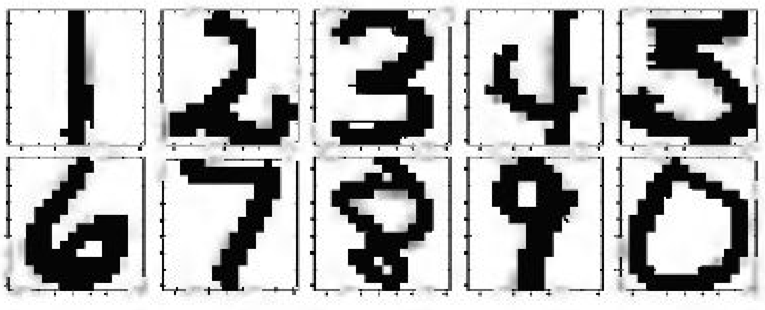
Mandub- I will finish in the morning. I have a study group for Bioinformatics from 1
:30, so let's meet before then/ I can come to your house. Here is the google link for this doc:

https://docs.google.com/document/d/1u706KZ2ase_VxaF2FhEs-_gDxCKS0WI6KguWCMpUJF
U/edit?usp=sharing

Intro- project explanation
Data explanation
Methodology
Experiments with best k value
Results

Our project was to computer recognition of handwritten digits. This is a classic machine learning problem and has many solutions. We used technique from linear algebra called singular value decomposition (SVD), which is a method for factoring a matrix. It has many different applications. It is also known as principal component analysis, or PCA. In our case, we are using it to reduce the dimensionality of the data to make our digit recognition program run in a reasonable amount of time.

The data we are working with comes from the U.S. Postal Service database of hand-written digits.

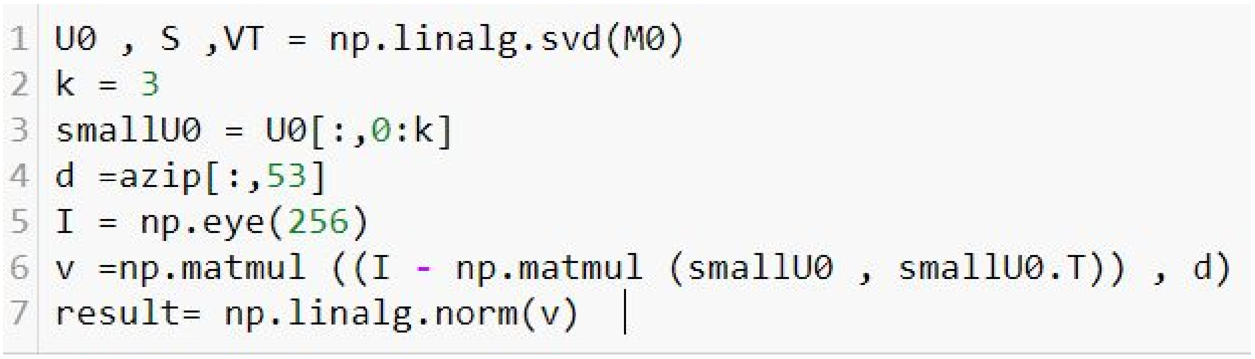


Each number is represented by 16 by 16 array of floats, representing the grayscale values, ranging from -1 to 1. It is actually stored as a list of 256 values. For example, here are the first 64 values of a digit (which I split into four rows of 16):

```
-1.   , -1.   , -1.   , -0.567, -0.064,  0.979, -0.009, -0.167, -0.999, -1.   , -1.  , -1.  , -1.  , -1.   , -1.   , -1.  ,
-1.   , -1.   , -1.   ,  0.405,  1.   ,  1.   ,  1.   ,  1.   , 0.482, -0.701, -1.   , -1.   , -1.   , -1.   , -1.   , -1.   ,
-1.   , -1.   , -1.   , -0.242,  1.   ,  1.   ,  1.   ,  1.   , 1.   ,  0.809,  0.351, -0.764, -1.   , -1.   , -1.   , -1.   ,
-1.   , -1.   , -1.   , -0.091,  1.   ,  1.   ,  1.   ,  1.   ,  0.848,  1.   ,  1.   ,  0.748, -0.536, -1.   , -1.   , -1.   ,
```

These lists of 256 are actually columns in a 256 row table. Each column corresponds to one digit. The training set had 1707 digits, so our initial array was a 256 x 1707 array. A separate array that was 1 x 1707 was an index which had the digit , so we could know what each column was supposed to represent. For example, if position 3 in the 1 x1707 array had a zero, we knew that column 3 of other array represented a zero.

