between all the elements of the two basis is used to give a numerical answer for the correlation (μ) [**?**].

$$\mu(\Phi, \Psi) = \sqrt{n} \max |\langle \Phi, \Psi \rangle|$$

Compressed Sensing focuses on low coherence or incoherence, where the two basis are as dissimilar as possible. This is needed for a number of reasons; to ensure that in some basis the signal is sparse, to ensure that if the original signal contained noise, the representation basis does not have the same issue, the minimum number of measurements remains small and that the Restricted Isometry Property will hold for the algorithms described later. The minimum number of measurements is proportional to both the sparsity and correlation through the relation

$$m \geq C \cdot (\mu(\Phi, \Psi))^2 \cdot S \cdot logn$$

where C is a positive constant.

It has been found that the basis share pairs, meaning a random matrix is largely incoherent with any random chosen basis. In more specific scenarios, the Fourier basis and the spike basis work together to produce incoherence, and wavelets and noiselets are paired together.

3.4 Compressed Sensing

Information in the modern world is transported through electrical, sound or electromagnetic signals that are composed of a complex combination of smaller waves. Receiving these signals and transforming them back into useful information may be easy for a human eye or ear, however it is more difficult for a computer. In order for a computer to receive the signal, a sensor must be used to collect and store the information. Every sensor has a limit to the speed at which it can take another measurement after the previous one, this value is known as the sampling frequency. The sampling frequency, or the Nyquist rate, for any situation is determined as at least two times the maximum frequency within the signal, or

$$f_{sampling\ rate} \geq 2f_{maximum\ signal\ frequency}$$

Compressed sensing is a technique used in the collection and processing of these signals that reduces the total number of samples the sensor must collect. By having a lower number of total samples, the compressed sensing sampling frequency can also be lower that the Nyquist rate,

$$f_{Nyquist\ rate} \geq f_{compressed\ sensing\ sampling\ rate}$$

All of these samples are represented as vectors and matrices which make the linear system

$$y = Ax$$

where the vector $y \in R^M$ is the observed data coming in from the signal, $A \in R^{M \times N}$ is the observation matrix, and the vector $x \in R^N$ is the reconstructed signal. With compressed sensing requiring less observations, this makes the dimension M of the system much smaller than N, $M << N$. Within the language of numerical linear algebra, this is an under-determined system, or that there are more unknowns than equations to solve for them. One algorithm to make an approximate solution can be found using the l_1 norm

$$\min ||x||_1 \ in\ Ax = y$$

Many natural signals measured from the environment have the inherent property of being sparse. This means that the incoming signal can be represented as a sum of signals such as

$$x = \sum_{i=1}^{N} a_i x_i$$

By having only a specific number of non-zero components, very little information must be stored and the reconstruction of the signal can be done efficiently by only focusing on these components. Some of the more basic minimization algorithms used in compressed sensing are Matching Pursuit (MP) and Orthogonal Matching Pursuit (OMP). These algorithms are more efficient in the l_0 and l_1 norms than other more well known methods such as Newtons Method. The l_0 norm counts the number of total nonzero elements within the vector, which cannot be done by Newtons method. Instead, it would be easier to use the l_1 norm in the algorithms as it instead calculates the combined magnitude of all coefficients [3].

3.5 Compressed Sensing with Noise

Numerical Methods and its intersection with computers and mechanical devices agree that all systems carry error and limits as to how a number is represented. For computer systems, their design limits any number smaller than machine precision to be incorrectly represented. Real world signals carry error or noise, caused by measurement from undesired objects in the environment. This changes the compressed sensing algorithm. To introduce noise another vector will be added [4]

$$y = Ax + n \tag{2}$$

A similar algorithm to the case of no noise can be used to find an approximate solution to this problem.

$$min||x||_1 \ in||Ax - y||_2 \leq \epsilon$$

3.6 Algorithms

Once the approximate solution is defined, a one norm algorithm can be used to find the best vector x that gives the closest reconstruction to the original signal. One algorithm is the Matching Pursuit (MP) algorithm, which requires the incoming signal y and a chosen sensing matrix A. The matrix A can be chosen to match the pattern of the incoming signal. If it is known what kind of functions the signal is produced by, sinusoidal, Fourier Series, or a type of normal distribution like Gaussian or Bernoulli, then A can be chosen to match the specific function [5]. The next step is to calculate the inner product of the received signal with the sensing matrix

$$x_i = a_i \cdot y$$

and calculate a new signal that removes the part of the signal that was associated with one of the vectors in the sensing matrix.

$$y = y - x_i \cdot a_i$$

This process is repeated until the wanted value of the one norm is found or a maximum number of iterations is reached.

An improvement on the basic algorithm is to orthogonalize the sensing matrix after updating the reconstructed signal. This method is called Orthogonal Matching Pursuit (OMP) and checks that each new component of the reconstructed signal is independent of the previous components, which can further help in improving the sparsity of the reconstructed signal.

Orthogonal Matching Pursuit starts by calculating the inner product of the received signal with the sensing matrix with the addition that the maximum value is chosen as the new component of the reconstructed matrix

$$x_i = max[a_i \cdot y]$$

Any orthogonalization technique such as Gramm-Schmidt can be used on A to make all other vectors orthogonal to the one used in the above calculation. Then the new signal is calculated

$$y = y - x_i \cdot a_i \tag{3}$$

Once both values are found the process can be repeated until the wanted value of the one norm is found or a maximum number of iterations is reached.

4 K-SVD Algorithm

4.1 Dictionaries

In the K-SVD algorithm, a Dictionary is a collection of basis vectors that are used to represent the data samples. Each data sample can be approximated as a linear combination of these vectors. The key difference for a K-SVD Dictionary is that vectors are not restricted to be linearly independent, the dictionary can contain vectors that are combinations of the others, leaving the total number of elements larger than the number of points within the data. This phenomenon is called over-completeness or the dictionary is over complete. Over-completeness is often intentionally chosen in dictionary learning tasks to capture more complex and diverse patterns in the data. By having more vectors than dimensions, the dictionary can potentially represent richer structures and variations present in the data. The drawbacks of being over-complete is the additional computational cost and finding the best solution from the additional number of available solutions.

