MSAN 502 HW 5
Shannon McNish
8/14/17

My code consists of one main function called mySVD with two script parameters, k and filename. This function is called for each of three collections. It is set up to take the first script parameter as k, the next 10 script parameters for the first run (dogs collection), the next 10 script parameters for the second run (portraits collection) and the next set of 10 script parameters for the third run (random images collection). The random images are actually generated in the program and will be called random0.png - random9.png and saved to your working directory.

To run this program, have 2 collections of 10 images each saved to your working directory. Then you may run the file from your terminal, calling k first, then your files. For example, I used this code to test my collections with a k value of 50:

```
Python  mySVD.py 50 dog1.jpg dog2.jpg dog3.jpg dog4.jpg dog5.jpg dog6.jpg dog7.jpg
dog8.jpg dog9.jpg dog10.png portrait1.jpg portrait2.jpg portrait3.jpg portrait4.jpg
portrait5.jpg portrait6.jpg portrait7.jpg portrait8.jpg portrait9.jpg portrait10.jpg
random0.png random1.png random2.png random3.png random4.png random5.png
random6.png random7.png random8.png random9.png
```

Note that the random images will be created in the program so you don't need them saved before you run this script, but you do still need to call random0.png - random9.png in your script parameters.

PART A

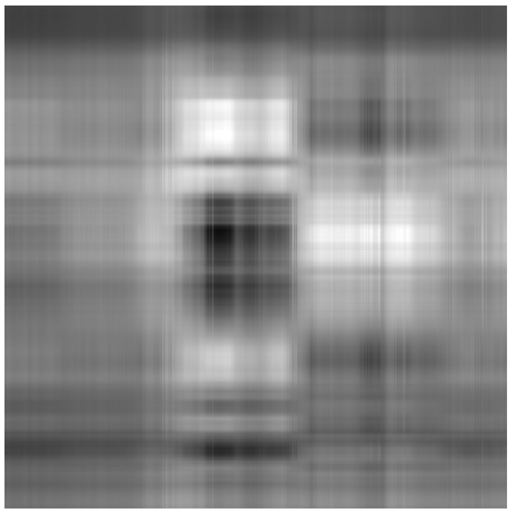
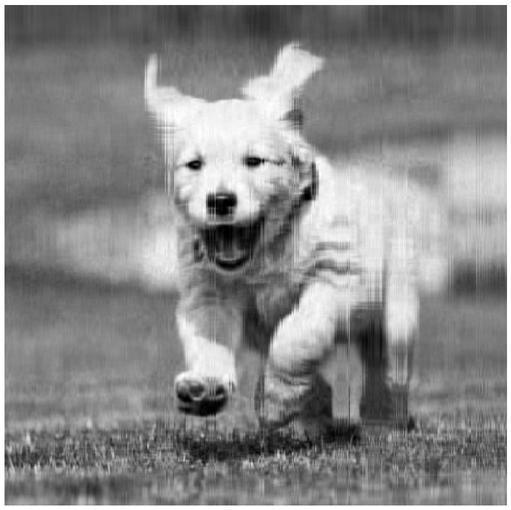
I chose 10 images of dogs, and 10 images of portraits, then generated 10 random grayscale images. All were 720x720.

Below is a sample of one image from each collection that shows what the images looks like after different number of singular values are used. The image quickly gets recognizable and K = 500 is indistinguishable from the full rank approximation.

The plots show the singular values in order of decreasing value averaged by index for each collection. They show a steep drop in the singular values, confirming that the first few singular values add the most information to the approximation. The slope of the singular values of the random grayscale images has a large drop at K=2 and then has a less flat slope than the two image collections, showing that each singular value is adding more information as compared to the singular values from the images.

