Numerical optimization and large scale linear algebra

Stelios Giagkos f3352410

Classification of Handwritten Digits

Introduction

In this assignment, we aim to construct an algorithm for classifying handwritten digits based on Singular Value Decomposition (SVD). The algorithm uses a training set to compute the SVD of each class matrix, where the first few singular vectors (ranging from 5 to 20) are used as the basis for classification. The classification process will involve determining how well an unknown test digit can be represented in terms of the respective basis using the relative residual vector in the least squares problem as a measure.

Specific Tasks:

1. **Tune the algorithm for accuracy of classification**: We will experiment with different numbers of singular vectors (from 5 to 20) and evaluate the accuracy of classification. A table or graph will be provided to show the percentage of correctly classified digits as a function of the number of basis vectors.
2. **Check if all digits are equally easy or difficult to classify**: We will examine the performance for each digit class to see if any digits are more challenging to classify. Additionally, we will analyze some misclassified digits to determine if poor handwriting contributes to the difficulty.
3. **Analyze the singular values of different classes**: We will inspect the singular values of each class to assess if it is worthwhile to use different numbers of basis vectors for different classes. Based on these observations, we will perform experiments to test if using fewer basis vectors for some classes improves classification accuracy.

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content
/drive", force_remount=True).
```

```
Train Set Shape: (257, 1707)
Test Set Shape: (257, 2007)
```

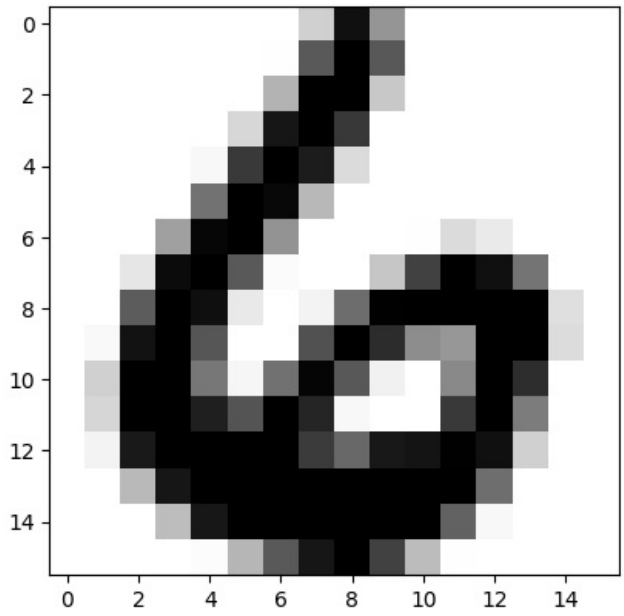
	0	1	2	3	4	5	6	7	8	9	...	1697	1698	1699	1700	1701	170
0	6.0	5.000	4.0	7.0	3.0	6.0	3.00	1.0	0.0	1.0	...	8.0	0.000	5.0	1.0	9.0	8.
1	-1.0	-1.000	-1.0	-1.0	-1.0	-1.0	-1.00	-1.0	-1.0	-1.0	...	-1.0	-1.000	-1.0	-1.0	-1.0	-1.