4.2 Algorithm

Using the Over-complete Dictionary, the idea is to reconstruct a signal or other type of input using the smallest number of Dictionary Elements, subject to the desired error. To Start the algorithm, the first estimate of the sparse solution is found using the Orthogonal Matching Pursuit (OMP) algorithm based the actual input signal and the dictionary. Once the first estimate is found, the iterations to reduce the error between the exact and estimated solutions can start. The error is stored as

$$E = Y - D_j \cdot X_j$$

where each iteration uses the next row of D, D_j, and the next column of X, X_j. Once the difference is calculated, the Singular Value Decomposition of E is calculated to find which singular value contains the highest difference.

$$E = U \Sigma V^T$$

The Dictionary row is updated by re-assigning the row to equal the first row of U, U_1

$$D_j = U_1$$

Once the dictionary has been updated, the sparse solution will then be updated by

$$X_j = \Sigma_1 V_1^T$$

After each reassignment, the two norm error is rechecked to see if the error is smaller than the tolerance. While this algorithm is more computationally costly to calculate the SVD at each iteration, the benefit is that each iteration ensures the most influential singular value is chosen [6].

5 Applications of Data Reduction Algorithms

5.1 Neural Network Reduction

One goal of a neural network is to predict the classification of a set of inputs. A supervised network must have some data that is labeled to train on in order for it to correctly predict the label of a new test set. The basis of machine learning relies on computers being able to solve linear and non linear systems. Inputs such as signals, pictures or sets of data that are represented through vectors or matrices are continuously transformed until it can predict the label of that information.

Each transformation is a step, and is also represented as a matrix, referred to as the weight matrix or weights. Using Python and its TensorFlow package to create a small neural network that classifies a randomly generated data set into one of two categories, it is possible to use the Singular Value Decomposition Algorithm, since the final weights of the neural network are structured as a matrix. The Singular Value Decomposition Algorithm can compress the size of the final weight matrix in order to save storage space and the amount of calculations required to classify new data that is being inputted into the network [7].

Using the Compressed Neural Network code from the GitHub to compare the accuracy between the original and compressed models, the number of components for the first test is set to 20. After the first test, two more will be shown with the number of components changed to 50.

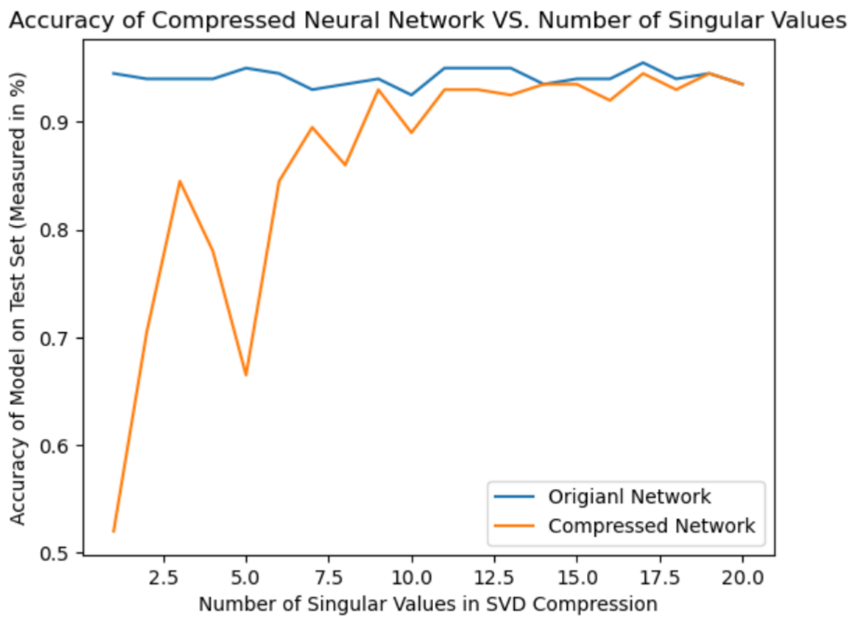


Fig. 2 Compares the accuracy between the two models versus the number of singular values kept in the SVD compression, with the number of components set to 20.

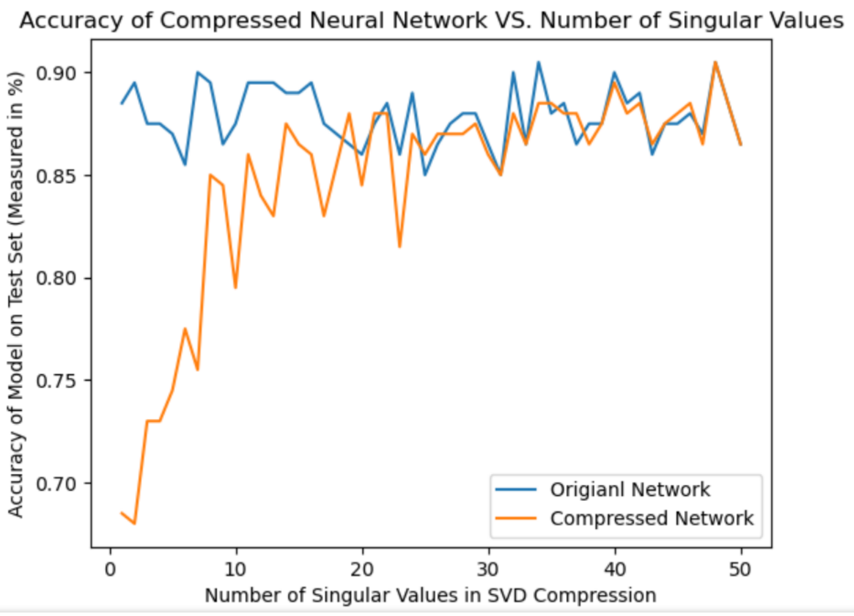


Fig. 3 Compares the accuracy between the two models versus the number of singular values kept in the SVD compression, with the number of components set to 50.

