PCA Example

Principal Components Analysis Real World Example - Project 4 Report for Machine Learning

Overview

Description

An example of Principal Component Analysis to reduce the dimensionality of datasets. Three test datasets are used, `Iris`, `Optdigits`, and `LFW Crop`. They're placed in a `/data` directory which is under `.gitignore`. As such they are not uploaded to this repository. All data has been preprocessed by removing all strings.

> Notice: This code base is not maintained. Many things are hard code. The code in this repository was strictly used for created the graphics for this report and **is not intended to be forked or reused.** It may be used for reference within the confines of the license.

Algorithm

The Basics

1. Normalize by Z-Score

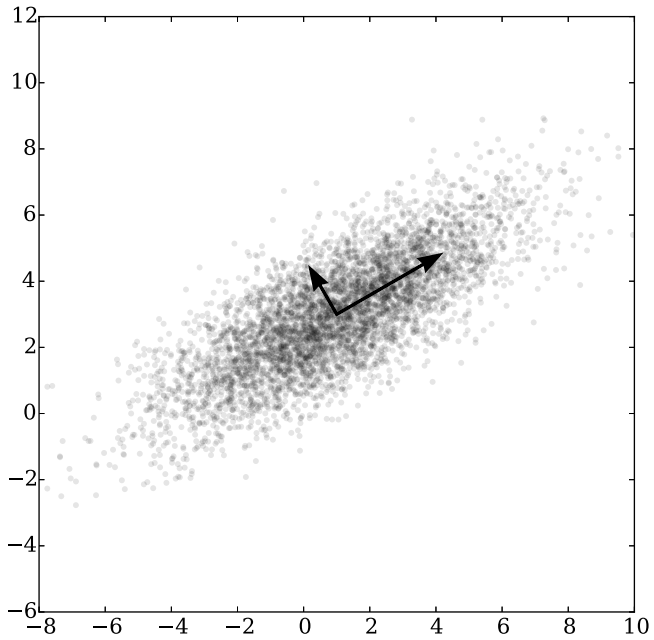
<

div style="text-align: right">$\hat{X} = \frac{X - \mu}{\sigma}$

<

div> 2. Calculate Principal Components

<



div style="text-align: right">

Order in decreasing value the eigenvalues of the covariance matrix

<

div>

3. Rotate onto Principal Components

<

div style="text-align: right">$Y = XP$

<

div> 4. Project onto the d-dimensional subspace defined by the first d principal component

