IMGPCA
A DATA VISUALIZATION TOOL FOR PCA ANALYSIS
MAI718 3/2017 FINAL PROJECT

Roberto Stelling *
PPGI - 117.335.792
Universidade Federal do Rio de Janeiro
Rio de Janeiro, RJ 21941-916, Brazil
roberto@stelling.cc

ABSTRACT

Traditionally PCA Analysis for dimension reduction in images is performed without a systematic visual aid approach. This paper describes a data visualization tool for PCA Analysis implemented in JavaScript aiming to improve the analyst decision process.

1 PROBLEM DESCRIPTION

There are many reasons to reduce a data set, some examples are: noise reduction, outlier removal, lossy image compression or even as a preliminary step in various types of data exploration and data analysis.

Principal component analysis (PCA) can be used as a lossy image compression solution, as you can apply PCA on a set of images of a data set and reduce its dimensionality with a certain, desirably controllable, loss of precision. Of course, one want to loose as little precision as possible while compressing as much as possible.With images, the main difficulty resides in the trade off between compression and precision: how many dimensions can be thrown away making sure that the resulting images still retain the desired level of quality or sharpness? Is there a general rule where you can certainly decide how many dimensions will be cut off the original data set? Of course, the type and amount of data available for compression, the data set, has a big influence on the precise point where the cut will happen, but can the analyst decide simply on the number of dimensions or amount of variance that will be thrown away and be sure that the results will be satisfactory? We propose that using a visual helping tool during the decision process can have a positive impact on the cut off selection.

2 BRIEF INTRODUCTION TO PCA

2.1 OBJECTIVE

According to Jolliffe (1986), the central idea of principal component analysis (PCA) is to reduce the dimensionality of a data set in which there are a large number of interrelated variables, while retaining as much as possible of the variation present in the data set.

This reduction is achieved by transforming the data set into a new set of variables, the principal components, which are not correlated, and which are ordered so that the first few retain most of the variation present in all the original variables.

2.2 BRIEF HISTORY OF PCA

Principal component analysis was first described by Pearson (1901) and later developed independently by Hotelling (1933) (Jolliffe, 1986).

*PPGI/UFRJ Graduate Student, http://stelling.cc.

Preisendorfer & Mobley (1988) state that in 1873, the Italian geometer Beltrami, formulated a modern form of the resolution of a general square matrix into its singular value decomposition (SVD), the decomposition that stands at the base of PCA.

Craw & Cameron (1992) describe a method for face recognition using principal component analisys, showing that PCA can be used as an effective tool in image analysis.

2.3 INTUITION

PCA can be thought of as the problem of fitting an n-dimensional ellipsoid to the m-dimensional data, where $n <= m$ and each axis of the ellipsoid represents a principal component. The larger the axis of a component, the larger the variance for that component. So, the objective of PCA is to build a transformation of m-dimensional space to n-dimensional space while preserving most of the the m-dimensional space variance. To find that transformation and the components, we compute the singular value decomposition of the data. The singular value decomposition will provide a computationally efficient method of finding the principal components and the scaled versions of the principal component scores.

2.4 SINGULAR VALUE DECOMPOSITION

Given an arbitrary $D_{m\times n}$ matrix, then D can be written as

$$D_{m\times n} = U_{m\times r} S_{r\times r} V^T_{r\times n} \tag{1}$$

where

(i) U and V, each of which with orthonormal[1] columns so that $U^T U = I_r$, $V^T V = I_r$;

(ii) S is a diagonal matrix;

(iii) r is the rank[2] of D.

S is a diagonal matrix such as:

$$S = \begin{bmatrix} s_1 & & \\ & \ddots & \\ & & s_r \end{bmatrix}$$

Where s_1 to s_r are the principal components scores and $s_1 >= s_2 >= ... >= s_{r-1} >= s_r$.

U is the eigenvector matrix, with eigenvectors ordered by the component scores, their eigenvalues.

2.5 PRINCIPAL COMPONENT ANALYSIS

To apply PCA we will use equation (1) from the singular value decomposition and apply it to the covariance matrix Σ of a data set. If the data set D has dimensions $m \times n$ then the covariance matrix Σ will be a symmetrical square matrix of dimensions $n \times n$. So

$$\Sigma_{n\times n} = U_{n\times n} S_{n\times n} V^T_{n\times n} \tag{2}$$

Where

- U is orthonormal and holds Σ eigenvectors
- S is a diagonal matrix with $s_1 ... s_n$ as the descending ordered eigenvalues.