From Figure 2, it can be seen that including 9 singular values in the compressed network only decreases the accuracy by roughly one to two percent, but reducing the size of the network in half.

From Figure 3, it can be seen that including approximately 20 singular values in the compressed network on average is comparable to the same accuracy of the original network but at 40% of the size. For this random data set, the compressed neural network is effective in reducing the size while maintaining the accuracy of the model.

In order to test on a larger data set and network, the mnist data set from TensorFlow Keras will be used. Below are the first ten test images of the mnist data set and their corresponding correct values which will be used to compare to the predicted values generated from the Neural Network.

```
The Correct Label is shown above each image:
```

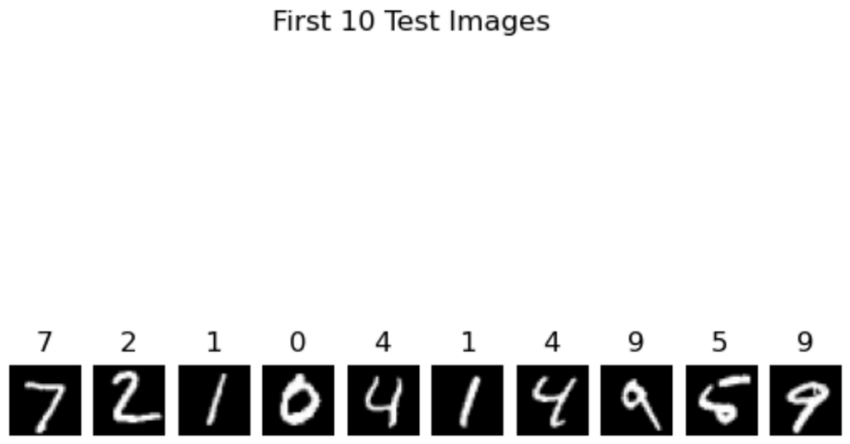


Fig. 4 Shows the first 10 images of the test set as well as the correct labels for each image.

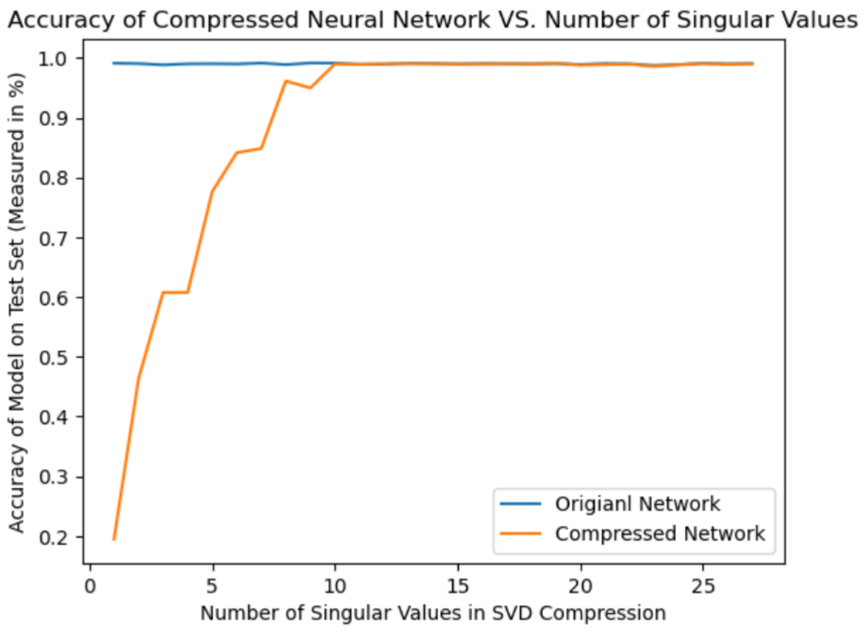


Fig. 5 Compares the accuracy of the two networks versus the number of singular values in the compressed model.

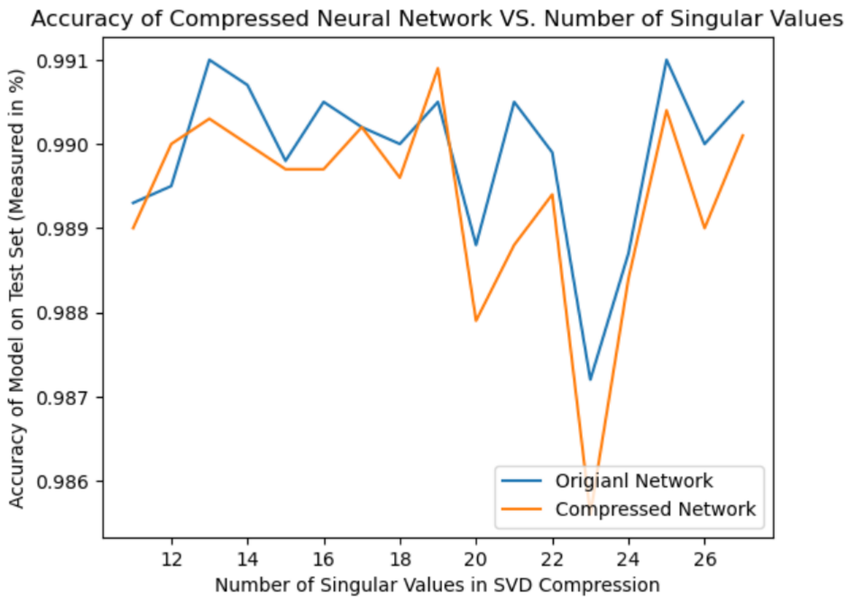


Fig. 6 Compares the accuracy of the two networks versus the number of singular values in the compressed model, zoomed in on values 10-28.