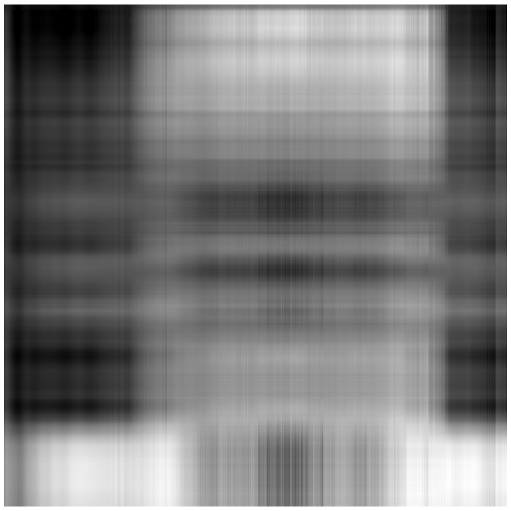
Dogs

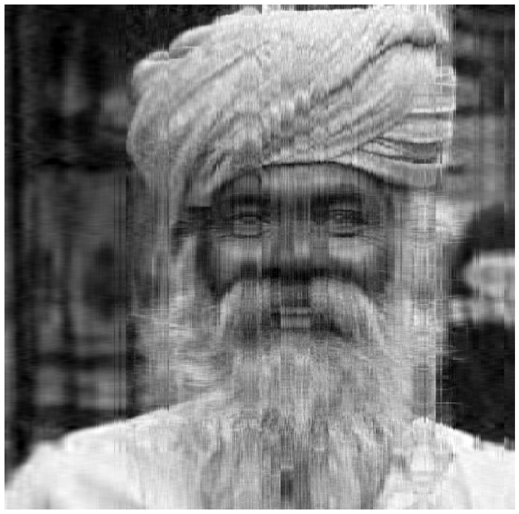
Sample of one dog with k=2, k = 50, k=500, and the original (k=750)

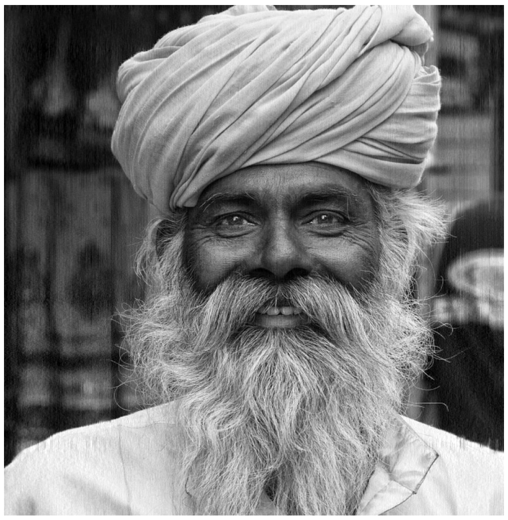



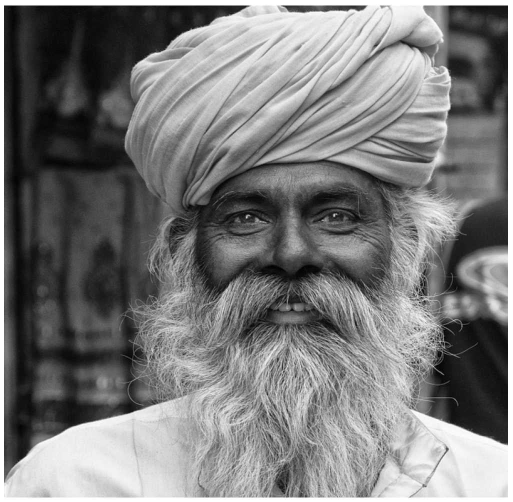
Portrait

Sample of one portrait with k=2, k = 50, k=500, and the original (k=750)