As U is orthonormal then we can transform the original D data set into P

$$P_{m\times n} = D_{m\times n} U_{n\times n}$$

and restore it back with:

$$PU^T = DUU^T = DI_n = D$$

P is a transformation of D that retains all the information of the original data set.

[1] both orthogonal and normalized

[2] corresponds to the maximal number of linearly independent columns of D

3 PCA FOR IMAGE COMPRESSION

A computer image is usually thought of as a two dimensional matrix, with m lines and n columns representing the horizontal and vertical pixels of the image. A simpler, albeit equally meaningful, representation is a single vector with $m \times n$ cells for the whole image. The content of each cell, in either representation, depends on the selected image mode. For example: RGB, RGB grayscale, CMYK, etc. For the purposes of the following argument and the solution implementation, we assume that each cell is an integer between 0 and 255, representing the grayscale RGB value of the corresponding pixel. We will also assume that our data set has m samples of n pixels; if h is the number of horizontal pixels and v is the number of vertical pixels, then $n = h \times v$.

Lets assume that D is a data set with m images where each image has n pixels. Computationally the method can work even if $m < n$ but it is recommended that $m >= n$, as that will result in better compression gains and finer eigenvector tuning.

Given that $D_{m\times n}$ is a data set with m data points $\in \mathbb{R}^n$ where $m >= n$. Then we define $D^*_{m\times n}$ as the normalized data set,

$$D^* = \frac{D - \overline{D}}{s}$$

where $\overline{D}$ is the mean of D and s is the sample standard deviation of D. Let $\Sigma_{n\times n}$ be the covariance matrix of $D^*_{m\times n}$

$$\Sigma = \frac{1}{m} D^{*T} D^* \tag{3}$$

Then, according to (1) and (2), the singular value decomposition of Σ is:

$$\Sigma = USV^T \tag{4}$$

where:

- U is an $n \times n$ orthonormal matrix
- S is an $n \times n$ diagonal matrix with non-negative numbers on the diagonal
- V is an $n \times n$ unitary matrix and V^T is V transposed.

3.1 REDUCING DIMENSIONS

Given the original data set, $D_{m\times n}$ and $U_{n\times n}$ obtained from Σ as per (4), then we can build a new reduced data set $P_{m\times k}$ with the first $k < n$ eigenvectors. This new data set is computed as:

$$P_{m\times k} = D_{m\times n} U_{n\times k} \tag{5}$$

where $U_{n\times k}$, or U_k, is the eigenvector matrix truncated to the first k eigenvectors. The information contained on the truncated $(n - k)$ columns is lost in this transformation.

3.2 RESTORING DATA

The transformation in (5) is lossy, meaning that the information on the truncated $n - k$ columns is utterly lost during the transformation. Although it is possible to restore P back to D dimensions, the result will not be exactly D but rather an approximation of D. The whole rationale of using PCA for image compression is that the last $n - k$ eigenvectors will hold as little variance as possible and the recovered images will be an acceptable approximation of the original images. To transform P back into D space we compute

$$P_{m\times k} U^T_{k\times n} = D_{m\times n} U_{n\times k} U^T_{k\times n} \approx D_{m\times n}$$

4 HOW TO SELECT THE NUMBER OF COMPONENTS TO RETAIN

The problem of selecting how many components to retain is not new, Zwick & Velicer (1986), present the results of a Monte Carlo evaluation of five methods that have been proposed for determining how many factors or components to retain: Horn's parallel analysis, Velicer's minimum

average partial, Cattell's scree test, Bartlett's chi-square test, and Kaiser's eigenvalue greater than 1.0 rule. The determination of the number of components or factors to retain is likely to be the most important decision a researcher will make (Zwick & Velicer, 1986).

We propose that a graphical supporting tool, with a graph similar to Cattell's scree plot but displaying $\log_{10} eigenvalues$ instead of $eigenvalues$, plus on the fly representations of a subset of the compressed images, and a subset of the eigenimages, can be instrumental in the decision of how many dimensions must be retained. We suggest that the use of eigenimages can increase the analyst understanding of the variation and trends of main eigenvectors of the data set.