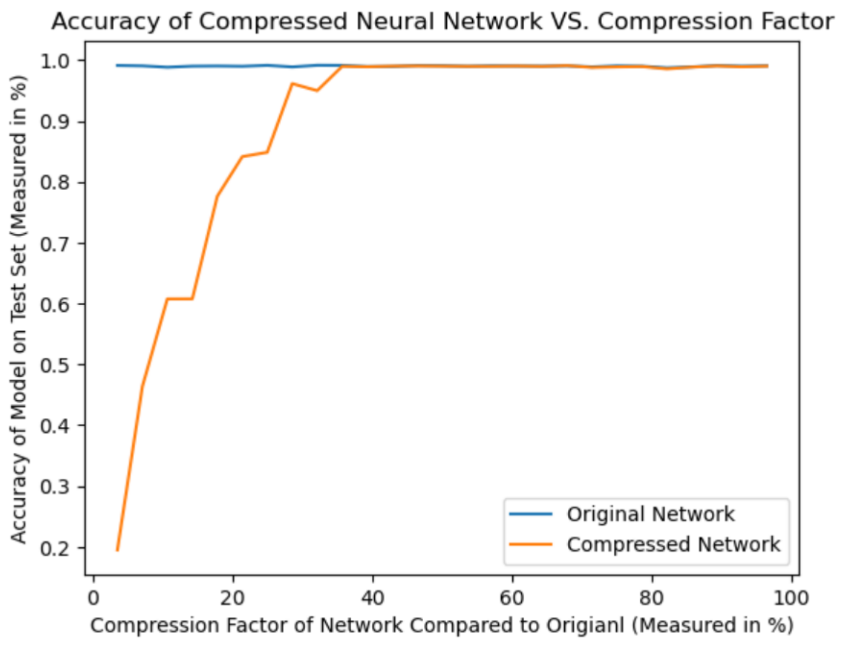


Fig. 7 Compares the accuracy of the two networks versus the percent of singular values in the network.

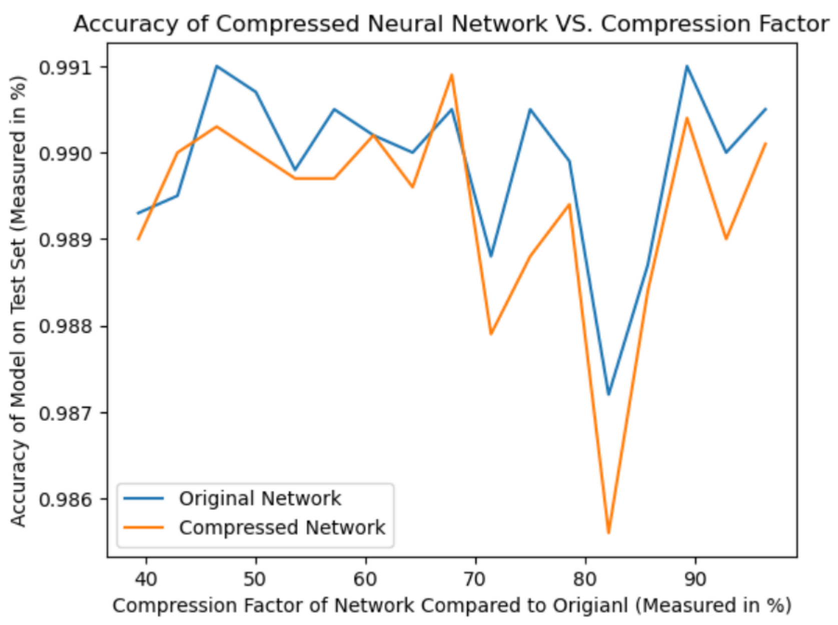


Fig. 8 Compares the accuracy of the two networks versus the percent of singular values in the network, zoomed in on values 10-28.

After running the Neural Network, it can be seen from the Accuracy Plots that the number of singular values needed to obtain roughly the same accuracy is 10, or just under 40% of the total number of singular values. This means that using SVD on the Network can have significant benefits without any noticeable negative effects.

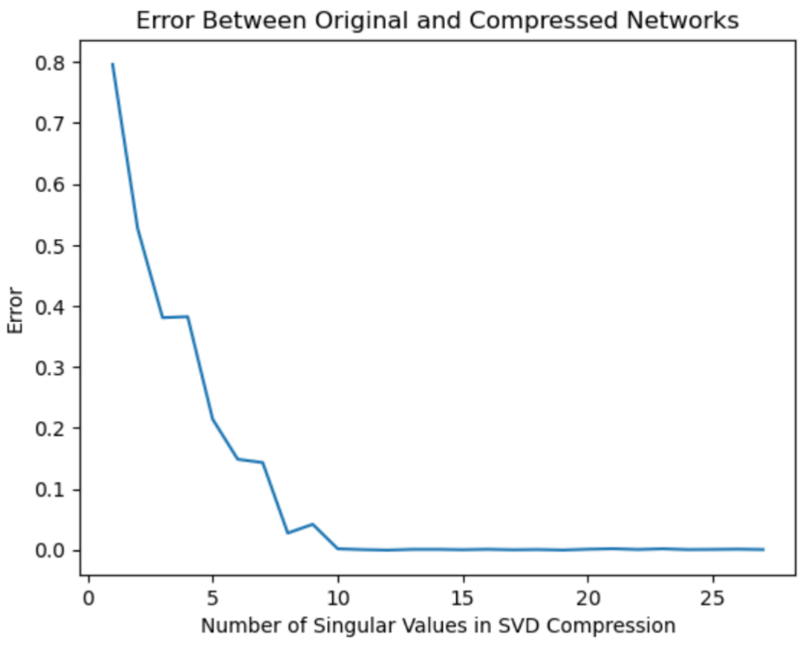


Fig. 9 Plots the error between the original model and the compressed model versus the number of singular values in the model.