<

div style="text-align: right">$Y[:, 0:d]$

<

div> 5. Reconstruct

<

div style="text-align: right">$X^* = (YP^T)\sigma\mu$

<

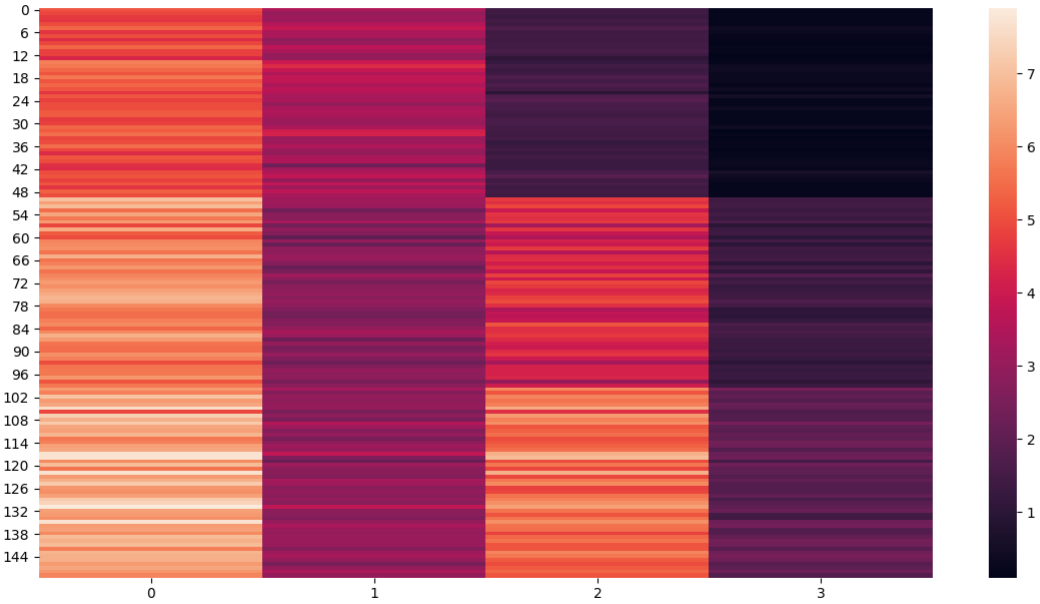
div>

Report

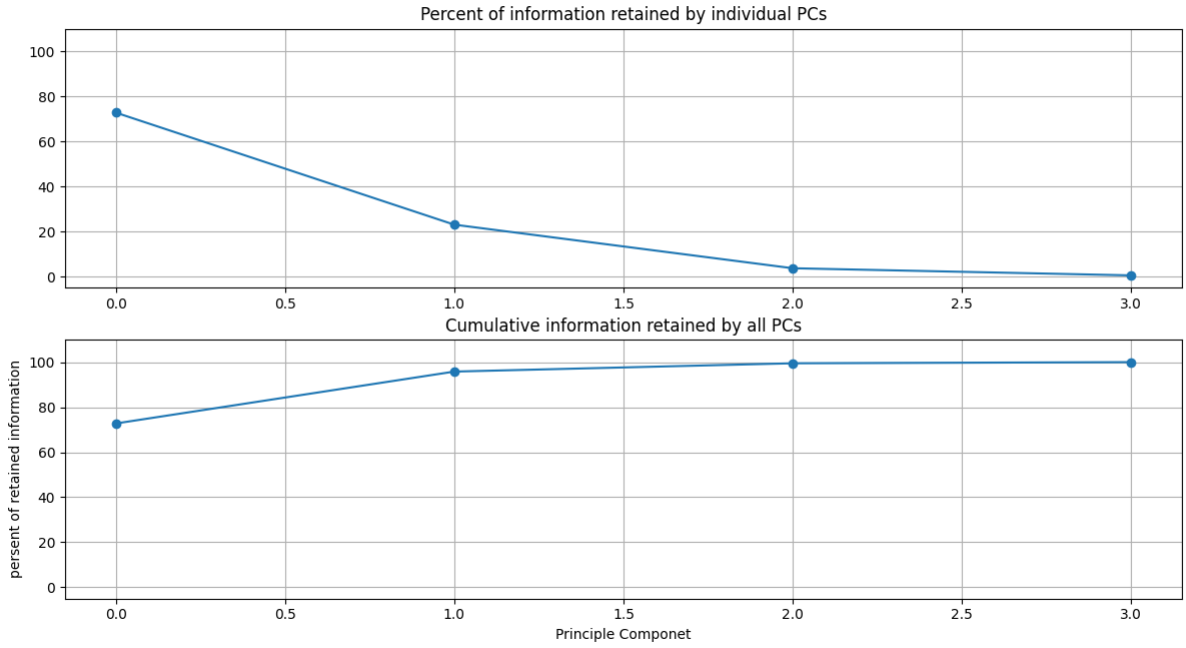
There are three parts to this report. Each part includes an eigenvector heatmap, a scaled eigenvector scree plot, and reconstructions of four stages of data retention. The Optdigits and LFW Crop sections include the showcase of random images alongside a data retention showcase.

