Randomized SVD for Fast, Online, and Parallel Data Decomposition: Applications to Neuroscience, Image Processing, Audio Processing

Blake Bordelon, Patrick Ohiomoba, Eleonora Shantsila

January 18, 2021

1 Introduction

Singular value decomposition (SVD) is a linear algebra method used for matrix factorisation with far reaching applications, including in machine learning, image processing and statistical analysis. However, the computation of the SVD for a matrix can be both computationally expensive and very time consuming. The time and space complexity of computing the standard SVD differs from method to method, but for instance the standard Matlab algorithm has $O(mn \quad min(n, m))$ time complexity for an mxn matrix [1]. Consequently, it would be impractical to implement the method for large matrices leading to the development of a set of randomised SVD methods. In this report, we firstly explore a series of applications, where the computation of SVD is required but too computationally expensive. We then outline a number of random SVD algorithms and our results from implementing these.

In the present work, we will show the benefit that randomized SVD can provide on large data analysis problems that arise in neuroscience, audio processing, and visual foreground-background separation. In these settings, high dimensional datasets typically exhibit low-rank structure which can be identified with high accuracy utilizing random projections to lower dimensional spaces.

2 Background on SVD

Although, SVD was originally discovered independently by Eugenio Beltrami and Camille Jordan, as well as James Joseph Sylvester, Erhard Schmidt and Hermann Weyl using an alternative approach. One particular SVD algorithm of note is the Golub and Reinsch algorithm from 1970 [**?**], which proposed the now 'standard' computational method for calculating SVD. Modern packages use various approaches depending on the matrix structure, giving rise to slightly different computational complexities.

In 1998 the first randomised SVD method was introduced by Alan M Frieze, Ravi Kannan and Santosh S Vempala, where they develop an algorithm which samples the entries of the matrix according a probability distribution. The algorithm complexity is independent of m and n, meaning that it is constant in time. A more detailed exploration of the algorithm is given below.

3 Singular Value Decomposition

The (reduced) singular value decomposition of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ with $m \geq n$ can be written as

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

where $\mathbf{V} \in \mathbb{R}^{n \times n}$ is an orthogonal matrix, $\mathbf{U} \in \mathbb{R}^{m \times n}$ contains n orthonormal column vectors, and $\mathbf{\Sigma} \in \mathbb{R}^{n \times n}$ is a diagonal matrix with non-negative entries. This decomposition is a workhorse of data processing systems. It is straightforwardly related to principal component analysis (PCA), where the correlation matrix $\mathbf{C} = \mathbf{A}\mathbf{A}^\top$ can be obtained in terms of the singular values and left singular vectors of $\mathbf{A}$

$$\mathbf{C} = \mathbf{A}\mathbf{A}^\top = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^\top \tag{2}$$

We therefore see that the rank of $\mathbf{C}$ is limited by the rank of $\mathbf{A}$, the principal values are given by the diagonal elements of $\mathbf{\Sigma}^2$, and the principal vectors are the column vectors of $\mathbf{U}$. Further, $\mathbf{A}$ has better condition number than $\mathbf{C}$ so the SVD provides a more numerically stable method for computing principal components analysis.

3.1 The Compute and Memory Requirements of SVD

As we discussed in class, the The Golub-Kahan SVD method requires $\sim 4mn^2 - \frac{4}{3}n^3$ floating point operations. If $m > 5/3n$ then the Lawson-Hanson-Chan update obtains a slightly smaller number of $\sim 2mn^2 + 2n^3$ operations. In either case, when $m >> n$, the leading behavior requires $O(mn^2)$ operations. To improve this scaling, we will explore a randomization strategy.

4 Randomization to Reduce Memory and Compute

4.1 Approximation of the span of A

If $\mathbf{A}$ is rank $k \leq n$, then the image of k random vectors under $\mathbf{A}$ will, with probability one, span the column space of $\mathbf{A}$. We can therefore generate a sequence of independently sampled random vectors $\mathbf{x}_j \in \mathbb{R}^n$, $j = 1, ..., k$ and compute

$$\mathbf{y}_j = \mathbf{A}\mathbf{x}_j \in \mathbb{R}^m \; j = 1, ..., k \tag{3}$$

The matrix $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_k] \in \mathbb{R}^{m \times k}$ approximates the span of $\mathbf{A}$. Performing the QR decomposition on $\mathbf{Y} = \mathbf{QR}$ will only cost $O(mk^2)$ floating point operations. The matrix $\mathbf{B} = \mathbf{Q}^\top \mathbf{A} \in \mathbb{R}^m$. After, SVD can performed on $\mathbf{B} = \mathbf{Q}^\top \mathbf{A} \in \mathbb{R}^{k \times n}$ in $O(nk^2)$ operations.

$$\mathbf{B} = \mathbf{Q}^\top \mathbf{A} = \mathbf{Q}^\top \mathbf{U} \mathbf{\Sigma} \mathbf{V}^\top \tag{4}$$

We thus see that the left singular vectors of $\mathbf{B}$ are $\tilde{\mathbf{U}} = \mathbf{Q}^\top \mathbf{U}$ so we can approximately recover $\mathbf{A}$'s singular vectors with $\mathbf{U} \approx \mathbf{Q}\tilde{\mathbf{U}} = \mathbf{Q}\mathbf{Q}^\top \mathbf{U}$. This relies on the assumption of $\mathbf{x}_j$ being in general position and thus spanning $\mathbf{A}$'s column space. Keeping track of all of the matrix multiplications and factorizations, our total number of FLOPS scales like $kT_{mult} + O((m+n)k^2)$ where T_{mult} is the time required to compute a matrix vector product with $\mathbf{A}$, $O(mn)$. When m and n are much larger than k, this multiplication cost is leading. This motivates exploration of *sparsity, parallelism and structured transforms*, which reduce the time required to perform these parallelizable operations.

4.2 Power Iteration

The eigenvectors of the covariance matrix $\mathbf{C} = \mathbf{A}\mathbf{A}^\top = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^\top$ are the same as $\mathbf{A}$'s left singular vectors. This fact can be used to construct a linear map which accentuates the "stretching" effect of the linear map

$$\mathbf{y}_j^{(q)} = \mathbf{C}^q\mathbf{A}\mathbf{x}_j = \mathbf{U}\mathbf{\Sigma}^{2q+1}\mathbf{V}^\top\mathbf{x}_j \tag{5}$$

As before, we compute $\mathbf{Y} = \mathbf{Q}\mathbf{R}$ and $\mathbf{B} = \mathbf{Q}^\top\mathbf{C}^q\mathbf{A}$. Performing SVD on $\mathbf{B}$ again gives an approximation of the singular vectors and singular values of $\mathbf{A}$. The number of operations is now $k(2q+1)T_{mult} + O(k^2(m+n))$.

4.3 The Basic Randomized SVD Algorithm

1. Compute $\mathbf{Y} = \mathbf{C}^q\mathbf{A}\mathbf{X}$ for random $\mathbf{X} \in \mathbb{R}^{n\times k}$ and $q \geq 0$.
2. Find the QR decomposition for $\mathbf{Y} = \mathbf{Q}\mathbf{R}$ where $\mathbf{Q} \in \mathbb{R}^{m\times k}$
3. Calculate $\mathbf{B} = \mathbf{Q}^\top\mathbf{A} \in \mathbb{R}^{k\times n}$.
4. Find the singular value decomposition for $\mathbf{B} = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\mathbf{V}^\top$.
5. Approximate $\mathbf{A}$'s left singular vectors $\mathbf{U} \approx \mathbf{Q}\tilde{\mathbf{U}}$
6. Approximate the singular values of $\mathbf{A}$ with $\mathbf{\Sigma} \approx \tilde{\mathbf{\Sigma}}^{1/(2q+1)}$
7. Return $\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}$, which provide a low rank approximation of $\mathbf{A} \approx \hat{\mathbf{A}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$

This justification and theoretical guarantees of this algorithm are discussed in greater detail by Halko, Martinson, and Tropp (2010) [2].