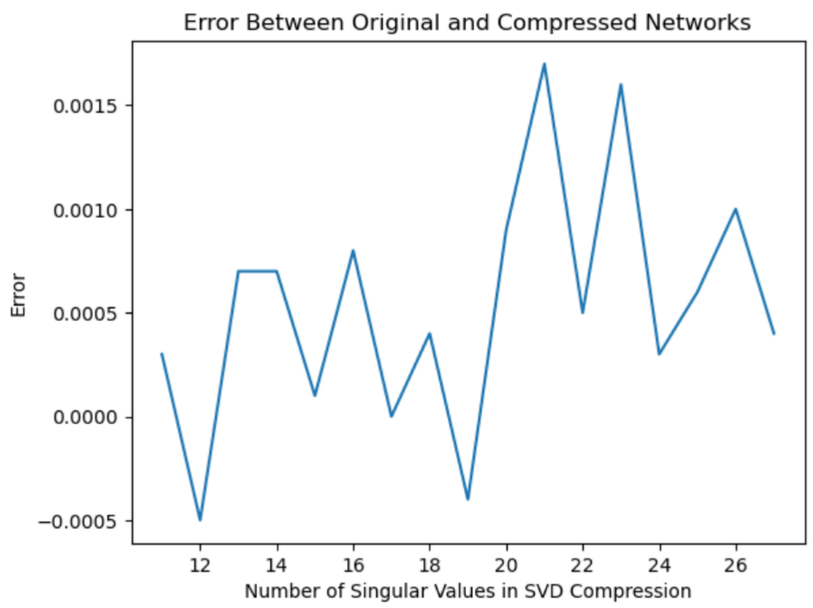
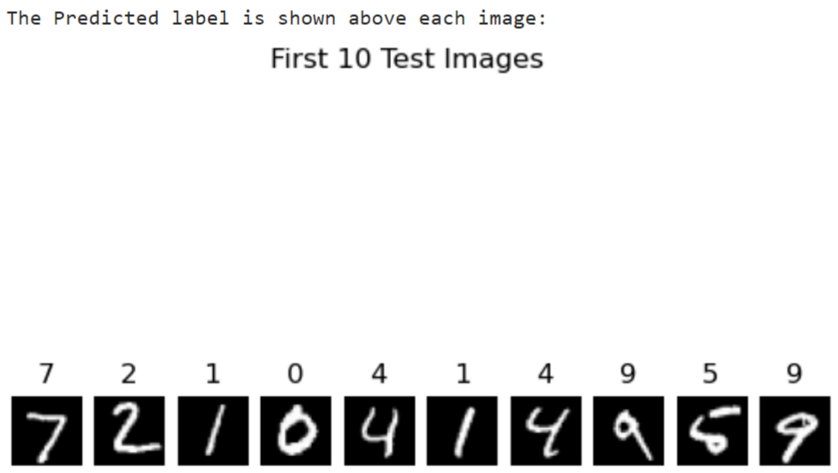


Fig. 10 Plots the error between the original model and the compressed model versus the number of singular values in the model, zoomed in on values 10-28.

Looking at the error between the Networks, the error roughly exponentially drops until 10 singular values where it takes on a constant value, showing that 10 singular values is the minimum number of values that can be taken.



```
The Predicted label is shown above each image:
```

Fig. 11 Shows the first 10 images of the test set as well as the predicted labels for each image from the compressed model.

Evaluating the predicted values of each of the first 10 test images reveals that all were labeled correctly by the Neural Network.

In terms of computational complexity, when new data sets are applied to the neural network, the computational complexity of the new members is significantly cheaper to calculate using the compressed final weights instead of the original networks' final weights. This is important when the neural network needs to quickly determine which class the new set member belongs to, as unnecessary calculations are wasting time. As well, when the network is running on a small or end-device, where the amount of storage and processing power the device has cannot handle processing the whole network, but could be designed to store and manage a smaller network, reducing cost of manufacturing and maintaining.

5.2 Image Reconstruction and Compression

Using the compressed sensing algorithm on different data types can explore the usability and efficiency of the algorithm. Starting with an example using a sparse signal in a Fourier Sampling Basis and a Spike Reconstruction Basis, the Compressed Sensing Fourier program will show the implementation of the algorithm [8].