Part I - Iris

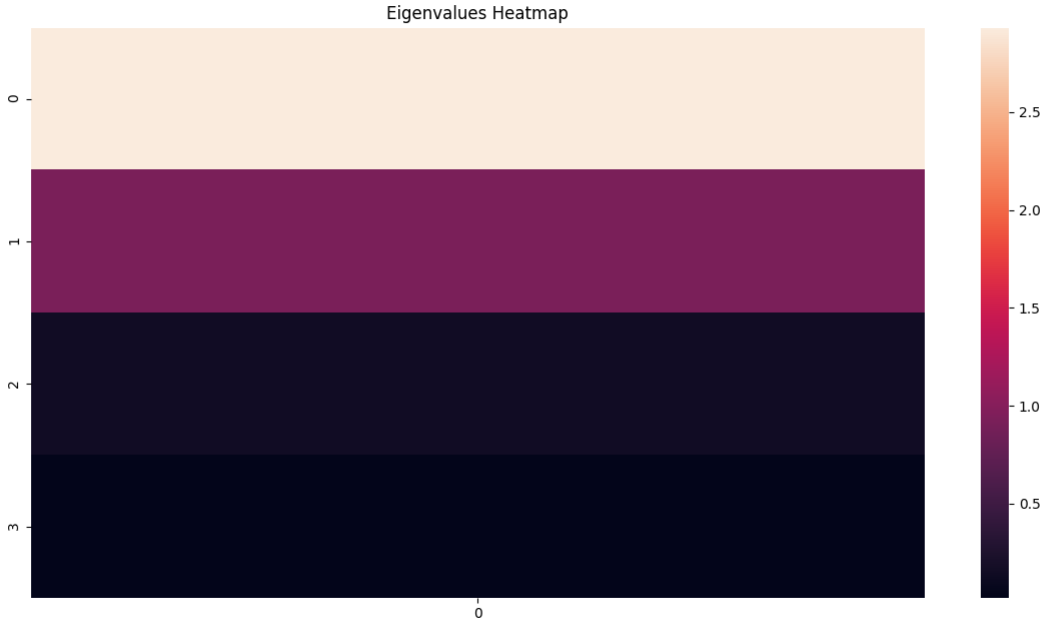
Iris Raw



Iris PCA Scree Map

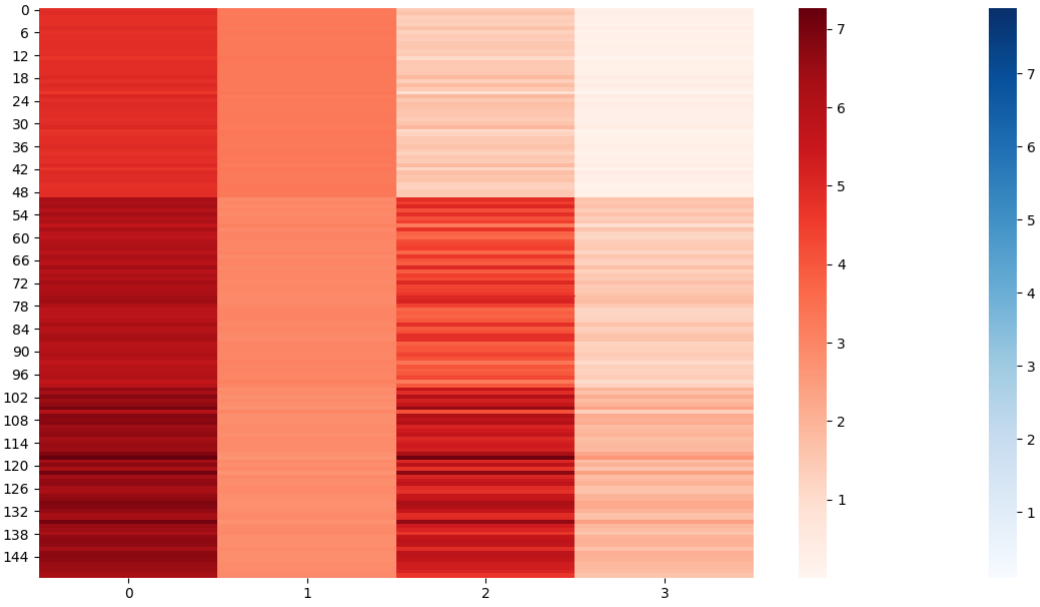


Eigenvalue Heatmap



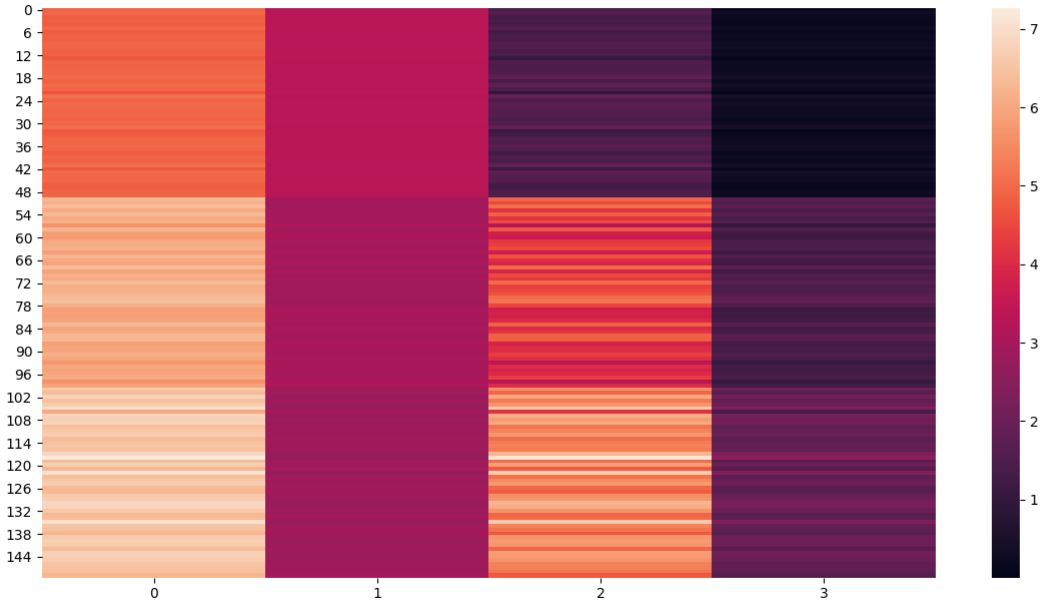
Iris Red Vs Blue

Iris Raw - Blue Iris 25% PCA - Red For this graph I tried to use contrasting colors but I think red overtook blue. That and because blue holds more detail it might not be able to stand its ground visually against red 20% PCA.



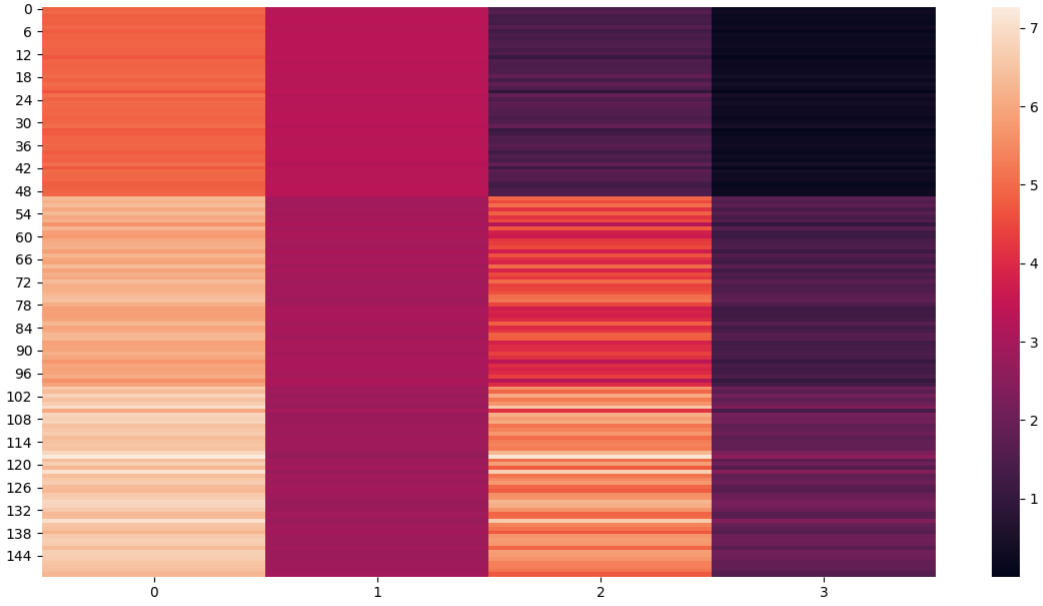
Iris at 80% retension used 2 dimensions.