The number of dimensions to retain will eventually be a decision based on the supporting graphs and the recovered images. We include a couple of numeric measures on the cut off point:

- Accumulated variation retained.
- Number of components retained.
- Component score on the cut off point, the cut off point eigenvalue.

If we were to use a rule similar to Kaiser's Rule[3], based on our experience with the proposed tool with 32×32 grayscale images, then we would suggest a $\frac{1}{10}$ Kaiser's Rule: eigenvalues greater than 0.1. Trials with a few data sets suggest that a "*One Tenth Kaiser's Rule*" with eigenvalues > 0.1 is a reasonable trade off between compression and sharpness.

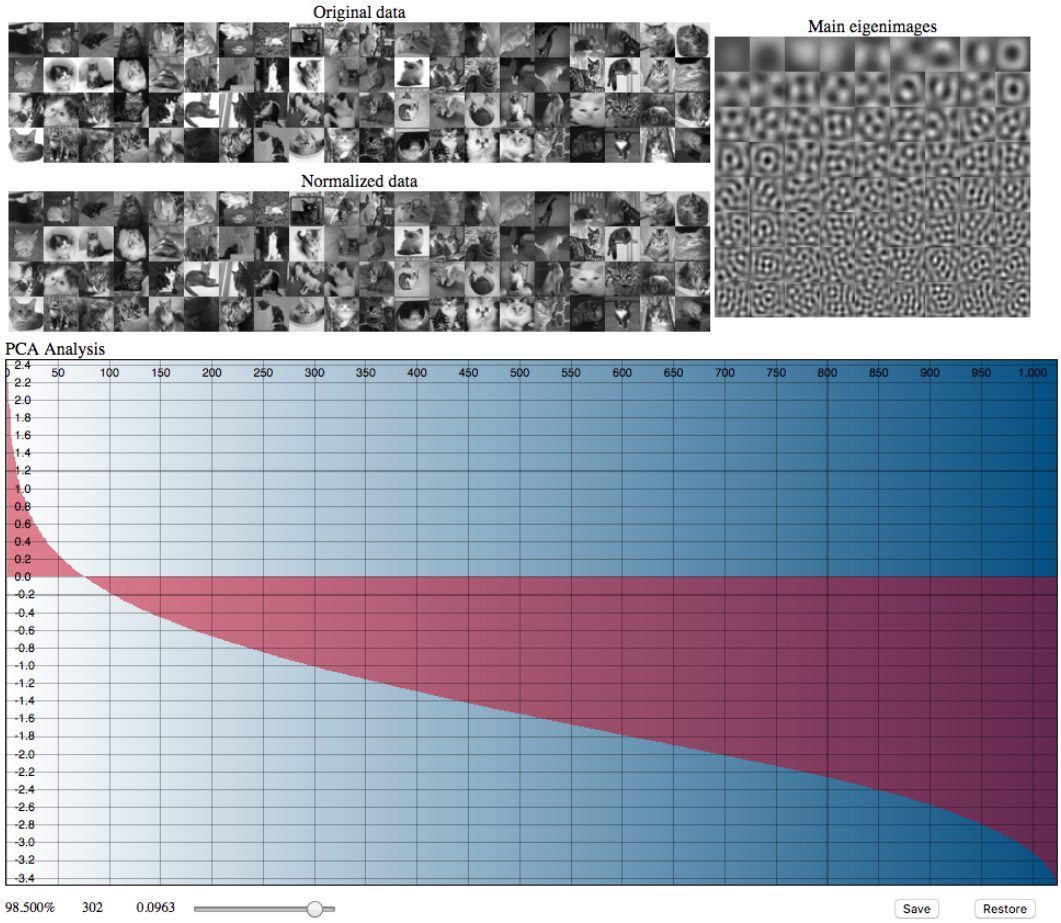


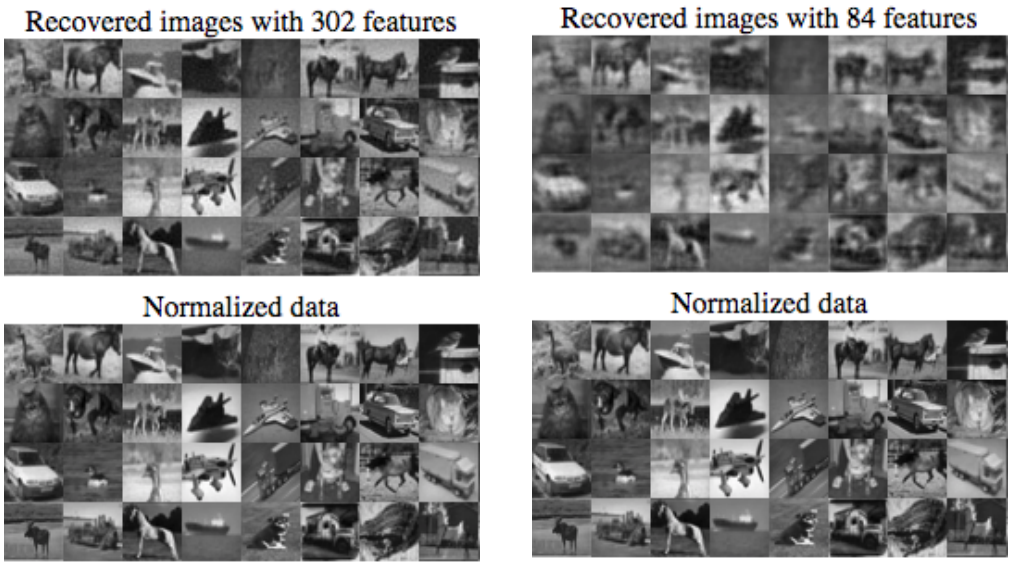
Figure 1: ImgPCA screenshot with a typical log_{10} *eigenvalues* $\times$ *#of components* curve.

[3]Eigenvalues greater than one

Note that the Kaiser's Rule cut occurs exactly where the graph crosses 0 ($\log_{10} eigenvalue = 0$) and that our "*One Tenth Kaiser's Rule*" occurs when $\log_{10} eigenvalue = -1$.

5 IMGPCA