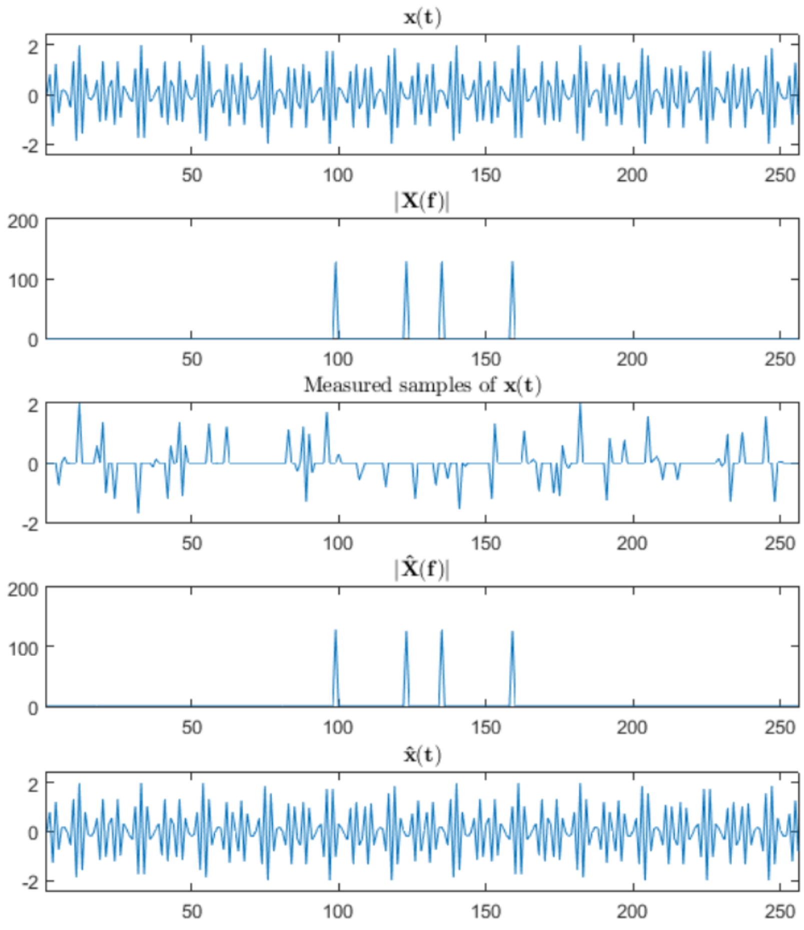


Fig. 12 Compares the original signal, the non-zero elements in a Fourier Sensing Basis, what parts of the signal are measured, the non-zero elements in the Spike Reconstruction Basis and the reconstructed signal.

It is seen that the number of measured samples of the original signal is much lower than both the number of samples the original signal would have and the Nyquist rate.

A secondary example takes a grey-scale section of the camera-man image, transforms the matrix into a one dimensional signal and applies the compressed sensing algorithm on it. The MATLAB program Compressed Sensing Image compares the two reconstruction methods, the least squares approximation and the compressed sensing algorithm [9].

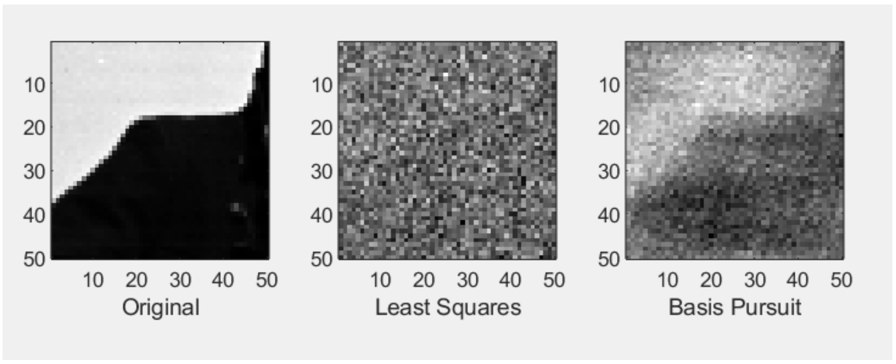


Fig. 13 Compares the original grey-scale image with the reconstructed least squares approximation image as well as the discrete cosine basis paired with basis pursuit for reconstruction. Only 16.67% of the Signal was being measured.

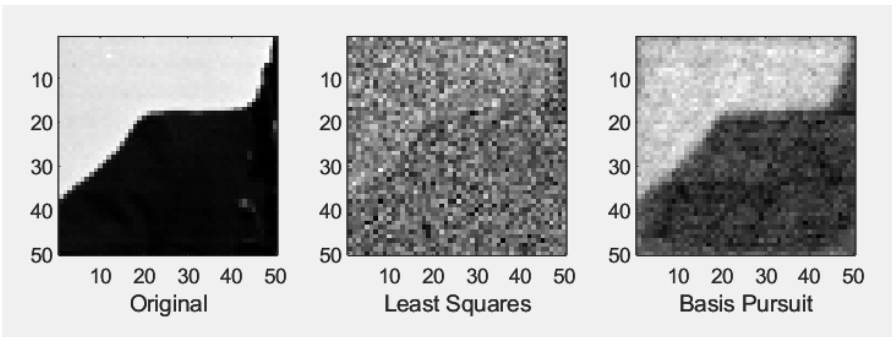


Fig. 14 Compares the original grey-scale image with the reconstructed least squares approximation image as well as the discrete cosine basis paired with basis pursuit for reconstruction. Only 50.00% of the Signal was being measured.

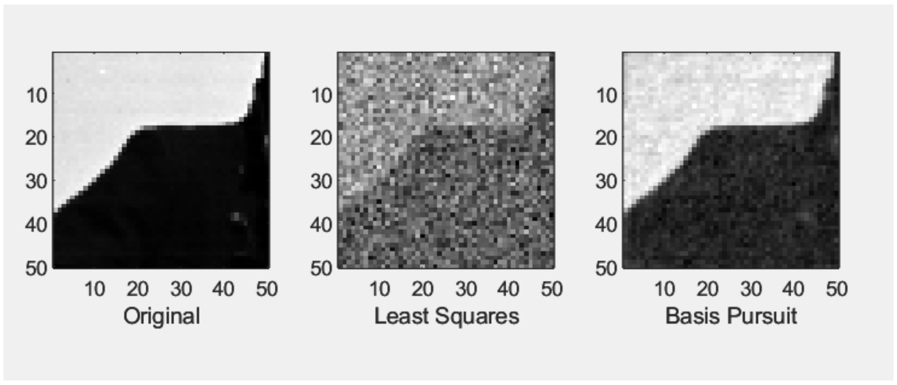


Fig. 15 Compares the original grey-scale image with the reconstructed least squares approximation image as well as the discrete cosine basis paired with basis pursuit for reconstruction. The full signal was being measured.

By inspection, the basis pursuit paired with compressed sensing significantly outperforms a basic least squares approximation in reconstructing the section of the image, no matter how much of the signal was observed.

Using the natural property of sparsity in real world signals, compressed sensing can allow for substantial reduction in the dimension and calculating of those signals. This is useful in the world of machine learning and neural networks as it acts as a pre-processor of data before it enters the network. When the data sets entering the network are smaller, the network can complete its calculations faster and more efficiently. This is the motivation to use many different compression schemes, as the worlds data becomes larger the cross-section of all sciences must work towards managing the cost of using big data.