Iris: percent = 20 d = 1

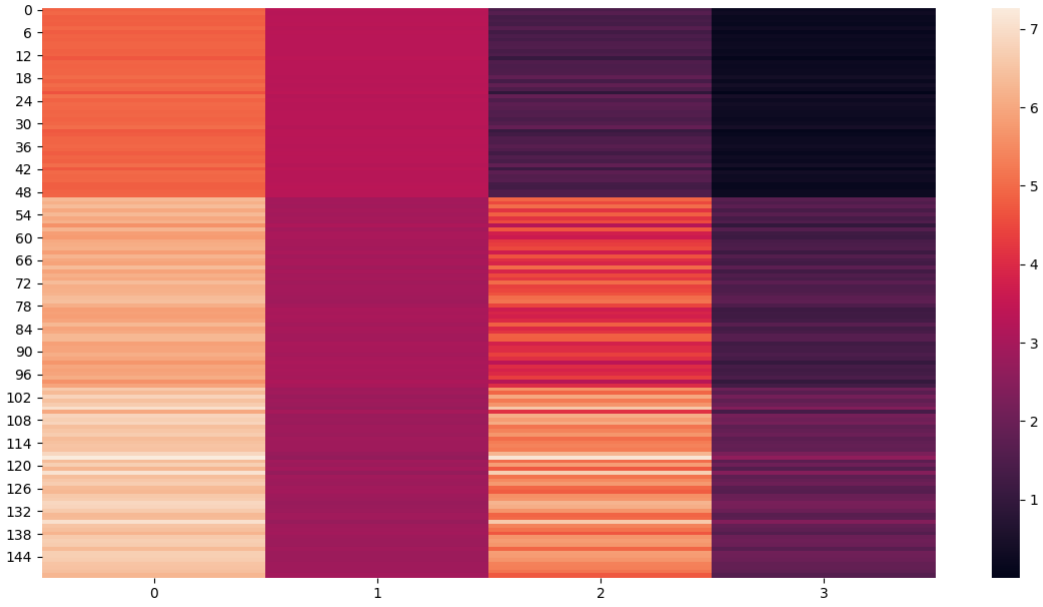


At d = 1 this holds 25% of our data. It clearly still hold enough to identify Irises.