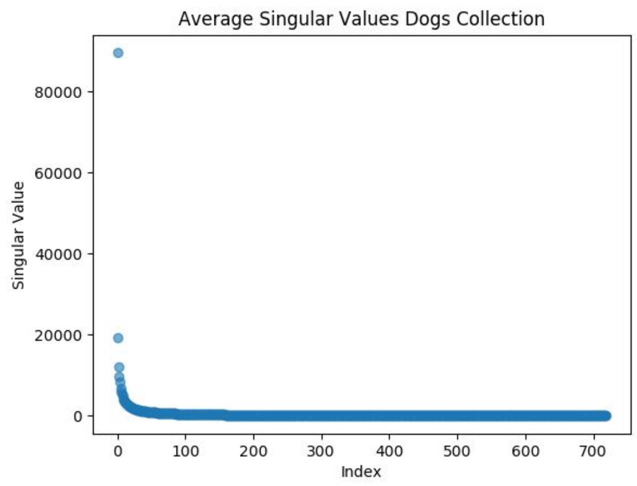


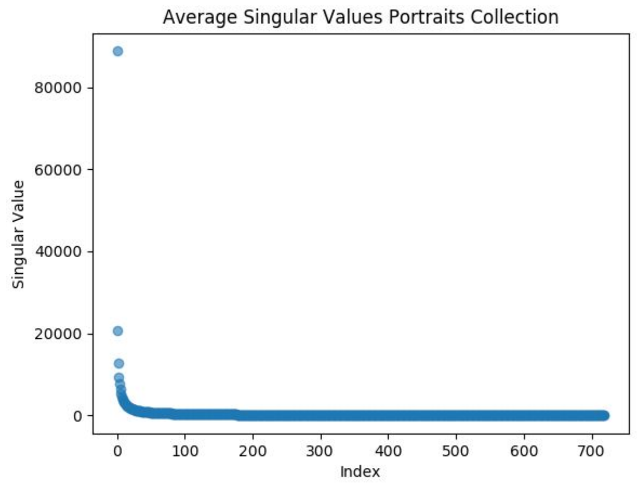


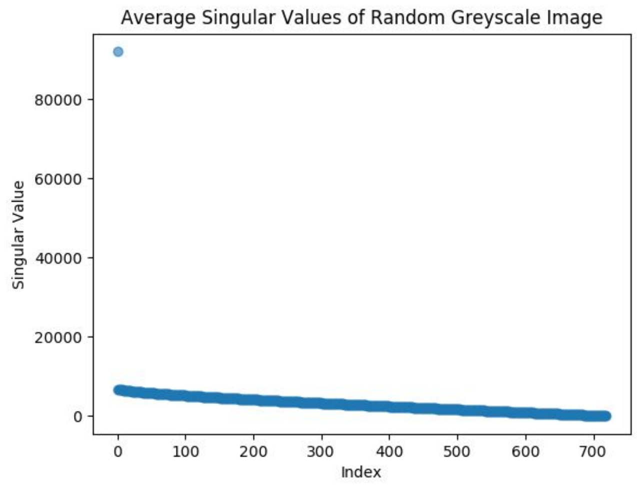




Plots


Average Singular Values Dogs Collection
Singular Value
Index
0
20000
40000
60000
80000
0
100
200
300
400
500
600
700


Average Singular Values Portraits Collection
Singular Value
Index
0
20000
40000
60000
80000
0
100
200
300
400
500
600
700


Average Singular Values of Random Greyscale Image
Singular Value
Index
0
20000
40000
60000
80000
0
100
200
300
400
500
600
700

PART B

The compression ratio on each collection is the same for each k since each image is 720x720.
The singular value ratio for each collection represents the total mass of the singular values associated with my compression divided by the total mass of the entire singular values. I calculated this using the average singular values for the collection. Comparing the values of the singular value ratio for the dogs collection vs portrait collection at k=50 shows that the singular value of the dogs collection has slightly more mass than that of the portrait collection for the first 50 singular values. This means that, on average, you are able to reconstruct more of an image in the dog collection using the same number of singular values.

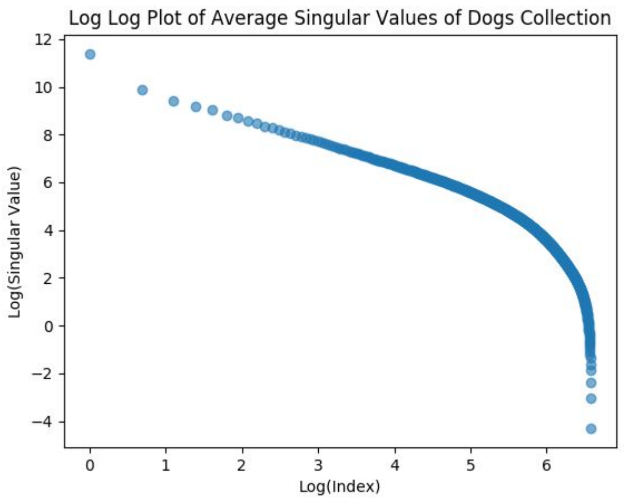
	Compression Ratio	**Dogs Collection Singular Value Ratio**	**Portrait Collection Singular Value Ratio**
K = 1 (Too messy)	0.003	0.284	0.286
K = 50 (Slightly messy)	0.139	0.749	0.72
K = 400 (Indistinguishable)	0.972	0.981	0.968