For the next few examples we are going to use a subset of CIFAR-10 (Krizhevsky & Hinton (2009), chapter 3) images modified to fit our purposes. CIFAR-10 is a collection of 32×32 color images that are available in python, Matlab and binary versions. We converted CIFAR-10 images to grayscale (see Materials and Methods) and rotated the images 90 degrees to produce the input files for our usage.



(a) 302 components (b) 84 components

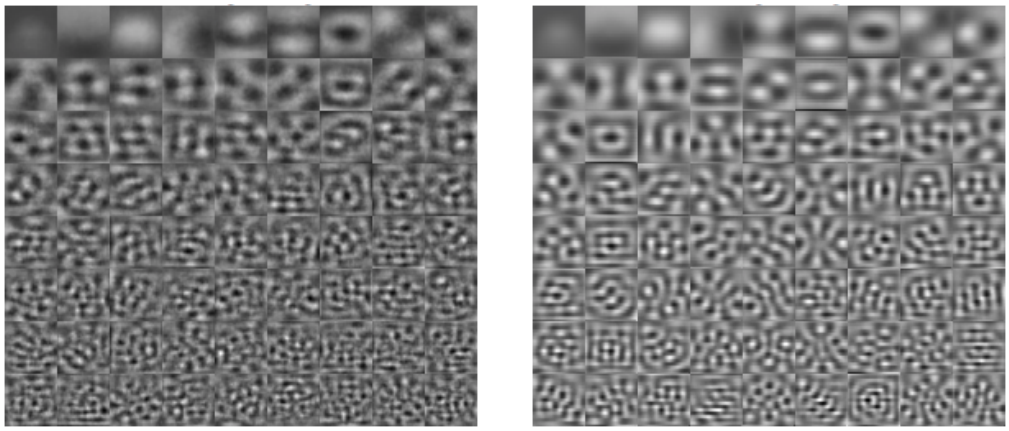
Figure 2: CIFAR-10 images recovered using "*One Tenth*" (2a) and regular Kaiser's rule (2b).

Figure 2 shows recovered images from a data set of 1024 images with 1024 (32×32) pixels of all CIFAR-10 classes, in the order they appear on the original file. Figure 2a shows the images recovered with 302 components and figure 2b shows the images recovered 84 components. The 302 components represent 98.789% of variance, with the smallest component score being 0.1009, following our "*One Tenth Kaiser's Rule*". There are a few artifacts in some of the recovered images but the overall result is visually satisfactory. The 84 components example on Figure 2b follows Kaiser's Rule of keeping eigenvectors with $eigenvalues > 1.0$. It is clear that cutting at this point, for our particular application, produces blurred images when dimensions are restored.