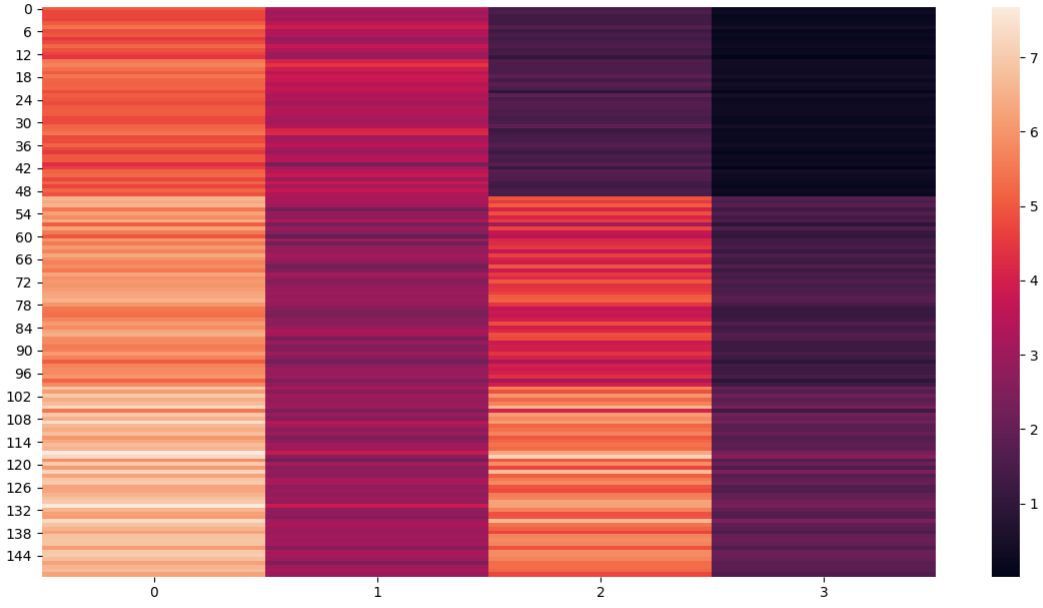
Iris: percent = 40 d = 1



Iris: percent = 60 d = 1

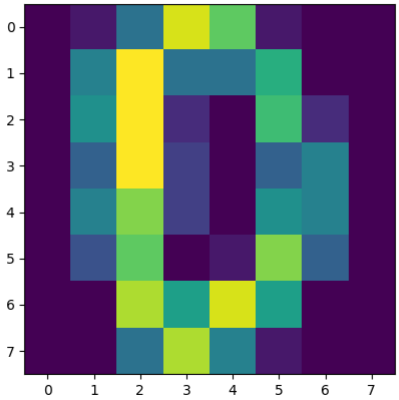


Iris: percent = 80 d = 2



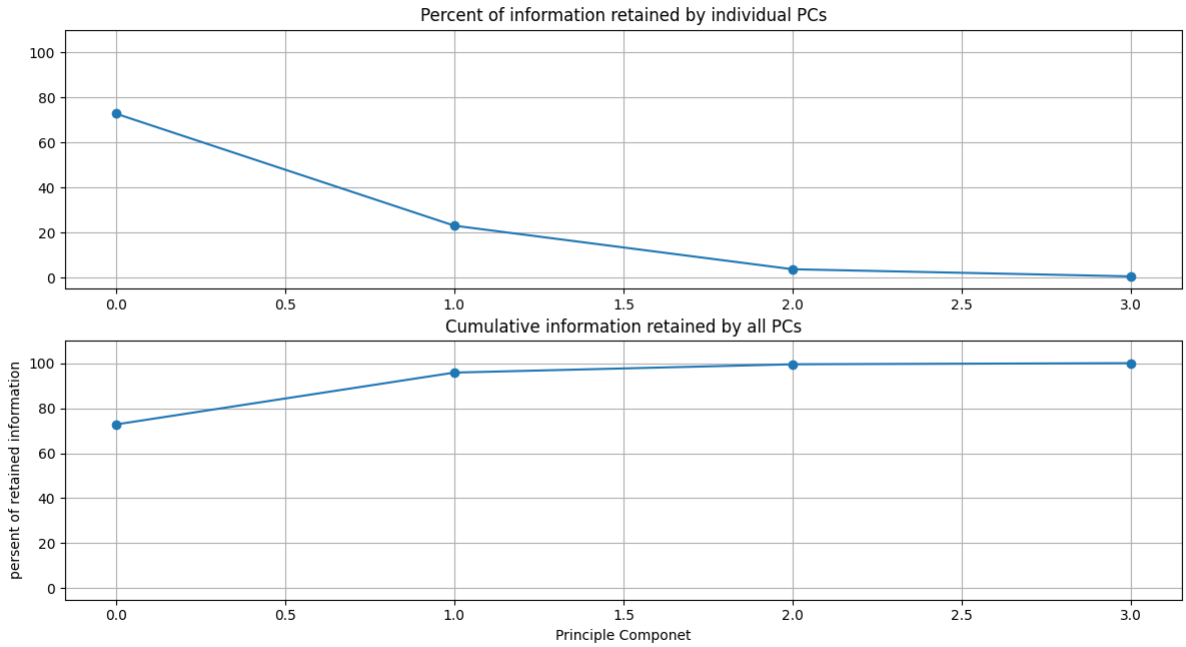
Part II - Optdigits

Sample taken = 0