4.3.1 Reconstruction Error

A simple bound on the reconstruction error for a rank-k approximation of $\mathbf{A}$ provides insight into the accuracy of this randomized method. Let $\sigma_1 > \sigma_2 > ... > \sigma_p$ be the singular values of $\mathbf{A}$. By considering perfect estimation of the top k singular vectors and singular values provides a lower bound on the reconstruction error

$$||\mathbf{A} - \hat{\mathbf{A}}||_F \geq \sqrt{\sum_{j=k+1}^{p} \sigma_j^2} \tag{6}$$

Therefore, the relative error $E_R(\mathbf{A}, \hat{\mathbf{A}})$ enjoys the bound

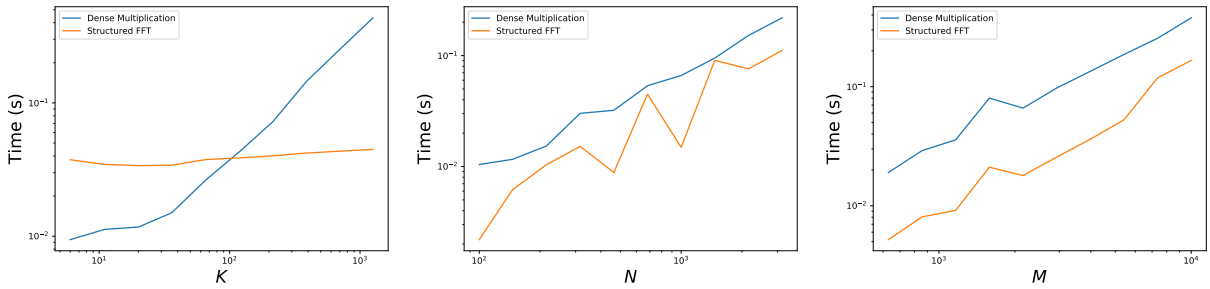
$$E_R(\mathbf{A}, \hat{\mathbf{A}}) \geq \frac{||\mathbf{A} - \hat{\mathbf{A}}||_F}{||\mathbf{A}||_F} = \sqrt{\frac{\sum_{i=k+1}^{p} \sigma_i^2}{\sum_{i=1}^{p} \sigma_i^2}} \tag{7}$$

For example, a geometric decay in the singular values, $\sigma_j^2 = a^j$ for some $a < 1$ leads to reconstruction error bound $E_R \geq \sqrt{a^k \frac{1-a^{p-k}}{1-a^p}}$. Even as $p \to \infty$, this reconstruction error decays exponentially $E_R \sim a^{k/2}$. For an error tolerance of ϵ, only $k = \Omega(\log(1/\epsilon))$ dimensions are required. Similarly, under the assumption of power-law spectral decays $\sigma_j^2 = j^{-a-1}$ the reconstruction error converges algebraically as well $E_R \sim k^{-a/2}$ as $p \to \infty$. In this case $k = \Omega(\epsilon^{-2/a})$ dimensions are required to achieve the desired error tolerance. This demonstrates why matrices with rapidly decaying spectra can be estimated accurately using only a small number of dimensions k.

4.4 Alternative Matrix Transforms

Assuming that $k < \min\{m, n\}$, the computational bottleneck in the randomized SVD method is the time required to compute the matrix matrix product $\mathbf{Y} = \mathbf{AX}$ with random $\mathbf{X}$. If we use a dense, unstructured random matrix for $\mathbf{X}$, the cost of this operation is $O(mnk)$ time. Alternatively, use of structured matrix transforms, such as Fast Fourier transform or the Hadamard transform can, in principle, reduce the time complexity of this operation by utilizing a divide and conquer strategy. Both the Hadamard and Fast Fourier transform methods would reduce this matrix matrix product to $O(mn \log(k))$ time, which can allow significant improvements if k is large.

Below we plot the average time for dense matrix multiplication or Fast Fourier Transformation of a random matrix $m \times n$. We see that the FFT time scales sublinearly with K, but linearly with M and N.



(a) $M = 5000, N = 2500$ (b) $M = 5000, K = 500$ (c) $N = 2500, K = 500$

Figure 1: Structured Matrix Transformations such as Fourier and Hadamard Transformations can utilize recursion to obtain improvements in the time complexity of the dimension reduction step. For example, we see that the wall time required for Fast Fourier transformation of the $M \times N$ matrix scales very weakly with K, the embedding dimension, while the dense matrix-matrix multiplication scales linearly with the embedding dimension K. The time required for both dense and structured transforms scales linearly with the dimensions N and M of the matrix.

5 Parallelization of Randomized SVD

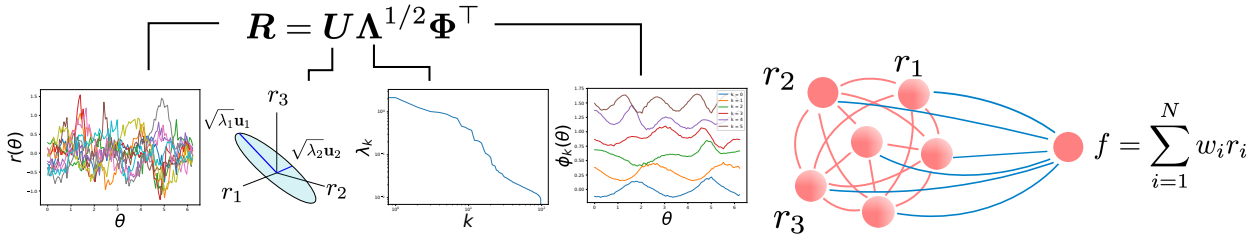
The matrix product computation in the randomized SVD requires $O(mnk)$ time. For sufficiently small k (compared to m, n), this is the leading cost for randomized SVD. With the use of parallel computation, however, large matrix matrix products can be improved performance.

6 Application to Neuroscience: Understanding Population Codes

Due to recent advances in fluorescent imaging, systems neuroscience is experiencing a revolution. The number of neurons N which can be simultaneously recorded under different stimulus conditions is now on the order of $N \sim 10^5$. Calcium imaging experiments of the visual cortex now routinely generate recordings with thousands of distinct presented visual stimuli $P \sim 10^3$. These firing rates R_{ij} of the i-th neuron's response to image j generate a large matrix $\mathbf{R} \in \mathbb{R}^{N \times P}$ containing each neuron's response to the P available stimulus variables. Each of the N rows of the matrix are known as *tuning curves*, which characterize the response profiles of single neurons.

The measured neural responses $\mathbf{R}$ can be thought of as a sample of the brain's *population code*, which specifies how neurons respond to different stimuli. The singular value decomposition of the responses $\mathbf{R}$ provides structural as well as functional information about how neurons encode different stimuli. For instance, the left singular vectors $\mathbf{U}$ indicate the principal componenets of the

covariance of responses in the N dimensional neural response space. The singular values $\sigma_k = \sqrt{\lambda_k}$ specify the standard deviation along each principal component. The right singular vectors Φ are known as *kernel eigenfunctions*, and provide an ordering over which kinds of functions can be estimated in a downstream neuron through linear combinations of the responses.



(a) SVD of Population Responses (b) Linear Readout Model

Figure 2: The singular value decomposition provides insight into neural codes. For a grating stimulus oriented at an angle θ, the tuning curves $r_i(\theta)$ specify the response of each neuron to different gratings. The left singular vectors $\mathbf{u}_k$ $k = 1, ..., p$ are the top k principal components of the neural covariance matrix $\mathbf{C} = \mathbf{R}\mathbf{R}^\top$, which provide an orthgonal basis in the N dimensional space of responses. The singular values, when ordered, identify the amount of variance along each principal component direction. The right singular vectors $\phi_k(\theta)$ are the *kernel eigenfunctions*, which, when ordered by singular values, are the easiest functions to estimate through linear regression of the responses $f \approx \sum_{i=1}^{N} w_i r_i(\theta)$.

6.1 Toy Model of V1 as a bank of Gabor Filters

Before analyzing real neural recordings from the mouse primary visual cortex (V1), we will first work through the various components of randomized SVD and how they will be influenced by properties of the neural population. A simplified, but motivated, model of the visual cortex, inspired by [3], provides many insights into the benefits that randomized SVD will have for neuroscience applications.