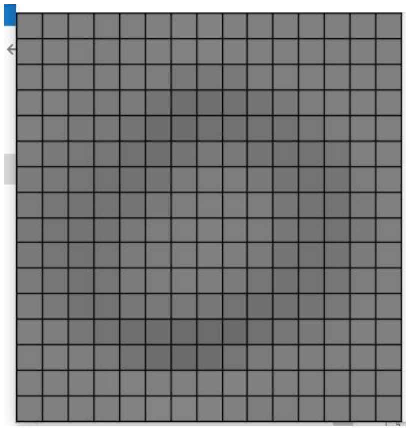
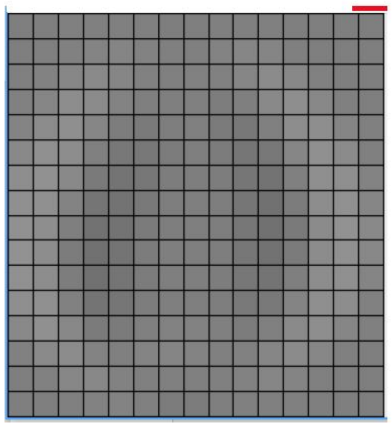
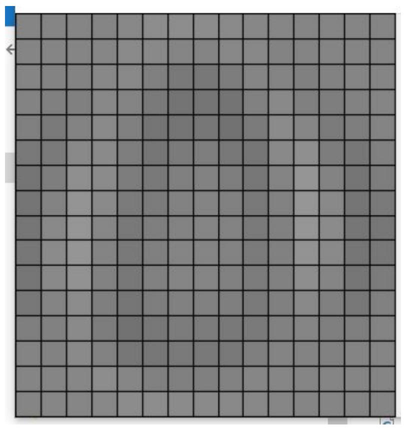
The first step in processing the data was to group like digits into their own array. For example, there were 319 zeros, so we created an array 256 x 319 that contained all the zeros (M0). Next we put all the ones together (M1), etc. At this point, we are ready for SVD.



```
1 U0 , S ,VT = np.linalg.svd(M0)
2 k = 3
3 smallU0 = U0[:,0:k]
4 d =azip[:,53]
5 I = np.eye(256)
6 v =np.matmul ((I - np.matmul (smallU0 , smallU0.T)) , d)
7 result= np.linalg.norm(v)
```

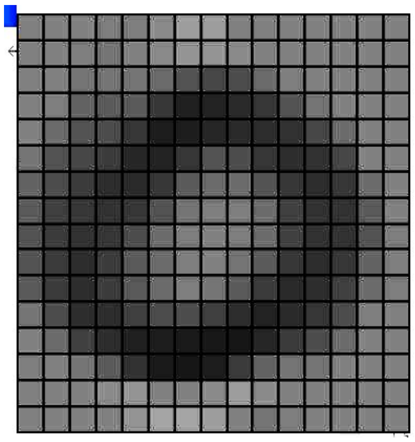
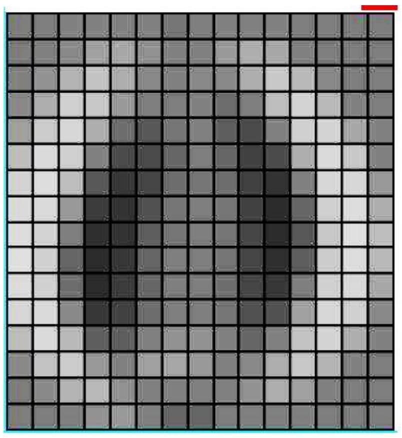
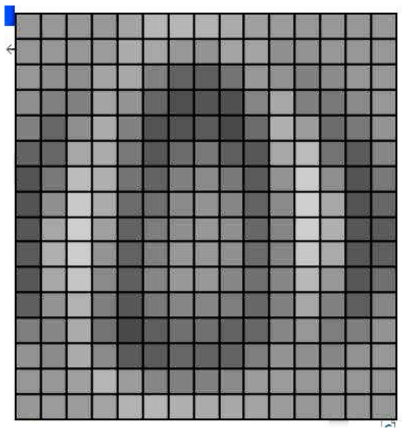
Line 1 uses numpy to perform the SVD, creating three matrices. We are interested in the first matrix, U0. U0 is a 256 x 256 matrix. The first column represents the strongest common signal coming from all the matrices put together, the second, the second strongest signal, etc.

First column Second Column Third Column

After increasing the contrast in an image editor:

First column Second Column Third Column

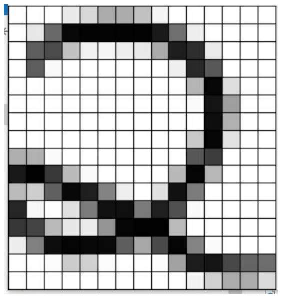




So U0 contains our reference values. The next step in line 3 is to take a subset of U0. In this case, we will only use the first three columns of our 256 column matrix. Next we take the norm of the least squares distance between our “unknown” matrix and our small U0. This gives us a number rating how good a match our number is to zero. In this case it was 10.257. Now we repeat the process with matrices for all the other digits. The matrix that produces the best score is our number.

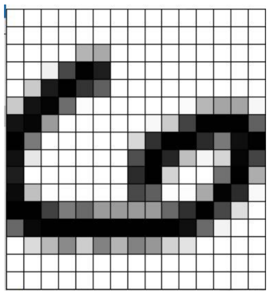
[1464, 131, 45, 23, 9, 10, 7, 2, 1, 2, 1, 3, 1, 1, 0, 2, 2, 0, 1, 0]

Examples of numbers that were misclassified;

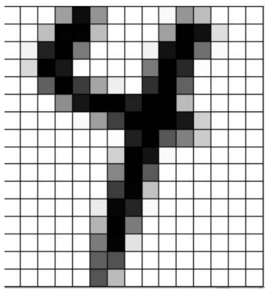
Zero



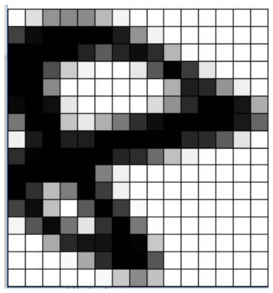
Zero



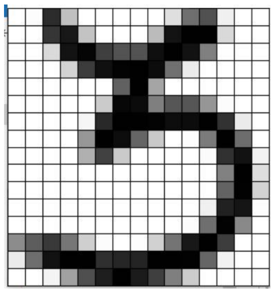
Nine



Four



Five



Seven