Table 1 shows the stats when we apply the *One tenth Kaiser's rule* to two datasets of different sizes. It is important to notice that all images of the first data set are included on the second. If we increase the number of images on the data set, from 1024 to 15000 images then we notice a few changes on ImgPCA statistics display, the most notable visual difference is the definition and ordering of the eigenimages.

Table 1: Measures for CIFAR-10, all classes, at *One tenth Kaiser's rule*

MEASURE	**1024 IMAGES**	**15000 IMAGES**
Features included	302	341
Accumulated variance	98.789%	98.330%
Last eigenvalue included	0.1009	0.1008

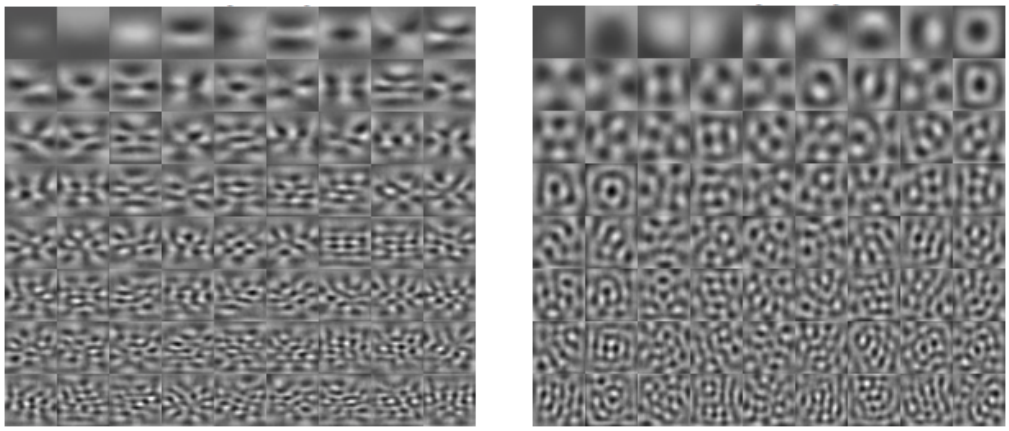


(a) From 1024 images (b) From 15000 images

Figure 3: Eigenimages generated from 1024 and 15000 images of all CIFAR-10 classes.

5.1 EIGENIMAGES

Figure 3 shows the first 72 eigenimages generated from data sets of size 1024 (Figure 3a) and 15000 (Figure 3b) images of all CIFAR-10 classes. The eigenimages reveal some characteristics of the images on the data set: centrality of subject, dark central subject on a clear image, clear central subject on a dark image, left to right symmetry, bottom to top symmetry among others. It is not surprising that the ordering of the eigenimages changes when we increase or decrease the number of images of our test sample but the improvement of eigenimage definition, contrast and sharpness when the size of the data set increases is striking.



(a) Class 0 - Planes (b) Class 3 - Cats

Figure 4: Eigenimages generated from 5000 images of two different classes.

The CIFAR-10 subset used to produce the displays in Figure 3 includes images from all 10 categories (numbered from 0 to 9) and has as much variance as possible given the chosen subject of the data set.

In Figure 4 we can see the eigenimages generated from 5000 images of CIFAR-10 class 0 (4a) and CIFAR-10 class 3 (4b). Not only the eigenimages are different but one can almost see vestigial planes on 4a and vestigial cats on 4b, the subject of their corresponding CIFAR-10 classes. This effect, applied to faces, or eigenfaces, was observed and used by Sirovich & Kirby (1987).