A simple model of the primary visual cortex (V1), the V1 neurons filter the responses of photoreceptors in the retina through random Gabor filters. At each point $\mathbf{x}$ in the 2D space of photoreceptors, the response to a given stimulus with parameters $\mathbf{k}, \phi$ is $p(\mathbf{x}, \mathbf{k}, \phi)$. The synaptic connections between the photoreceptor at position $\mathbf{x}$ and the i-th primary visual cortex neuron form a Gabor Filter $\mathcal{G}(\mathbf{x}, \mathbf{k}, \phi_i)$ which has the form

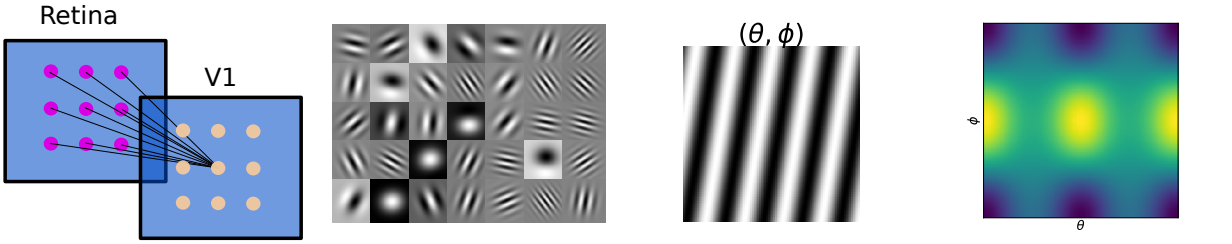
$$\mathcal{G}(\mathbf{x}, \mathbf{k}_i, \phi_i) = \frac{e^{-\frac{1}{2\sigma^2}||\mathbf{x}||^2}}{\sqrt{2\pi\sigma^2}} \cos(\mathbf{k}_i \cdot \mathbf{x} - \phi_i) \tag{8}$$

Each V1 neuron i has a unique Gabor filter with parameters $(\mathbf{k}_i, \phi_i)$. We will consider the responses of V1 cells for a static grating stimulus $p(\mathbf{x}, \mathbf{k}, \phi) = \cos(\mathbf{k} \cdot \mathbf{x} - \phi)$ is obtained through filtration of the photoreceptor response with the Gabor function and rectification through a (possibly nonlinear) function $\psi(z)$

$$r_i(\mathbf{k}, \phi) = \psi(\mathcal{G}_i \cdot \mathbf{p}(\mathbf{k}, \phi)) = \psi\left(\int \mathcal{G}(\mathbf{x}, \mathbf{k}_i, \phi_i) \cos(\mathbf{k} \cdot \mathbf{x} - \phi) d\mathbf{x}\right) \tag{9}$$

It is straightforward to verify that if there is no nonlinear rectification ($\psi(z) = z$), the singular values of an infinite population (with each Gabor randomly sampled $\mathbf{k}_i \sim \mathcal{N}(0, \mathbf{I}), \phi_i \sim \mathcal{U}[0, 2\pi]$) decay as $\sigma_k \sim \frac{1}{\sqrt{k!}}$, which saturates machine precision $\sim 10^{-16}$ around $k \approx 30$. This indicates incredibly low rank structure in the population code, which can be discovered through randomized SVD. On the other hand, if ψ is nonlinear with discontinuous n-th derivative, then the singular value spectrum decays algebraically $\sigma_k \sim k^{-n-1}$. This holds, for instance, for the rectification functions of the form $\psi(z) = \max\{z - a, 0\}^n$ where a is a threshold which sets the sparsity level of the code.

Figure 3 provides an overview of this model of visual cortex responses. In this setting, the V1 neurons process retinal ganglion activity by filtration with a Gabor wavelet, shown in (a). Each neuron's Gabor filter $\mathcal{G}(\mathbf{x}, \mathbf{k}_i, \phi_i)$ responds to a particular orientation $\theta_i = \arctan(k_2/k_1)$, spatial frequency $|\mathbf{k}_i|$ and phase ϕ_i. We consider input stimuli as gratings $p(\mathbf{x}, \mathbf{k}, \phi) = \cos(\mathbf{k} \cdot \mathbf{x} - \phi)$ with spatial wavevector $\mathbf{k}$ and phase ϕ.



(a) Retina to V1 Map (b) Gabor Receptive Fields (c) Grating Stimuli (d) Example Tuning Curve

Figure 3: In a simplified model of early visual processsing, each V1 neuron receives as input a Gabor-filtered response of the photoreceptors in the retina. Each neuron has a possibly different Gabor filter, giving rise to variable tuning curves.

6.1.1 Randomized SVD Performance on Gabor Model

We now discuss the performance of randomized SVD within this Gabor model of V1. Generating the responses of N neurons to P random grating stimuli, we represent the resulting dataset with a response matrix $\mathbf{R}$. Using a freely provided GPU from Google Colab, we analyze the time and relative reconstruction error E_R for different number of random projection dimensions K. The resuls are summarized in Tables 1 and 2. When the population code is infinitely differentiable in the stimulus parameter θ, the response matrix $\mathbf{R}$ has very low rank structure. This smoothness can be exploited and captured through random projection to a space with lower dimension K. This operation has virtually no cost to accuracy until $K \leq 32$, when σ_K begins to exceed machine precision.

Linear Neurons	Full Standard SVD	$K = 10$	$K = 100$	$K = 200$
Time (s) (GPU)	36.7	$(9.27 \pm 0.46) \times 10^{-3}$	$(2.15 \pm 0.03) \times 10^{-2}$	$(4.95 \pm 0.12) \times 10^{-2}$
E_R (GPU)	0	$(1.33 \pm 0.59) \times 10^{-3}$	$5.70 \pm 0.42 \times 10^{-7}$	$(7.82 \pm 0.43) \times 10^{-7}$

Table 1: The accuracy and time of randomized SVD on the Gabor model with linear neuronal activations. The number of neurons is $N = 2^{13}$ and the number of stimulus grid points is $P = 4096$. This experiment was performed on a Google Colab instance with the freely available GPU. Several orders of magnitude reduction in time can be obtained through random projection to $K = 100$ dimensions with a relative error cost of only $\sim 10^{-7}$. This is due to the low rank structure of the linear neuron code.

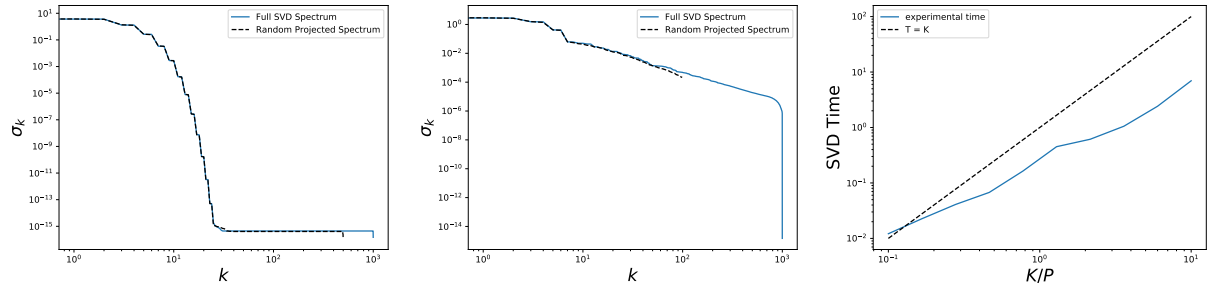
For nonlinear neurons, the matrix has much higher dimensionality, since σ_k decays in a power law. As a consequence, projection to lower dimension K produces a greater relative error E_R. For $K = 100$, the randomized method produces a reconstruction error on the order of 10^{-3}, much more significant than the 10^{-7} obtained for the population of linear neurons.

Nonlinear Neurons	Full Standard SVD	$K = 10$	$K = 100$	$K = 200$
Time (s) (GPU)	20.53	$8.98 \pm 0.23 \times 10^{-3}$	$2.09 \pm 0.05 \times 10^{-2}$	$(4.12 \pm 0.03) \times 10^{-2}$
E_R (GPU)	0	$4.41 \pm 1.01 \times 10^{-2}$	$1.05 \pm 0.02 \times 10^{-3}$	$(3.691 \pm 0.03) \times 10^{-4}$

Table 2: The accuracy and time of randomized SVD on the Gabor responses in a population of nonlinearly rectified neurons with the same parameters as table 1. Random projection to $K = 100$ dimensions reduces the time by a factor of 100 and introduces an relative reconstruction error of $\sim 10^{-3}$, a modest price to pay for such a dramatic improvement in time.

The differences between the spectral properties of these two codes can be examined by plotting how the ordered singular values σ_k decay with index k. Figure 4 shows the spectra for the linear neuron population as well as the nonlinear population. The time required to perform a random

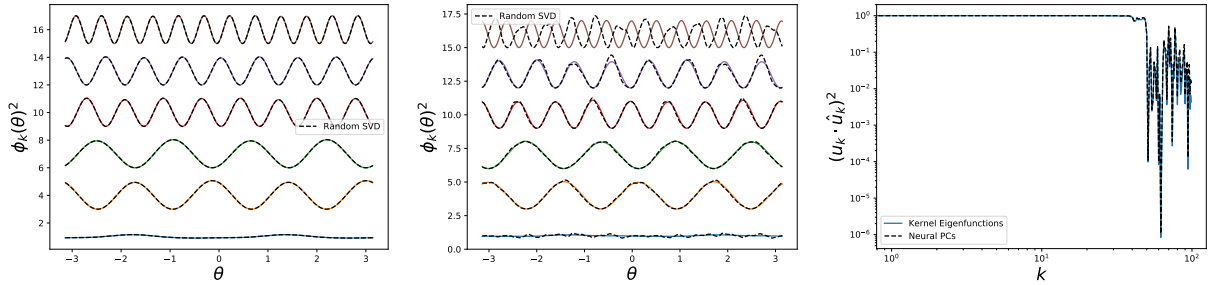
SVD scales roughly sublinearly with the embedding dimension K. For large K, the eigenfunctions can be obtained exactly, while for small K only the top modes are recovered with high accuracy. The plot of the alignment of the ground truth singular vectors and those obtained from randomized SVD are shown in (f).