2	-1.0	-1.000	-1.0	-1.0	-1.0	-1.0	-1.00	-1.0	-1.0	-1.0	...	-1.0	-1.000	-1.0	-1.0	-1.0	-1.
3	-1.0	-1.000	-1.0	-1.0	-1.0	-1.0	-1.00	-1.0	-1.0	-1.0	...	-1.0	-1.000	-1.0	-1.0	-1.0	-1.
4	-1.0	-0.813	-1.0	-1.0	-1.0	-1.0	-0.83	-1.0	-1.0	-1.0	...	-1.0	-0.567	-1.0	-1.0	-1.0	-1.

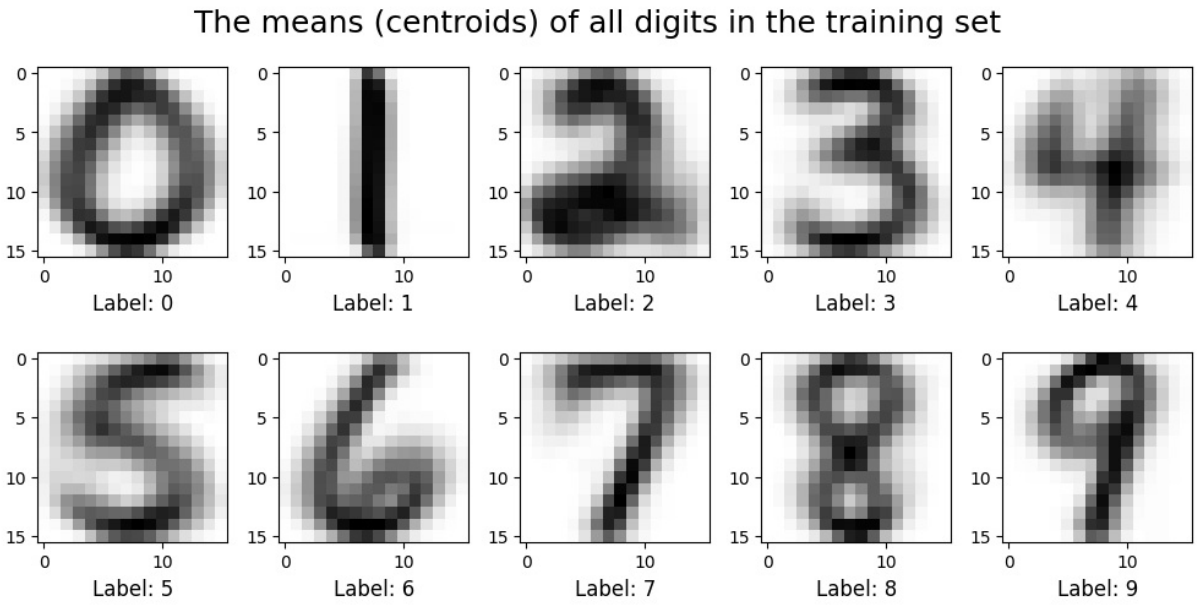
5 rows × 1707 columns

	0	1	2	3	4	5	6	7	8	9	...	1997	1998	1999	2000	2001	200
0	9.0	6.0	3.000	6.0	6.0	0.0	0.0	0.0	6.0	9.0	...	7.0	5.000	2.00	0.0	4.0	3.
1	-1.0	-1.0	-1.000	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	...	-1.0	-1.000	-1.00	-1.0	-1.0	-1.
2	-1.0	-1.0	-1.000	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	...	-1.0	-1.000	-1.00	-1.0	-1.0	-1.
3	-1.0	-1.0	-1.000	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	...	-1.0	-1.000	-1.00	-1.0	-1.0	-1.
4	-1.0	-1.0	-0.593	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0	...	-1.0	-0.885	-0.98	-1.0	-1.0	-1.

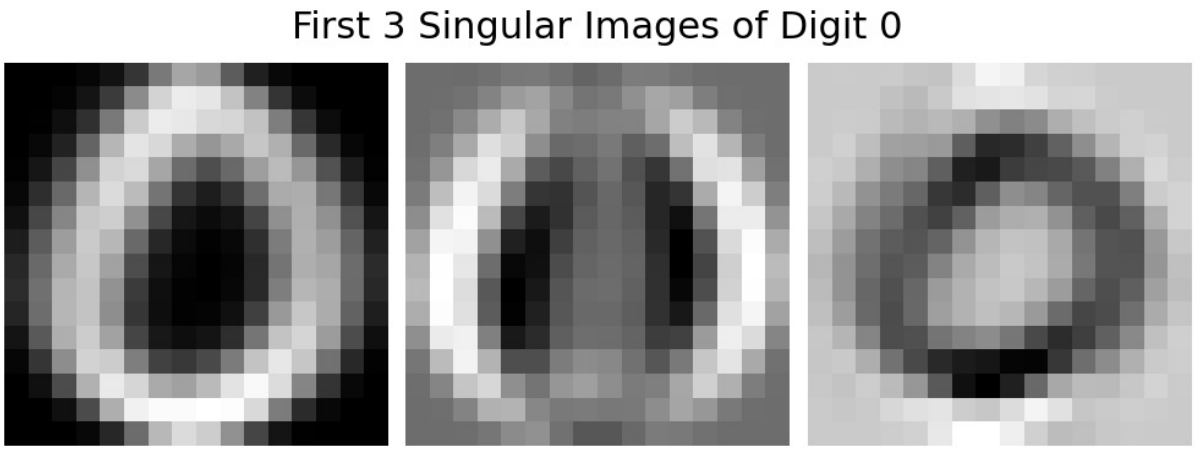
5 rows × 2007 columns