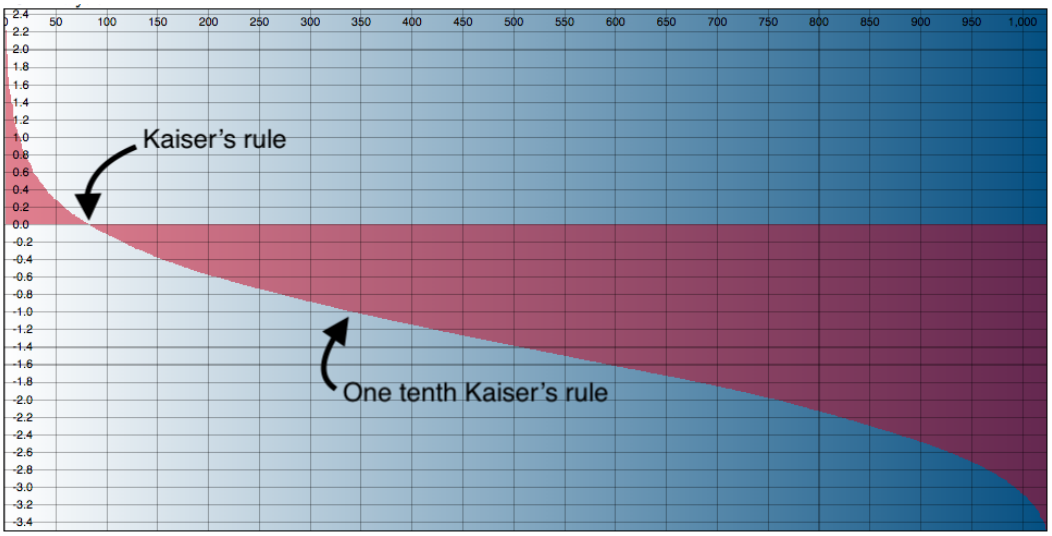


Figure 5: ImgPCA plot with Kaiser's Rule and *One tenth Kaiser's rule* indicated.

5.2 IMGPCA PLOT

Instead of plotting an *eigenvalues* $\times$ *# of components* graph, as in the regular Cattell's scree plot, we propose plotting $\log_{10}$ *(eigenvalues)* $\times$ *# of components*. Similar to the scree plot, we can derive some understanding about the data based on how fast the $\log_{10}$ curve decays but we also have a quick visual and mathematical way of finding the Kaiser's Rule, that occurs when $\log_{10}$ *(eigenvalues)* $= 0$. Additionally, every one tenth of eigenvalue decrease can also be easily found, as they correspond to the points $0, -1, -2$ etc. For example, to apply our *One tenth Kaiser's rule* we only need to check what are the number of components that correspond to $\log_{10}$ *(eigenvalues)* $= -1$, as shown in Figure 5.

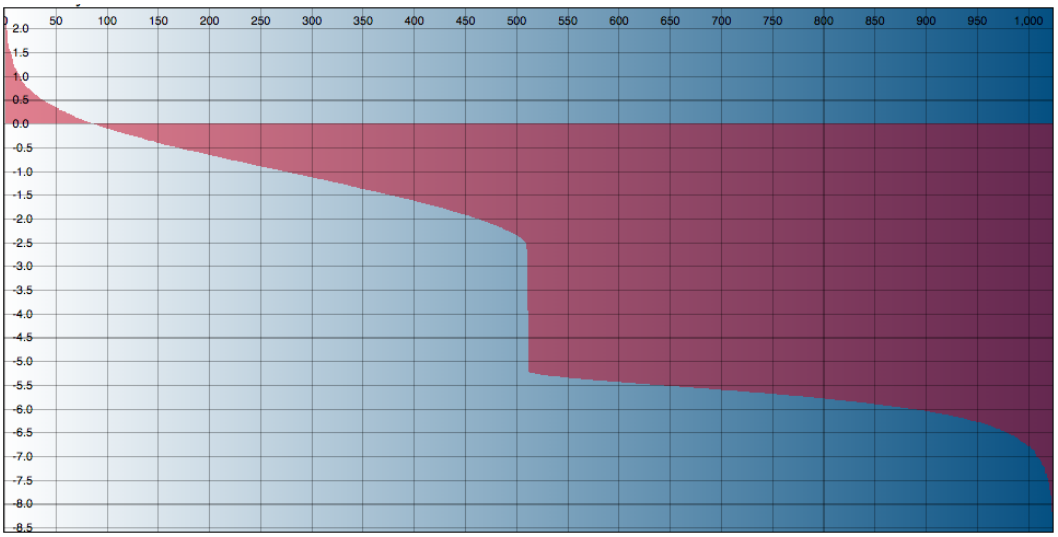


Figure 6: ImgPCA plot of a data set with 512 images of 1024 pixels.