5.3 Learned Dictionary Reconstruction

In this example, the K-SVD algorithm will be applied to a randomly generated data set, with the max sparsity of the dictionary changed to test the accuracy of the reconstructed signal at different values of sparsity [10] [11].

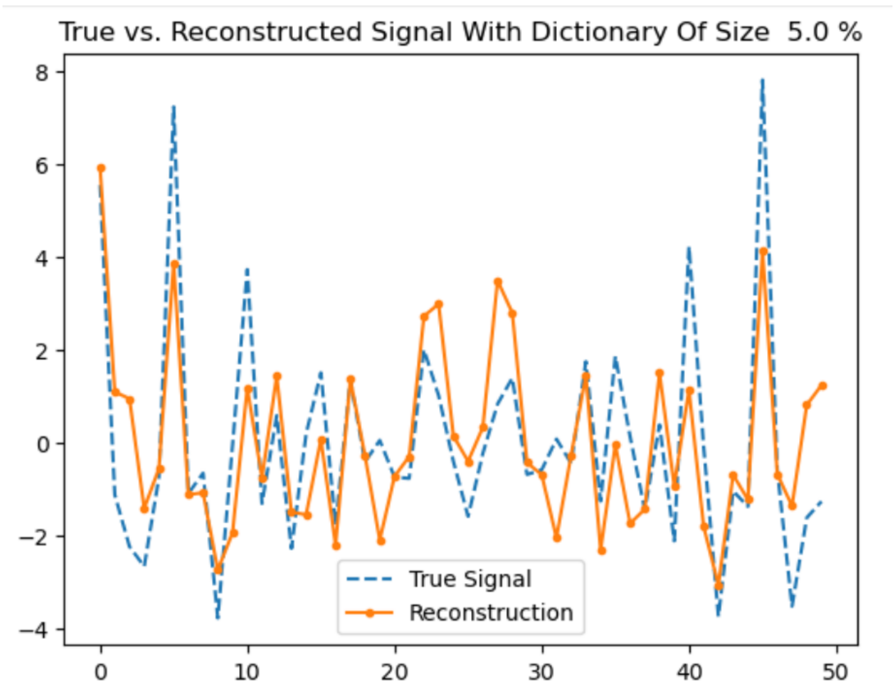


Fig. 16 Comparison between the original signal and the K-SVD reconstruction based on 5% of the total dictionaries elements.

Figure 16 starts the program with the smallest sparsity values, or 5% of the total elements of the dictionary. The error difference between the exact and reconstructed signal is high at this stage, but will be reduced by increasing the maximum sparsity.

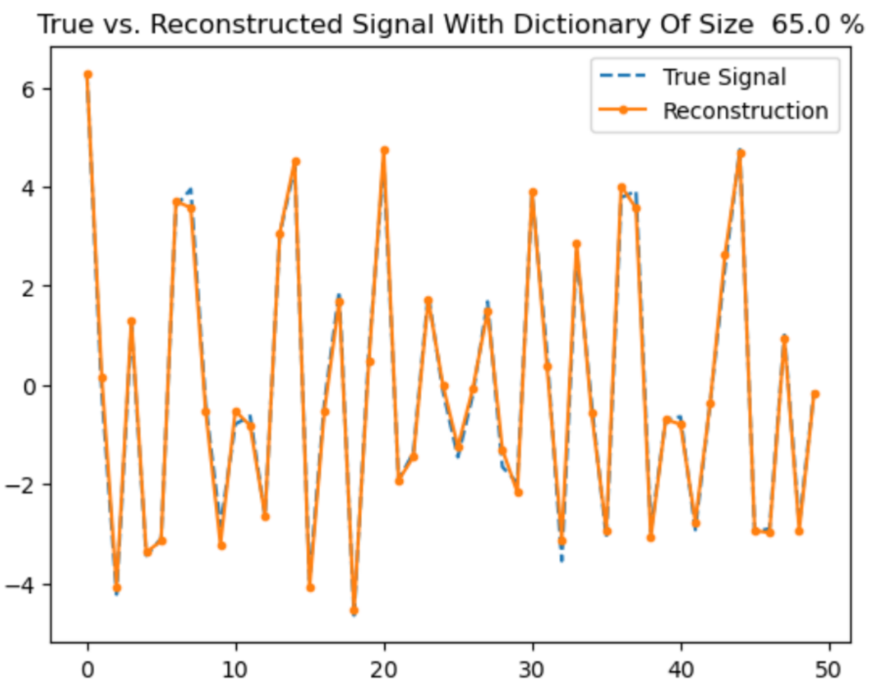


Fig. 17 Comparison between the original signal and the K-SVD reconstruction based on 65% of the total dictionaries elements.

Figure 17 shows that upon visual inspection that at 65% of the total dictionary elements included in the reconstruction of the original signal can produce similar results within a small error. By only including a percent of the total number of elements, the computational cost and memory space in storing and reconstruction are also reduced to 65%. Including more dictionary elements after this iteration will reduce the error further, but at the cost of more calculations and memory space.