(a) Low Rank Structure for Linear Neurons

(b) Rectification increases rank

(c) Random Projection SVD Time



(d) Recovered Eigenfunctions $K = 100$

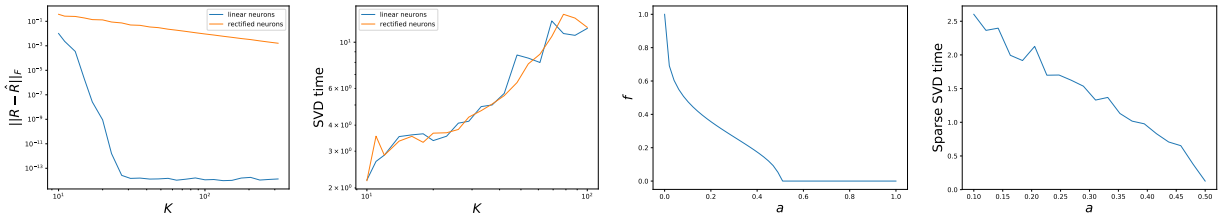
(e) Recovered Eigenfunctions $K = 10$

(f) Singular Vector Alignment $K = 100$

Figure 4: (a) Linear neurons $\psi(z) = z$ exhibit low rank structure, which can be recovered accurately through random projections to very low ~ 500 dimensions. The full SVD (blue) takes 9.06 seconds while the random SVD projected to $k = 500$ takes 0.07 seconds and accurately preserves the singular values. (b) Nonlinear rectification $\psi(z) = \max\{z - a, 0\}$ increases the rank of the population code, which demands a larger number of dimensions k to include in the random projection. The full SVD (blue, 7.99 seconds) is compared with the singular values from a random projection to $K = 100$ dimensional space, (black, 0.0156 seconds). (c) The time required for SVD with random projections scales sublinearly with the projection dimension K. (d,e) The theoretical eigenfunctions are Fourier modes $\phi_{2k}(\theta) = \cos(k\theta), \phi_{2k+1}(\theta) = \sin(k\theta)$. Direct and random SVD both recover the top eigenfunctions quite accurately provided K is sufficiently large. (f) The singular values obtained from randomized SVD $\hat{\mathbf{u}}_k$ are compared with the ground truth singular values $\mathbf{u}_k$. The alignment of these vectors provides an indication of how well the singular vectors are approximated. As expected the top modes are recovered with high accuracy but the accuracy degrades for $k \approx K$.

6.1.2 Sparsity

The sparsity of the neural population code controls the sparsity of the response matrix $\mathbf{R}$. By altering the threshold a of the nonlinear function $\psi(z) = \max\{z - a, 0\}$, we can adjust the sparsity level of the population code. Figure 5 (c) shows the relationship between the threshold and the proportion of nonzero entries f in the matrix $\mathbf{R}$. Using sparse SVD solvers based on ARPACK, we find that increasing the sparsity reduces the amount of time required for singular value decomposition.



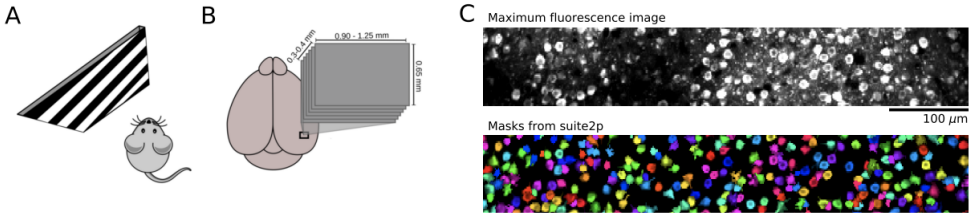
(a) Absolute Reconstruction Error (b) SVD Time (c) Sparsity depends on Threshold (d) Sparsity Reduces SVD Time

Figure 5: Linear and Nonlinear neural populations have dramatically different spectral properties and thus require different numbers of dimensions K to obtain low reconstruction error. (a) The threshold a of the neurons controls the sparsity of the population response matrix $\mathbf{R}$ and thus the expected fraction f of nonzero entries in the matrix $\mathbf{R}$. (b) This sparsifying effect can provide significant performance gains for SVD when utilizing sparse matrix routines.

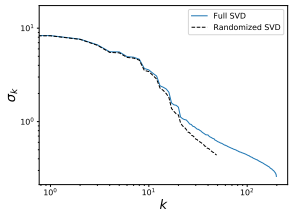
6.2 Experimental Recordings

The Gabor model indicates that the gains that random SVD provides depends on the structure of the neural population code. To identify whether the responses exhibit the low rank structure of an infinitely differentiable code or, rather, exhibit power laws, we will utilize experimental recordings of Mouse primary visual cortex (V1) in response to static grating stimuli [4]. The experimental protocol from the paper is provided in Figure 6 [4]. The dataset consists of a sequence of orientation and $\mathcal{D} = \{\theta_i, \mathbf{r}_i\}_{i=1}^{P_{tot}}$ for static grating orientations $\theta_i \in [0, 2\pi]$. To analyze the data, we bin $[0, 2\pi]$ into $P = 200$ separate bins and average the responses which fall into each bin. After this procedure we are left with the response matrix $\mathbf{R} \in \mathbb{R}^{N \times P}$ where $N \approx 20,000$. We provide a plot of the singular value spectrum σ_k in (b). We see that the decay in the spectrum is like a power-law rather than exponential in k, indicating a high dimensional population code. We visualize the eigenfunctions in (c) obtained from full SVD Randomized SVD for extracting the top singular

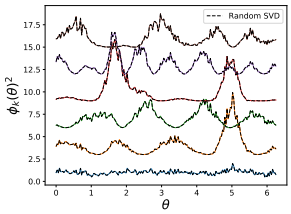
vectors and singular values. For $K = 50$, the time is reduced from 180 seconds to 0.1 seconds with a cost to relative reconstruction error of approximately 0.1.



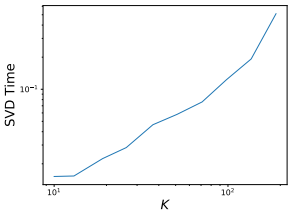
(a) The experimental protocol from (cite) Pachitariu et al



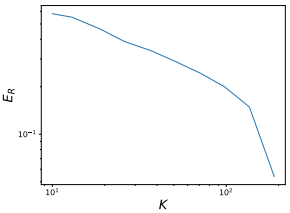
(b) Experimental Singular Values



(c) Experimental Eigenfunctions



(d) Time for Randomized SVD



(e) Relative Reconstruction Error

Figure 6: (a) Experimental responses of Mouse primary visual cortex (V1) are obtained from fluorescent microscopy. (b) The full SVD for a 200 sample grid of θ values, can be approximated by a rank 50 randomized SVD problem. This reduces time from ≈ 180 seconds to ≈ 0.1 seconds. (c) The recovered eigenfunctions from the rank 50 approximation (dashed black lines) are remarkably accurate compared to the ground truth (colors). (d) The time required for randomized SVD on this dataset scales in a power with K. The relative reconstruction error for randomized SVD with $K = 100$ is about $\sim 10^{-1}$, indicating a slow decay in the spectrum.

7 Application to Foreground-Background Segmentation

7.1 Background and Motivation

SVD has found application in the realm of video in foreground-background segmentation where the task is to split video into is a segmentation task, where the goal is to split the image into foreground and background. Also known as background subtraction, foreground-background segmentation plays an important role as a pre-processing step in change detection systems (e.g. video surveillance). In these systems foreground objects are often associated with moving entities in a relatively sparse background and in many cases are objects of interest. The static background hypothesis that underlies many classic approaches to background-subtraction can complicate the segmentation task when the background is dynamic (e.g. in windy or bad weather circumstances

or when the camera includes some jitter). Some common approaches to background subtraction in clude frame differencing, mixtures of gaussian and robust-PCA related methods. Recently Deep Learning methods have been applied to the background subtraction task.