Another feature of the log_{10} *eigenvalues* $\times$ *#of components* curve is that it is very easy to spot the point where data is missing from a data set. Figure 6 shows a plot of data set of 512 CIFAR-10 images. We see a big fall at 512 components, where the remaining components are not as significant as the first half. Compare Figure 6 with the typical curve of Figure 5.

6 MATERIALS AND METHODS

6.1 CIFAR-10 CONVERSION

All the data used in the text examples is available on Github at Stelling (2018), in the `/localData` folder. Images were taken from CIFAR-10 (Krizhevsky & Hinton, 2009) but were converted to grayscale and rotated for our usage. The conversion routine can be found at the `cifar2Gray.m` Octave file in the `/dataGen` folder of Stelling (2018) Github repository.

CIFAR-10 images were converted to grayscake with luminance signal coefficients following ITU-R Recommendation BT.709 (Union, 2015), as shown in the following Octave code excerpt:

```
gs_data=classData(1:nLines,1:1024)*0.2126 ...
      + classData(1:nLines,1025:2048)*0.7152 ...
      + classData(1:nLines,2049:3072)*0.0722;
```

and the resulting images were rotated 90 degrees clockwise, as per the following Octave code.

```
ndata(k,:) = ...
    reshape(rot90(reshape(gs_data(k,:), 32, 32), -1), 1, []);
```

The resulting data sets are used as input on ImgPCA web application.

6.2 SVD DECOMPOSITION

We used Numeric (Loisel, 2012) Javascript library to compute the SVD decomposition. Numeric is a stable numerical analysis library for Javascript. Computation results where in agreement with parallel computations performed in GNU Octave (Eaton, 2018). One area for improvement in ImgPCA would be the usage of a GPU-enabled library for the SVD decomposition, instead of Numeric.

6.3 COVARIANCE MATRIX

To accelerate the computation of the covariance matrix we used the GPU accelerated javascript library gpu.js (Sapuan & Plummer, 2017). To use gpu.js you need to create a kernel that will execute your code, as shown in the following JavaScript excerpt. The parameter for coVM a is the normalized data set matrix ($m \times n$). This kernel computes the covariance matrix, as defined by equation 3.

```
const coVM = getData.gpu.createKernel(function(a) {
  var sum = 0;
  for (var i = 0; i < this.constants.size; i++) {
    sum += a[i][this.thread.x] * a[i][this.thread.y];
  }
  return sum/this.constants.size;
}
```


ACKNOWLEDGMENTS

I'd wish to thank the whole of the MAI718/2017-3 class for their invaluable input during preliminary presentations of the current work. Special thanks go to our professor, Adriana Vivacqua, her ideas, the discussions she promoted and critiques to the preliminary versions where instrumental in improving the foundations of this work.

I'd specially like to thank PPGI/UFRJ for letting me use their computer laboratory for the duration of the 2017 term. Without that access I wouldn't have the means to work on this project with the intensity it required.

REFERENCES

Ian Craw and Peter Cameron. Face recognition by computer. In *BMVC92*, pp. 498–507. Springer, 1992.

John W. Eaton. Gnu octave, 2018. URL `https://www.gnu.org/software/octave/`.

Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

Ian T Jolliffe. Principal component analysis and factor analysis. In *Principal component analysis*, pp. 115–128. Springer, 1986.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Sébastien Loisel. Numeric javascript, 2012. URL `http://www.numericjs.com/`.

Karl Pearson. Principal components analysis. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 6(2):559, 1901.

Rudolph W. Preisendorfer and Curtis D. Mobley. Principal component analysis in meteorology and oceanography. *Elsevier Sci. Publ.*, 17:425, 1988.

Fazli Sapuan and Robert Plummer. Gpu accelerated javascript, December 2017. URL `http://gpu.rocks/`.

Lawrence Sirovich and Michael Kirby. Low-dimensional procedure for the characterization of human faces. *Josa a*, 4(3):519–524, 1987.

Roberto Stelling, 2018. URL `https://github.com/RobStelling/imgPca`.

International Telecommunication Union. Reccomendation itu-r bt.709-6, 06 2015. URL `http://www.itu.int/rec/R-REC-BT.709-6-201506-I/en`.

William R Zwick and Wayne F Velicer. Comparison of five rules for determining the number of components to retain. *Psychological bulletin*, 99(3):432, 1986.