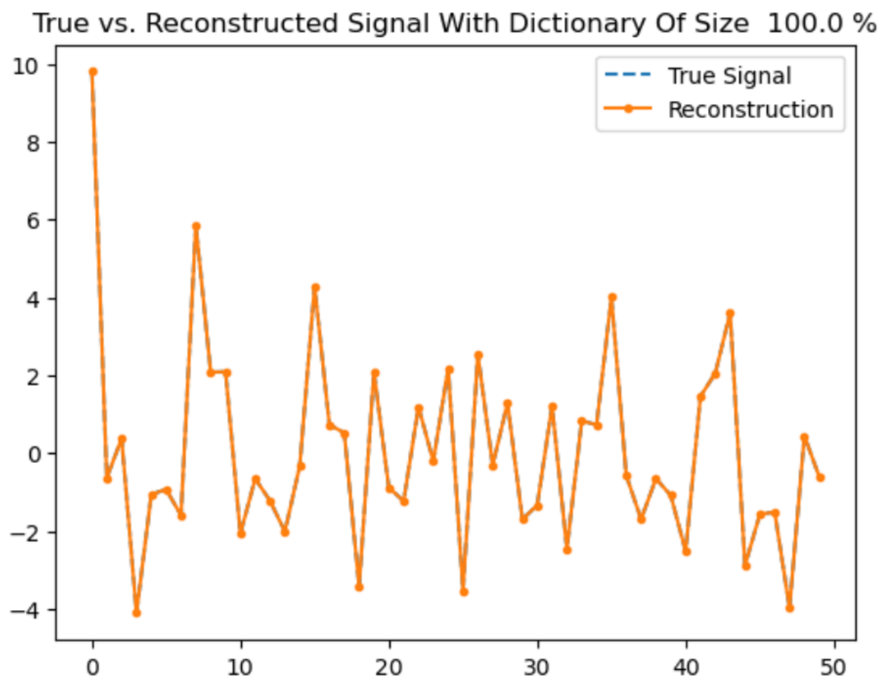


Fig. 18 Comparison between the original signal and the K-SVD reconstruction based on 100% of the total dictionaries elements.

Figure 18 demonstrates that when all the elements of the dictionary are included in the reconstruction of the signal, the reconstruction signal simply becomes the true signal from the start. This makes the algorithm valid, since the limit of the K-SVD algorithm produces the exact signal.

In real world applications where the dictionary elements are no longer random, but contain large amounts of images for facial recognition, having a sparse algorithm such as the K-SVD is useful as it can limit the redundancy of including similar or less important images in the reconstruction stage. In the case of facial recognition with images including glasses or scarves, a full dictionary reconstruction does not allow for the classification of the correct face with the new facial coverings, since the elements will want to create an old picture, not a new one. With the K-SVD algorithm, only to most important features of the face are included allowing for some variation in the new image to correctly classify the face [12].

6 Conclusion and Future Work

The next stage of this research report would be to research the compression algorithms that compete with Singular Value Decomposition such as Discrete Cosine Transform and LZW compression. This would allow for more comparison of modern algorithms and might increase the effectiveness of the neural network application.

More investigation would be done into the compressed sensing example, in order to reconstruct the whole picture not just one piece of it. A limitation could be that the example did not construct the image sensing matrix as a wavelet but just a Fourier basis.

In order to further test the K-SVD algorithm, the example would be changed to have the same dictionary representation each iteration to reconstruct the signals. In addition, another program which would test on facial recognition images instead of randomly generated signals would be completed to explore extended applications of the algorithm.

Accompanying Files. All files used in the writing of this research paper are stored in this GitHub repository. This includes the MATLAB and JUPYTER code examples which reproduce the graphs in the paper, the l1-magic file needed for compressed sensing, as well as the journal timeline of work completed and the research papers of Lily Seebach and Zhenjie Qu which are cited in the references.

Acknowledgements. I would like to thank Dr. Nicholas Kevlahan for all his help and guidance in the writing of this research paper.

References

[1] Clissa, L., Lassnig, M., Rinaldi, L.: How big is big data? a comprehensive survey of data production, storage, and streaming in science and industry. Frontiers in Big Data **6** (2023) https://doi.org/10.3389/fdata.2023.1271639

[2] Seebach, L.: The implementation of singular value decomposition in machine learning, 1–16 (2023)

[3] Maronidis, A., Chatzilari, E., Nikolopoulos, S., Kompatsiaris, I.: In: Emrouznejad, A. (ed.) Smart Sampling and Optimal Dimensionality Reduction of Big Data Using Compressed Sensing, pp. 251–280. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30265-2_12 . https://doi.org/10.1007/978-3-319-30265-$2_1$2

[4] Qu, Z.: Compressive sampling, 1–13 (2021)

[5] Eisert, J., Flinth, A., Groß, B., Roth, I., Wunder, G.: In: Kutyniok, G., Rauhut, H., Kunsch, R.J. (eds.) Hierarchical Compressed Sensing, pp. 1–35. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-09745-4_1 . https://doi.org/10.1007/978-3-031-09745-4_1

[6] Bryt, O., Elad, M.: Compression of facial images using the k-svd algorithm. Journal of Visual Communication and Image Representation **19**(4), 270–282 (2008) https://doi.org/10.1016/j.jvcir.2008.03.001

[7] Cai, C., Ke, D., Xu, Y., Su, K.: Fast learning of deep neural networks via singular value decomposition. In: Pham, D.-N., Park, S.-B. (eds.) PRICAI 2014: Trends in Artificial Intelligence, pp. 820–826. Springer, Cham (2014)

[8] CodeProject: Compressed Sensing: An Introductory Tutorial with MATLAB. Available online. Accessed on April 13, 2024 (2014). https://www.codeproject.com/articles/852910/compressed-sensing-intro-tutorial-w-matlab

[9] Gibson, S.: Simple Compressed Sensing Example. https://www.mathworks.com/matlabcentral/fileexchange/41792-simple-compressed-sensing-example. MATLAB Central File Exchange (2024)

[10] Aharon, M., Elad, M., Bruckstein, A.: K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. IEEE Transactions on Signal Processing **54**(11), 4311–4322 (2006) https://doi.org/10.1109/TSP.2006.881199

[11] Rubinstein, R., Zibulevsky, M., Elad, M.: Efficient implementation of the k-svd algorithm using batch orthogonal matching pursuit. Technical report, CS Technion (April 2008)

[12] Zhang, Q., Li, B.: Discriminative k-svd for dictionary learning in face recognition. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 2691–2698 (2010). https://doi.org/10.1109/CVPR.2010.5539989