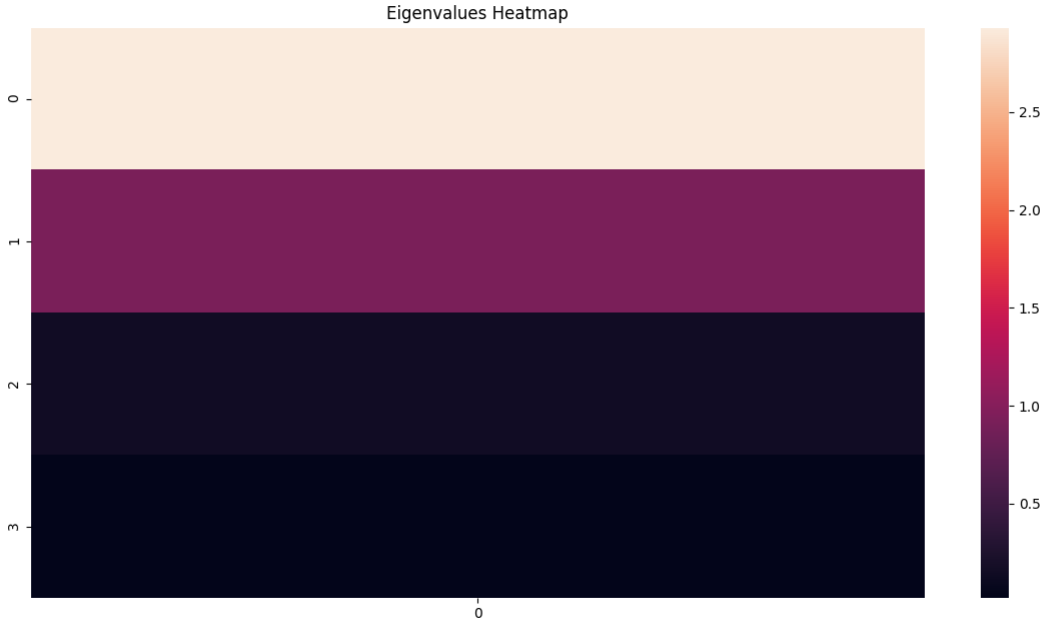


Octdigit Raw

Oct PCA Scree Map

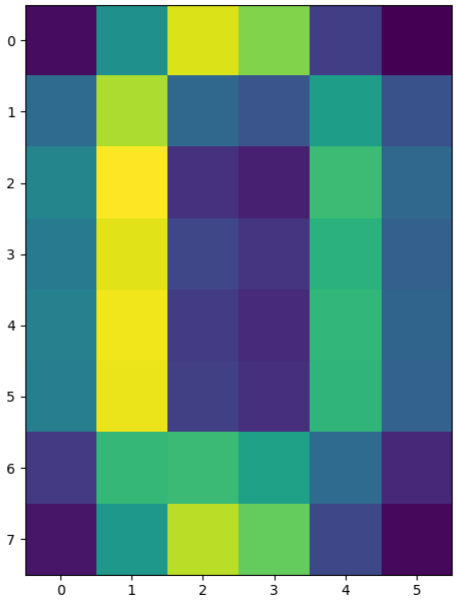


Eigenvalue Heatmap

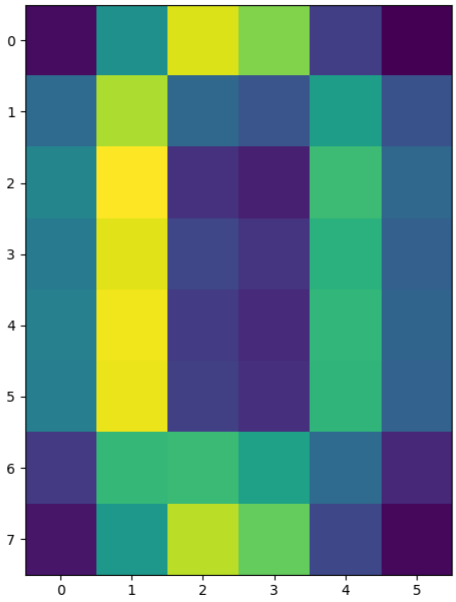


Optdigits at 80% retension used 2 dimensions.