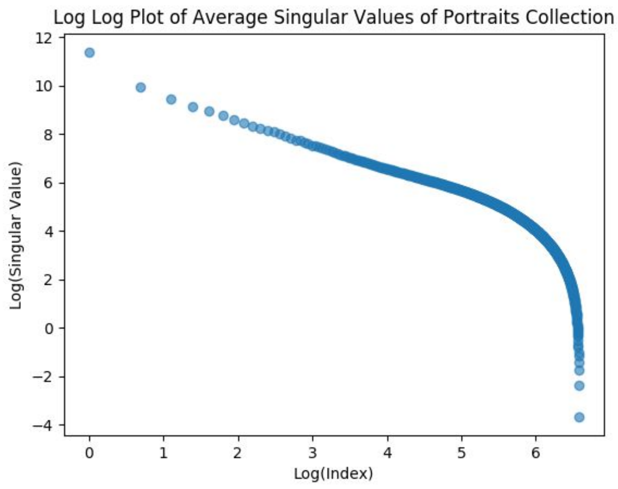
PART C

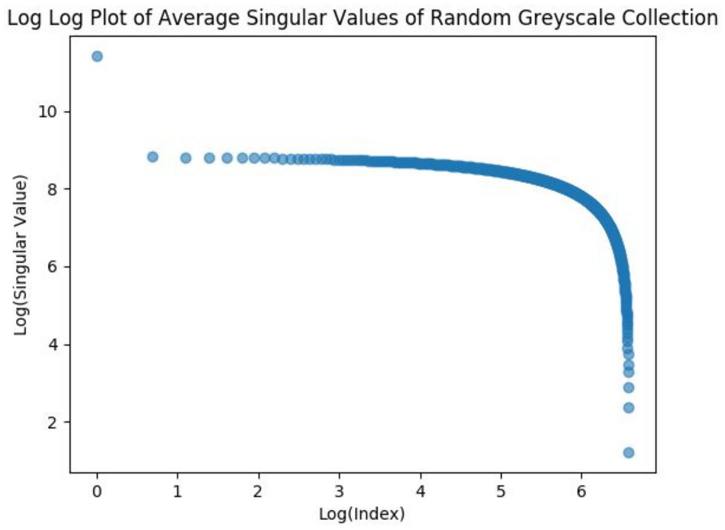
Power-law dependencies are of the form y(x) = kx^n. In the case of singular value decomposition, it means that the number the singular values needed to compress a picture at a certain ratio depends on the mass of the singular values. Taking the log of the singular values and index of singular values converts this to a relationship whose slope and intercept are related to the unknown values of n and k.

In the log-log plot of each collection, you can see that the slope of decay is very similar for the dogs and portraits collection, I estimate it to be about -0.8 for the first 20 k. For the random grayscale pictures, the slope is very steep for the first couple k, then goes almost flat for the next 20 k.

I would classify pictures into the third, random grayscale category that are seemingly random, and the algorithm can't learn from the other points around it. Any pictures that represent objects and shapes will have a log-log plot that looks like the dogs or portraits plots.


Log Log Plot of Average Singular Values of Dogs Collection
Log(Singular Value)
Log(Index)
12
10
8
6
4
2
0
−2
−4
0
1
2
3
4
5
6


Log Log Plot of Average Singular Values of Portraits Collection
Log(Singular Value)
Log(Index)
12
10
8
6
4
2
0
−2
−4
0
1
2
3
4
5
6


Log Log Plot of Average Singular Values of Random Greyscale Collection
Log(Singular Value)
Log(Index)
10
8
6
4
2
0
1
2
3
4
5
6