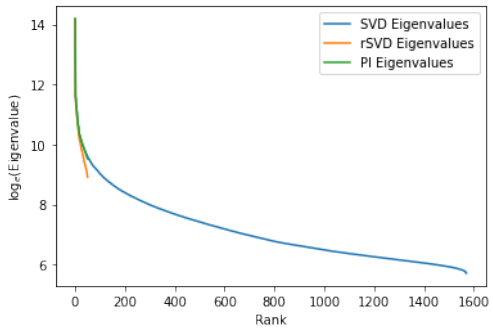
As described above, the aim of background subtraction in surveillance is to formulate the background according to an accurate model and then subtract the formulated background from every surveillance image leaving the foreground as a superset of objects of interest. We can represent the video as a sequence of images in which case we can take any set of the images, flatten them out and stack them into a matrix. Under the static background hypothesis we model the background as being static or at most change very slowly with time. In such a formulation we would intuitively expect the background to be present in almost all the image vectors and as such we expect the background to be a subspace of the column space of our matrix. In fact we would expect subspace corresponding to the background to have have high eigenvalues and be present in any low-rank approximation of the matix. This suggests the approach for background substitution that we'll be taking in this work – use SVD to generate a low rank approximation for the matrix formed by flattening stacking the the still frames of surveillance footage. Use the low-rank approximation as a representation of the background and the matrix formed by subtracting the low rank approximation from the original matrix as a representation of the foreground. One can imagine that a video with many stills (e.g. from long footage) recording in high resolution might result in a matrix for which traditional truncated SVD might be intractable. We'll take the approach of replacing the standard SVD operation with a randomized variant and compare the results.

7.2 Dataset

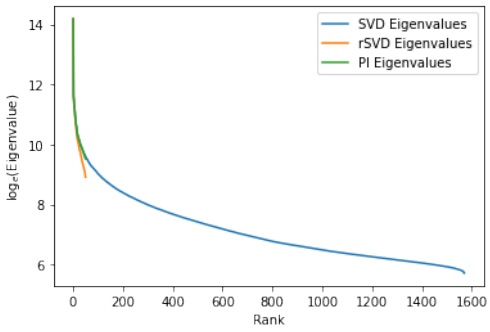
We chose videos from the 2014 dataset from changedetection.net. The videos in that dataset were curated to be representative of the challenges of outdoor surveillance. As such the videos are divided into categories representing situations such as camera jitter, night videos, bad weather, turbulence and shadow. We selected a baseline video on which to do our analysis as well as additional videos representing jitter and bad weather in order to understand the behaviour of our approach in videos not representing exactly the static background hypothesis. All in all we used four videos in our analysis – 1 from category baseline, 1 from category bad weather, and 2 from category jitter.

7.3 Analysis

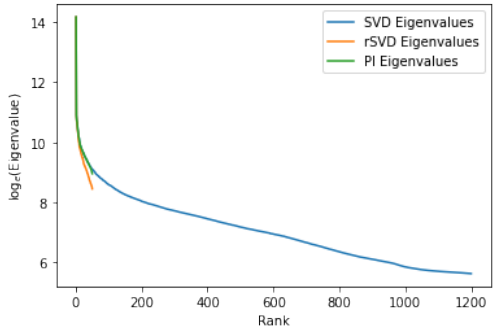
In 7 we see that the eigenvalue distribution for each of the positions is almost ideal for using the basic random SVD algorithm. Eigenvalues fall off logarithmically and the low-rank approximation captures much of the behaviour of M even with a relatively small number of eigenvalues. We generate the low rank approximation for each video with standard svd (using the numpy native formulation), with a hand implemented randomized SVD algorithm and with the power iteration version of SVD using 3 power iterations. One thing to note is that even for such a favorable eigenvalue distribution, close ot the regime of the low rank cut off the plain randomized SVD algorithm starts to more poorly approximate the full eigenvalue distribution whereas the power iteration version closely matches the distribution to the duration of the rank cut off.



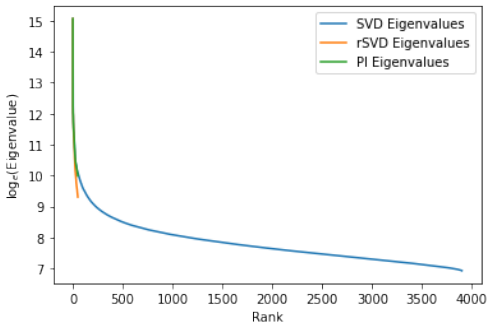
(a) Eigenvalue Distribution Road Video



(b) Eigenvalue Distribution Badminton Video



(c) Eigenvalue Distribution Room Video



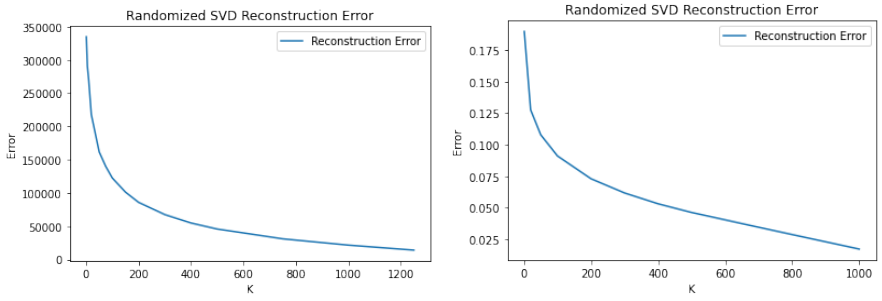
(d) Eigenvalue Distribution Skate Video

Figure 7: The eigenvalue distribution of each video falls off rapidly suggesting that standard random SVD algorithm should be sufficient for the low rank approximation

In Figure 8 we see the relative reconstruction error rates of the basic randomized SVD algorithm for reconstructing the matrix of flattened frame stills. We see that the low rank regime has a

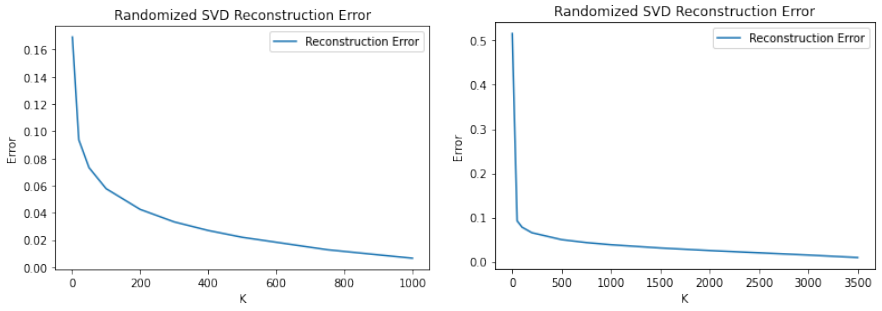
relatively low error rate for representing the whole matrix for three out of the four cases, but in the skating video there's a very high relative error rate for lowest rank approximations. Because of this we use a subspace of size 3 in for the low rank approximation for that case and 2 for the other cases.

The randomized SVDs performed identically to the truncated standard SVDs in the low rank regime of our background subtraction. We generated foreground and background movies based on the reconstruction of the randomized SVD (but not the power iteration although we have every reason to believe based on that algorithms eigenvalue ranks that it'd perform as well as the randomized SVD within the low-rank regime and better slightly outside of it) as well as the truncated full SVD. The differences between the foreground detection/background substitution for either algorithm are imperceptible and in fact a closer look shows they generate the exact same matrix reconstruction.



(a) Relative Reconstruction Error for Road Video

(b) Relative Reconstruction Error for Badminton Video

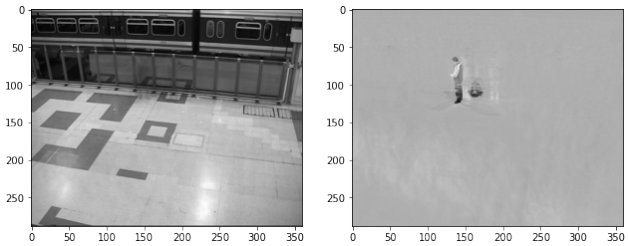


(c) Relative Reconstruction Error for Room Video

(d) Relative Reconstruction Error for Skate Video

Figure 8: The eigenvalue distribution of each video falls off rapidly suggesting that standard random SVD algorithm should be sufficient for the low rank approximation

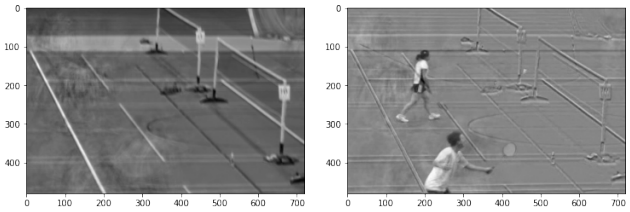
Examining the stills (or movies) generated from the the background subtraction process shows that both SVD algorithms successfully separated background and foreground figures. The SVD algorithms probably had the best performance on walking figures in a still room since that context best approximates the static background hypothesis.