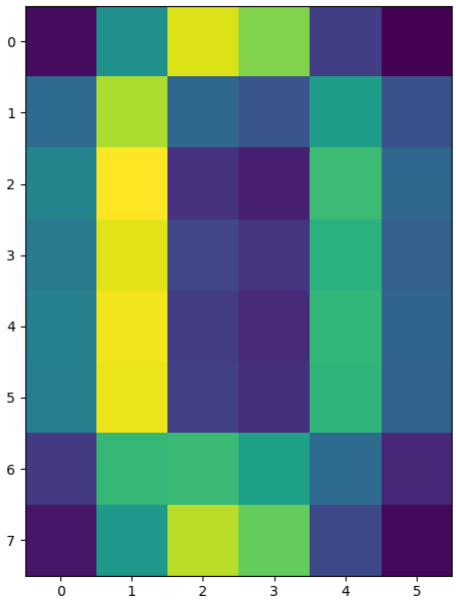
Optdigits: percent = 20 d = 1



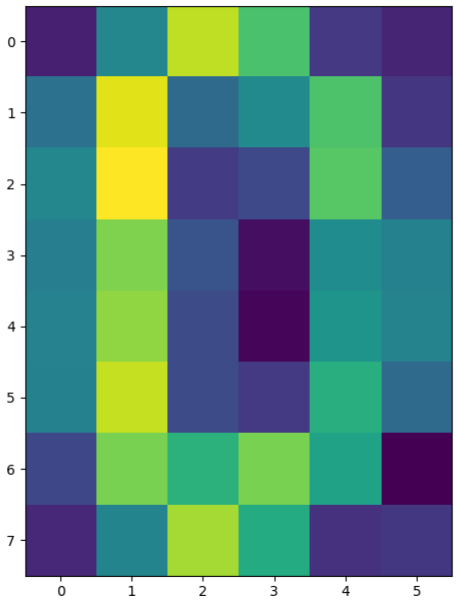
Optdigits: percent = 40 d = 1



Optdigits: percent = 60 d = 1

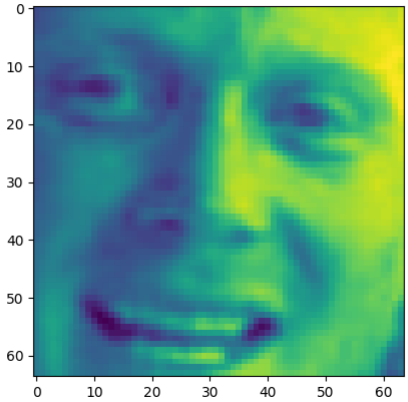


Optdigits: percent = 80 d = 2



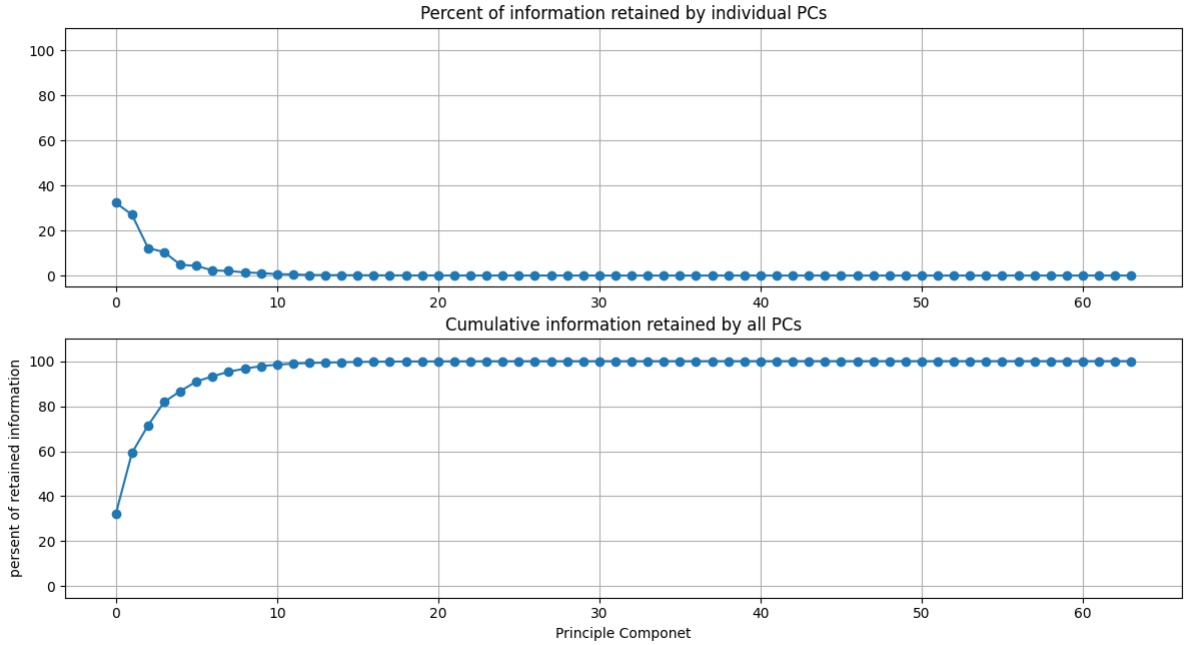
Part III - LFW Crop

Sample taken = 0

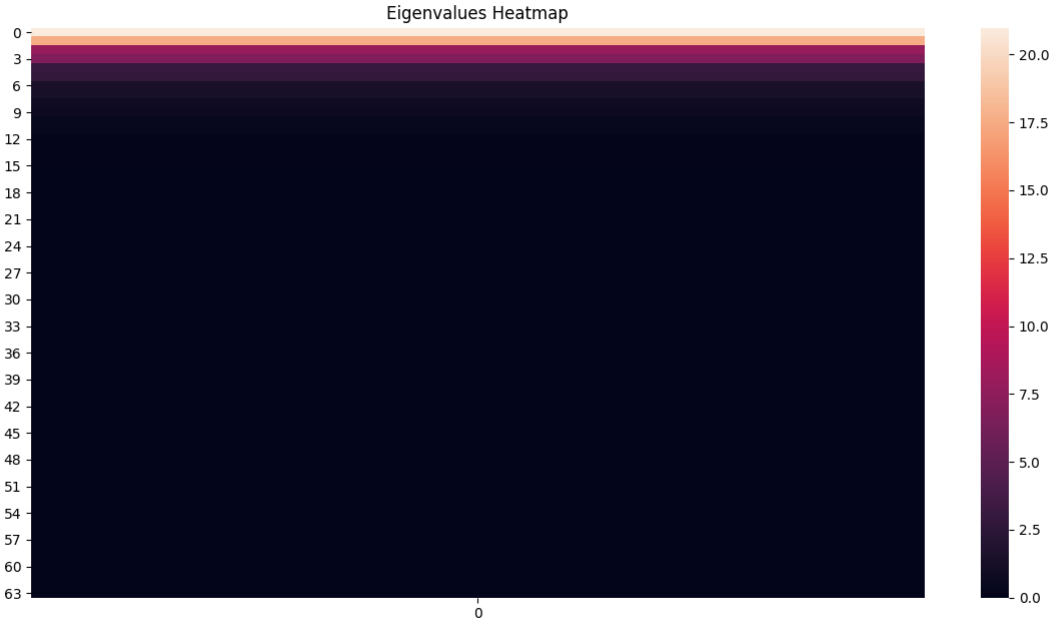


LFW Raw

LFW PCA Scree Map

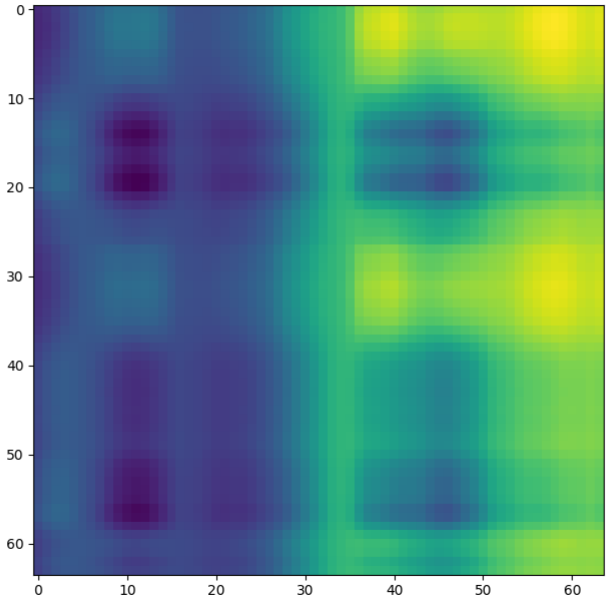


Eigenvalue Heatmap

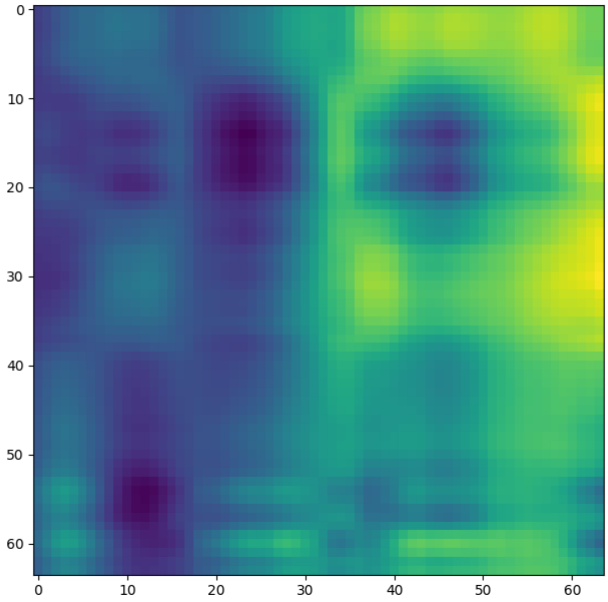


lfw at 80% retension used 4 dimensions.

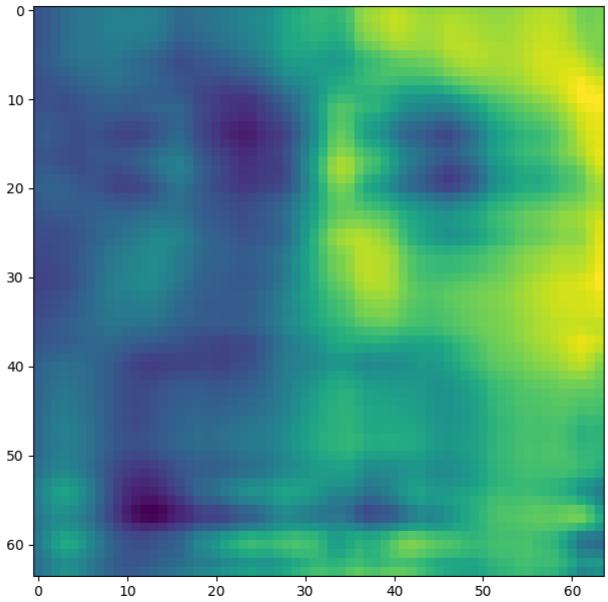
lfw: percent = 20 d = 1



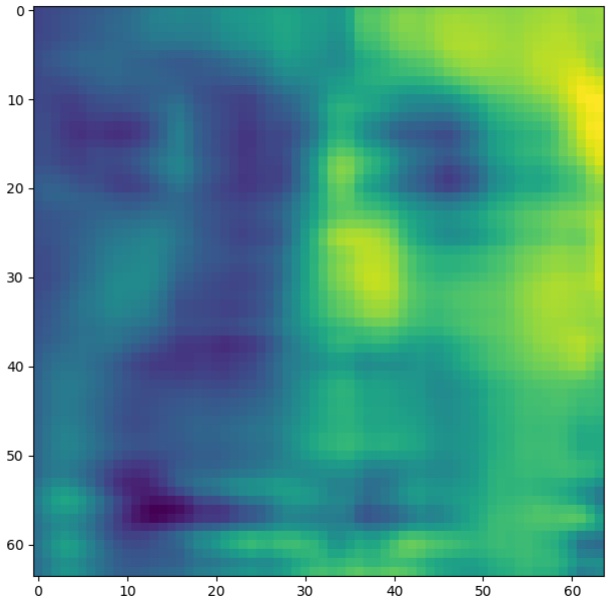
lfw: percent = 40 d = 2



lfw: percent = 60 d = 3



lfw: percent = 80 d = 4



Conclusion

The appearance of the reconstructions as we retain more dimensions sharpen. However, we might want to take advantage of this. For example in Iris the best looking set was the 20% or 1d. This is definitly not the case with the other datasets. LFW was unrecognizable bellow 80%.

y_projection size comparison

I saved an 80% or 4D projection of the first sample. The size of lfwcrop is 433,553,536, bites or aroun 433MB. There are 13231 frames inside of lfwcrop. That's around 32768 bytes per frame. Our projection is only 2176 bytes. That's 6.64% the size! That's a huge reduction! We can see that lfwcrop has 64 dimensions but our projection at 80% data retention has just 4 dimensions! 4/64 just happens to be 6.25%. That's very close to our real world value. I suspect the difference between our theoretical value and the one we got is overhead from numpy and the .npy filetype.

Licensing