(a) Room Still Background (b) Room Still Foreground

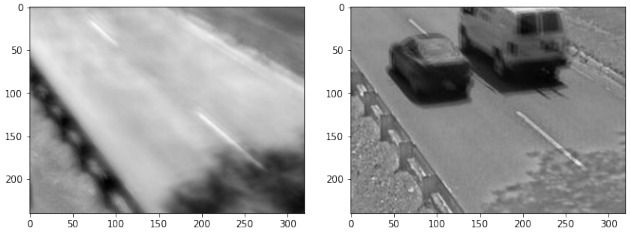
Figure 9: Background/Foreground Segmentation for single still of pedestrians walking in a room

The two videos (captive video of the road and of people playing badminton) taken with jitter also showed a relatively decent level of background-foreground segmentation although because of the camera motion the background didn't subtract quite as nicely as it did in the example of the room.



(a) Badminton Still Background (b) Badminton Still Foreground

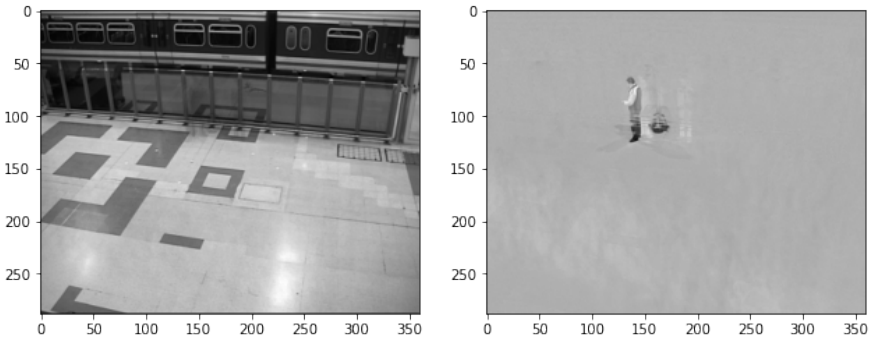
Figure 10: Background/Foreground Segmentation for a single still from video of badminton games take with jitter



(a) Road Still Background (b) Road Still Foreground

Figure 11: Background/Foreground Segmentation for a single still of a traffic'd road take with jitter

Finally in the case of skaters skating along a driveway in the snow, we see that while there was background subtraction, the motion of the snow caused it to be considered part of the foreground instead of the background.



(a) Skaters Still Background (b) Skaters Still Foreground

Figure 12: Background/Foreground Segmentation for a single room still

As far as SVD performance, as expected Standard SVD was consistently significantly (up to

orders of magnitude slower) than basic randomized SVD which was somewhat faster than Power Iteration.

context	(sec)	Metrix dim.	Std SVD (sec)	rSVD (sec)	Power Iteration(sec)
traffic	62 s	(76800, 1570)	62 s	3.59s	10.2 s
badminton	46 s	(345600, 1150)	106 s	6.03s	17.4 s
room	47 s	(103680, 1200)	24.9s	1.51 s	5.01 s
snowy lane	154 s	(48600, 3900)	124 s	2.67 s	7.01 s

Table 3: Table comparing time needed to compute standard SVD (np.linalg.svd), randomized SVD ($K = 50$) and Power Iteration ($K = 50, q = 3$) as part of background substitution for videos

8 Application to audio watermarking

8.1 Background and motivation

Audio watermarking is the technique of embedding or hiding a piece of information inside an audio file. The applications of this are widespread, ranging from covert message transmission to copyrighting music. There are a number of techniques that have been developed, with the goal of slightly amending the time or spectral components in such a way that the change is not picked up by humans. These techniques have varying degrees of robustness against audio manipulation, with an extensive evaluation produced by Stephan Wiefling in 'Comparison of Audio Watermarking Techniques' [5].

One such group of techniques, is the SVD based watermarking approach. The key idea is to embed a watermark in the singular values of the audio signal after transforming it into a 2D format and conducting singular value decomposition. One can select from a number of algorithms to be used for the encryption. In 'An SVD-Based Audio Watermarking Technique' Hamza Özer, Bülent Sankur and Nasir Memon used STFT (short time Fourier transform) to convert the signal into matrix form, obtained its singular values (using SVD) and modified them using coded binary sequences and a pseudo-random signal [6].

As discussed previously, SVD can be a time consuming process, especially for larger matrices. In Table 4 below we summarise the sizes of the time-frequency encoded matrices representing audio

files of various lengths, along with the time take to compute the SVD of these.

Length (s)	Metrix dim.	Time for SVD
10.4167	(129,2232)	0.9217719480002415
20.8333	(129,4464)	2.5760563550065854
31.2500	(129,6696)	6.638945705002698
41.6667	(129,8928)	13.58328325200273
52.0833	(129,11160)	13.483091465001053
62.5000	(129,13392)	22.085866300993075
72.9167	(129,15624)	37.425949323005625
83.3333	(129,17857)	88.62662779100356
93.75	(129,20089)	629.1455800450058

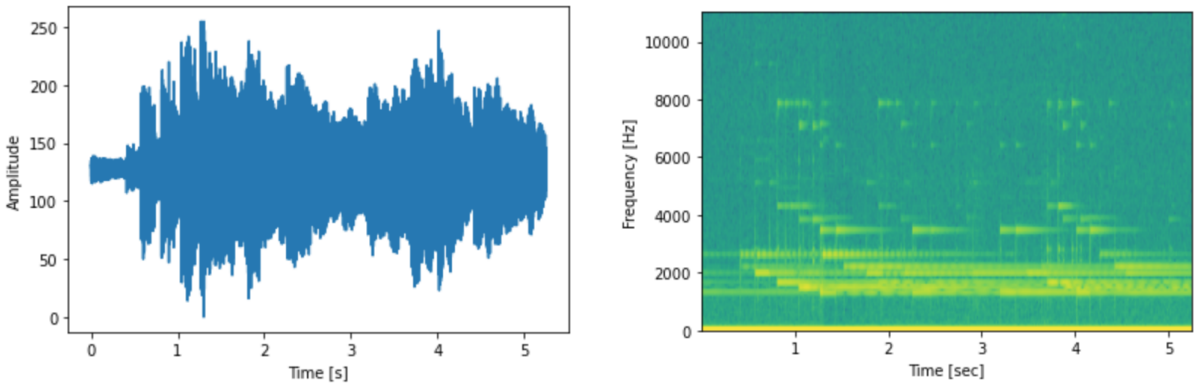
Table 4: Table comparing the time taken to compute SVD for recordings of various lengths and their vector format dimensions.

This motivates our investigation into potential ways to speed up SVD using the standard random SVD and power methods discussed previously. Since in this application, accuracy is also important we investigate the time taken for each of these methods to reach various accuracy thresholds.

8.1.1 Spectrograms

There are various ways to mathematically and visually represent sound. Perhaps the most common is the waveform representation, as shown in Figure 13 a), where the amplitude of the recording is plotted over time. However, these fail to capture other aspects of the sound, for instance frequency, leading to the use of spectrograms. Spectrograms are 2D heat maps of sorts: with the time on the x-axis, frequency on the y and the amplitude being represented by the heat map colouring. There are two ways to produce a spectrogram from a time-domain signal. The first is by approximating it as a filter bank, which is an array of band-pass filters separating the audio signal into a series of sub-components each with a single frequency sub-band of the audio signal. The second, and the approach we used, is calculating it using Fourier transforms. Here the data is broken up into chunks, with some overlap, and Fourier transforms are applied to calculate the magnitude of the

frequency spectrum for each overlapping chunk. Each chunk then corresponds to a vertical line in Figure 13 b).



(a) Plot of amplitude against time

(b) Spectrogram of audio file

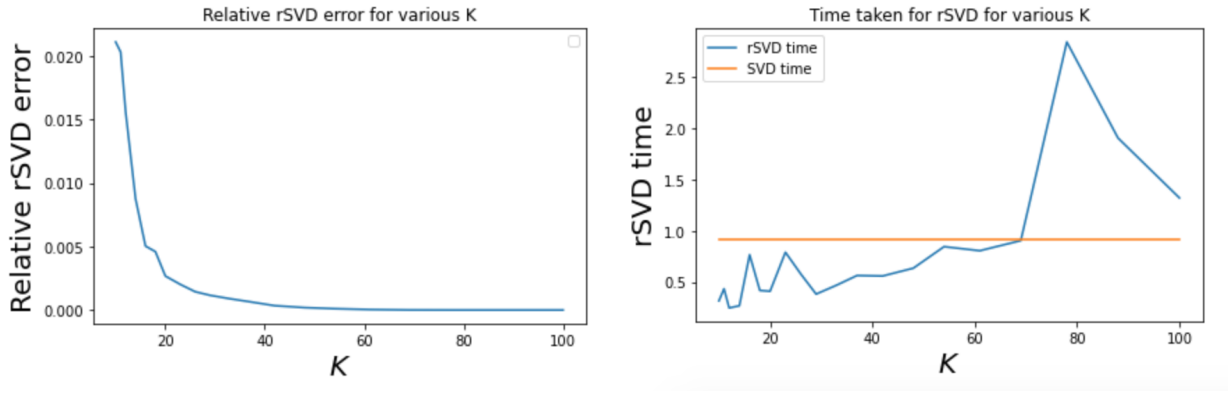
Figure 13: Visual representation of the audio in the windchimes.wav file (5s long). Plot a) shows the waveform representation of the sound and plot b) shows the spectrogram of the audio file.

8.2 Computational comparison of methods

In this section, we conduct our analysis on the audio from the AM205 2.12 low rank approximation lecture video available on YouTube. We take incrementally larger segments of the audio clip to conduct the analysis. For consistency, only this audio clip will be discussed in this section, however some plots of the analogous analysis for various short sound clips are included in the accompanying code.

8.2.1 Standard rSVD

Figure 14 a) shows the relative accuracy of the rSVD decomposition for the spectrogram of the first 10.4s of the audio file. We can see that a K value of around 40 is required to reach appropriate accuracy. Interestingly, from 14 b) we can also see that rSVD is only faster for small values of K.



(a) Relative error of the rSVD decomposition for various K.

(b) Time taken to compute rSVD decomposition for various K, compared to the SVD.

Figure 14: Comparison of the time and accuracy of rSVD method for various values of K conducted on the first 10.4s of the AM205 2.12 low rank approximation audio file. Plot a) shows that a K value of around 40 is needed to reach sufficient accuracy. Plot b) shows that rSVD only shows an improvement in speed for small values of K.

In Table 5 we can see the comparison of the time taken to compute SVD and rSVD for $K = 40$ for various lengths of audio. From the table you can see that for longer audio clips rSVD becomes faster than standard SVD (around 3 times faster for the audio clip of length 93.75s), however for greater improvements we look to the power method.

Length (s)	Metrix dim.	Time for SVD	Time for rSVD
10.416666666666666	(129, 2232)	0.9217719480002415	1.08539553200535
20.833333333333332	(129, 4464)	2.5760563550065854	3.148388847002934
31.25	(129, 6696)	6.638945705002698	6.8295845609973185
41.666666666666664	(129, 8928)	13.58328325200273	15.775480446995061
52.083333333333336	(129, 11160)	13.483091465001053	26.246498020998843
62.5	(129, 13392)	22.085866300993075	43.432488842001476
72.91666666666667	(129, 15624)	37.425949323005625	64.16156991600292
83.33333333333333	(129, 17857)	88.62662779100356	143.78583880200313
93.75	(129, 20089)	629.1455800450058	267.71861250000075

Table 5: Table comparing time taken to compute rSVD for $K = 40$ for various lengths of audio clips.

In Figure 15 we also plot the approximate spectrogram, resulting from multiplying out the components of rSVD for smaller values of K.

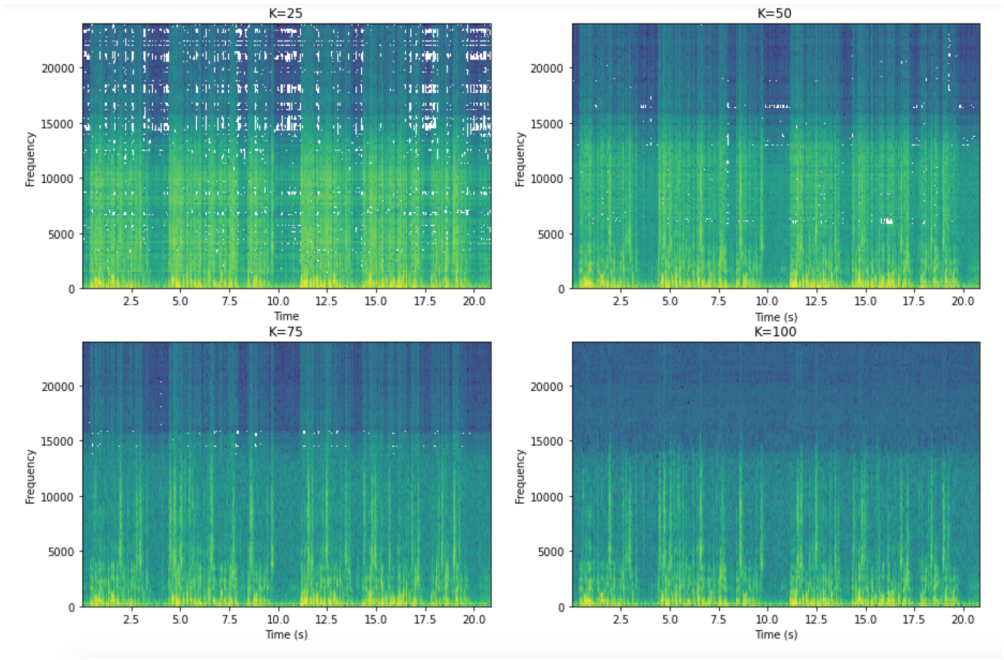
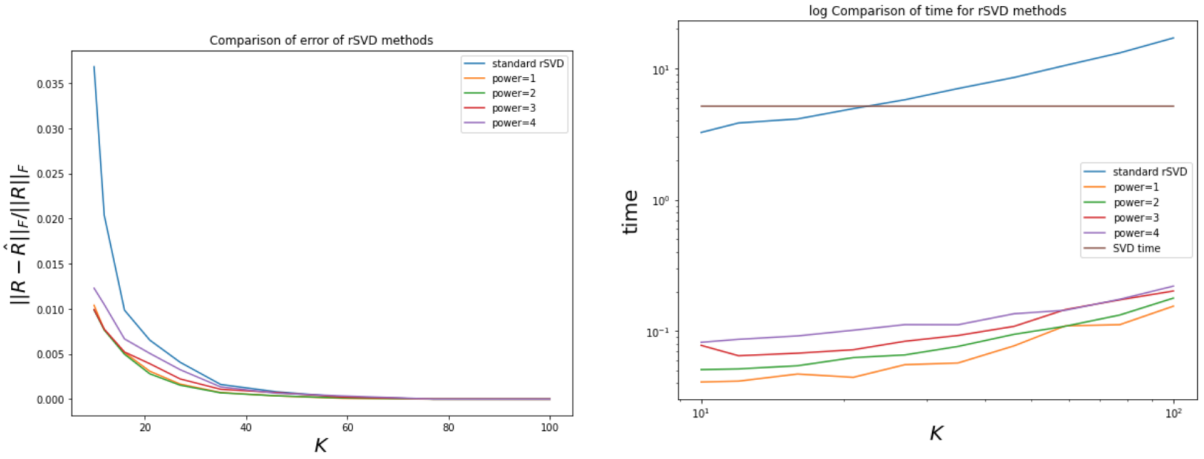


Figure 15: Approximate spectrograms generated from rSVD decomposition using various values of K=(25,50,75,100) for a 20.83s clip of audio. For smaller values of K, especially at higher frequencies rSVD doesn't prove to be very accurate, with great improvement seen for K=100.

8.2.2 Power method

We next compare the accuracy and the time taken to decompose the spectrogram matrix using the power method with $p = 1, ..., 4$ for a clip of the audio of length 41.67s. In Figure 16 a) we can see the relative error for the 4 power methods and rSVD. The power methods converge at small values of K indicating that the results for the various power would be comparable and that a lower power of $p = 2$ would be sufficient. Also in 16 b) we see that the power methods offer a much greater speed up than the standard rSVD method, showing that even if the chosen value of K is large (100) the method still shows significant time improvement over standard SVD.



(a) Relative error for rSVD and power methods

(b) Time taken for decomposition using rSVD and power methods

Figure 16: Comparison of the SVD, rSVD and power methods with p=1,...,4. Analysis conducted for audio snippet of length 41.67s. Plot a) shows that the relative error becomes sufficiently small at around $K = 35$ for the power methods and $K = 45$ for the standard rSVD method. Plot b) shows a log-log plot of the time taken for decomposition using the 5 methods. The plot shows significant speed up for the power methods compared to SVD with the rSVD method taking longer than standard SVD for $K > 40$

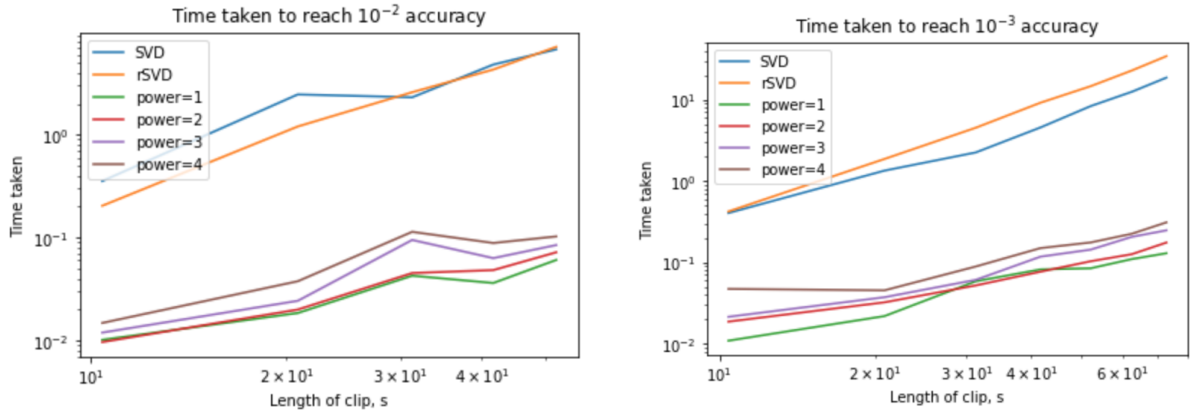
We present our results for the power method with $p = 2$ and $K = 40$ in comparison to the standard SVD in Table 6 below. Once again, the comparison was conducted on sections of the Low Rank Approximation file and shows that the power method provides a significant speed up over rSVD and SVD methods.

Length (s)	Metrix dim.	Time for SVD	Time for rSVD	Time for power
10.416666666666666	(129, 2232)	0.9217719480002415	1.08539553200535	0.04180979400007345
20.833333333333332	(129, 4464)	2.5760563550065854	3.148388847002934	0.07917833999999857
31.25	(129, 6696)	6.638945705002698	6.8295845609973185	0.07551688300009118
41.666666666666664	(129, 8928)	13.58328325200273	15.775480446995061	0.07972547600002144
52.083333333333336	(129, 11160)	13.483091465001053	26.246498020998843	0.12541791200010266
62.5	(129, 13392)	22.085866300993075	43.432488842001476	0.17302624000012656
72.91666666666667	(129, 15624)	37.425949323005625	64.16156991600292	0.17785789400022622
83.33333333333333	(129, 17857)	88.62662779100356	143.78583880200313	0.201802157000202
93.75	(129, 20089)	629.1455800450058	267.71861250000075	0.2046934599998167

Table 6: Table comparing the times for SVD and rSVD power method with $p = 2$ and $K = 40$ for audio files of varying lengths. The table shows that the power method offers a signigicant speed up in comparison to SVD and rSVD.

8.2.3 Method comparison for various recording lengths

So surmise, we present a more concrete comparison of the execution times of each of the methods when working to the same degree of accuracy. In Figure 17, for each of the 5 methods (rSVD and power method with p=1,..,4) we compute the value of K required to reach the accuracy of a) 10^{-2} and b) 10^{-3} and record the time taken for the decomposition for said K. We repeat the process for various clip lengths. Consistent with the analysis above, we find that in particular for larger audio clips the power method decomposition offers a significant speed up compared to SVD, with the rSVD method offering little (for longer files in a) 10^{-2}) or no speed up when K is taken to be large enough to offer high enough accuracy.

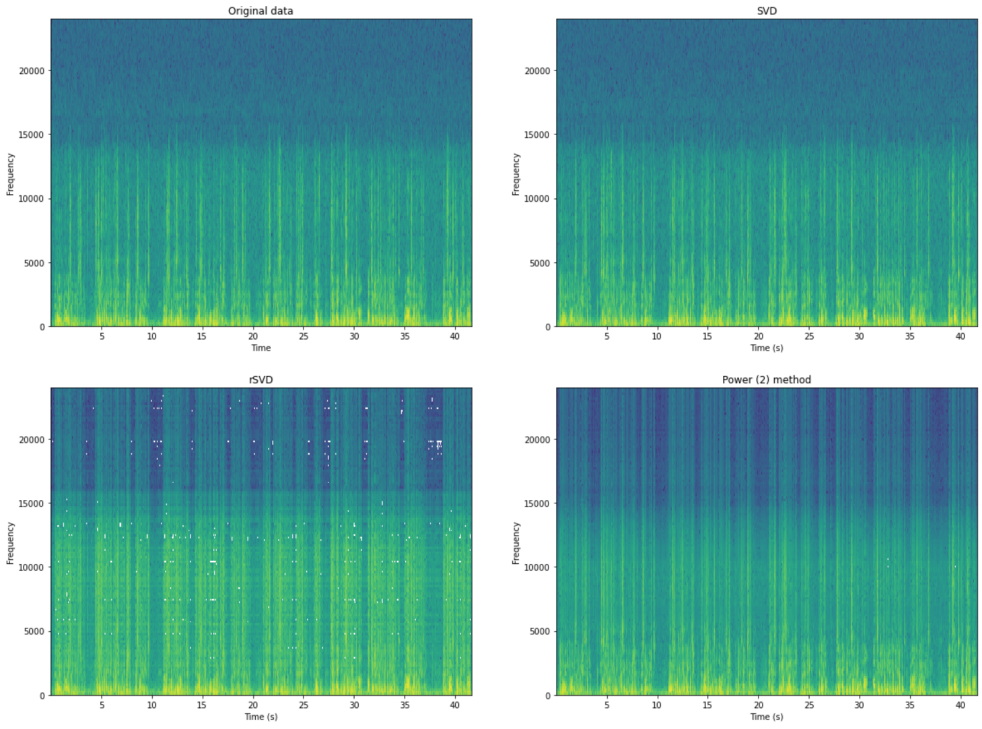


(a) Decomposition to accuracy of 10^{-2}

(b) Decomposition to accuracy of 10^{-3}

Figure 17: Comparison of the time taken for each of the 3 methods to reach the accuracy of a) 10^{-2} b) 10^{-3}. In both cases, rSVD shows little to no speed up compared to SVD. The power methods show a significant speed up in both cases.

In Figure 18 we plot the spectrogram for a 20.83s extract of the Low Rank Approximation clip alongside the SVD, rSVD (for K=40) and power (for $p = 2$, K=40) decomposition approximations. The figure shows that as we would expect, SVD provides an exact decomposition. The rSVD and power 2 methods both capture the lower frequency region better than higher frequency with the power 2 method proving more accurate. As in previous analysis, the rSVD method took longer to run than standard SVD, showing that for this application rSVD does not provide the speed up we were looking for. The power 2 method was around 7 times faster than standard SVD, offering significant speed up.



(a)

Figure 18: Comparison of SVD, rSVD and power 2 rSVD spectrograms to the spectrogram for a 20.83 second extract from the Low Rank Approximation clip. The methods took: SVD = 27.837459806003608s, rSVD = 36.619808862000355s and power 2 rSVD= 4.054416326005594s seconds to run. The power method looks to be a good approximation, in particular for lower frequencies.

9 Conclusion

In this report, we analyzed the performance of randomized singular value decomposition on a variety of significant applications. We find that random projections provide a convenient and easily implementable method to reveal the low rank structure in common datasets, allowing rapid reconstruction of the top singular values and singular vectors. Using an embedding dimension of size $K < m, n$, we find, consistent with pre-existing theoretical results [2], that the time complexity of SVD on a $m \times n$ matrix improves from $O(mn^2)$ to $O(mnK + (m+n)K^2)$. In the real world applications we considered, neuroscience, image processing and audio water-marking, this random projection step can provide significant speedups. If the chosen embedding dimension K is smaller

than the true rank of the matrix, this random projection introduces a reconstruction error, which depends on the nature of the problem. For a fixed embedding dimension K, matrices with rapidly decaying σ_k generate the lowest reconstruction error. In the neuroscience example, we show how toy models can help generate domain knowledge which can be helpful when choosing the embedding dimension K. Power iteration can also provide time reductions, as we show in the audio watermarking experiments. Sparsity and structured matrix transforms can speed up the projection step of rSVD.

References

[1] M.Ramakrishna Vinita Vasudevan. A hierarchical singular value decomposition algorithm for low rank matrices, 2019.

[2] Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, 2010.

[3] Oleg Rumyantsev, Jérôme Lecoq, Oscar Hernandez, Yanping Zhang, Joan Savall, Radosław Chrapkiewicz, Jane Li, Hongkui Zeng, Surya Ganguli, and Mark Schnitzer. Fundamental bounds on the fidelity of sensory cortical coding. *Nature*, 580:1–6, 04 2020.

[4] Carsen Stringer, Michalis Michaelos, and Marius Pachitariu. High precision coding in mouse visual cortex. *bioRxiv*, 2019.

[5] Stephan Wiefling. Comparison of audio watermarking-techniques. 2016.

[6] Bülent Sankur Hamza Özer and Nasir Memon. An svd-based audio watermarking technique. 2005.