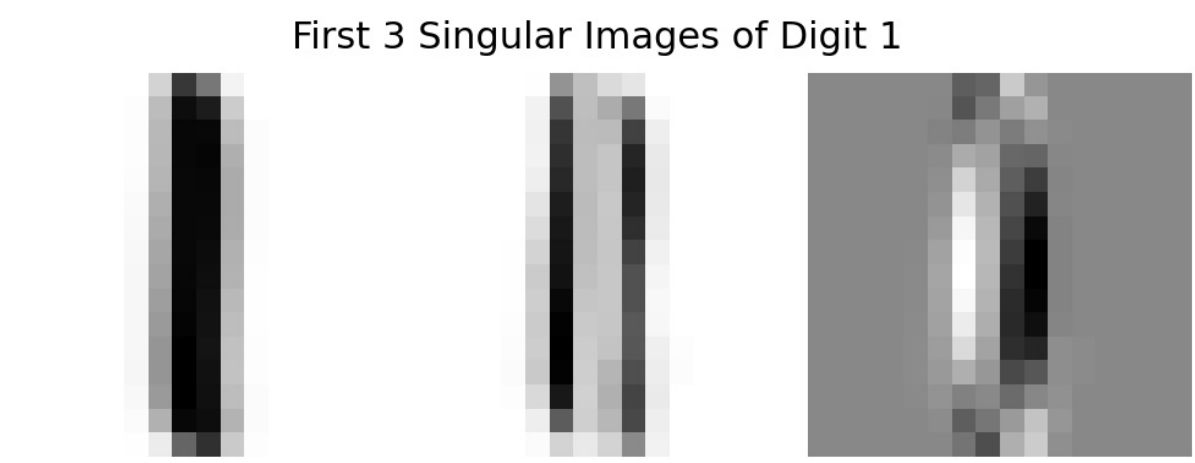
Number: 6.0

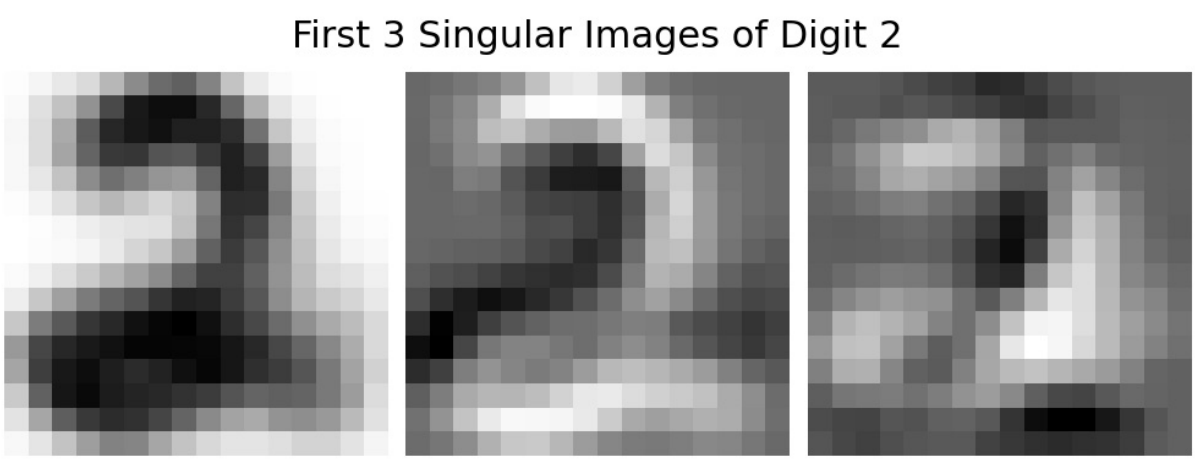




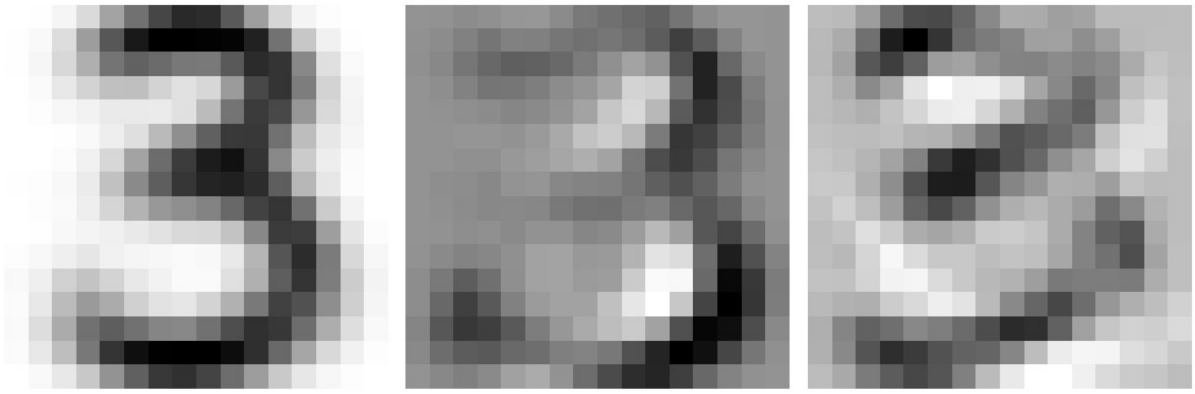
```
Sanity Check: u, s, vh shape for each digit:
Digit: 0, u shape: (256, 256), s shape: (256,), vh shape: (256, 319)
Digit: 1, u shape: (256, 252), s shape: (252,), vh shape: (252, 252)
Digit: 2, u shape: (256, 202), s shape: (202,), vh shape: (202, 202)
Digit: 3, u shape: (256, 131), s shape: (131,), vh shape: (131, 131)
Digit: 4, u shape: (256, 122), s shape: (122,), vh shape: (122, 122)
Digit: 5, u shape: (256, 88), s shape: (88,), vh shape: (88, 88)
Digit: 6, u shape: (256, 151), s shape: (151,), vh shape: (151, 151)
Digit: 7, u shape: (256, 166), s shape: (166,), vh shape: (166, 166)
Digit: 8, u shape: (256, 144), s shape: (144,), vh shape: (144, 144)
Digit: 9, u shape: (256, 132), s shape: (132,), vh shape: (132, 132)
```



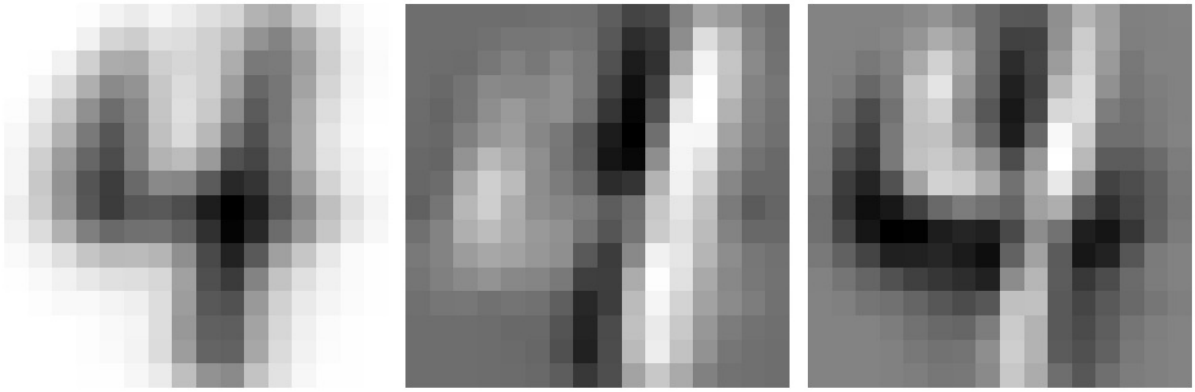




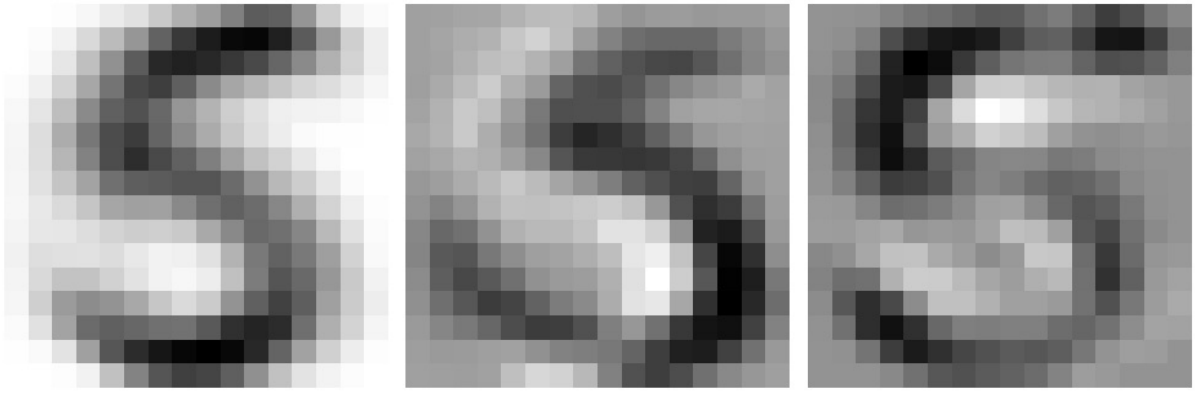
First 3 Singular Images of Digit 3



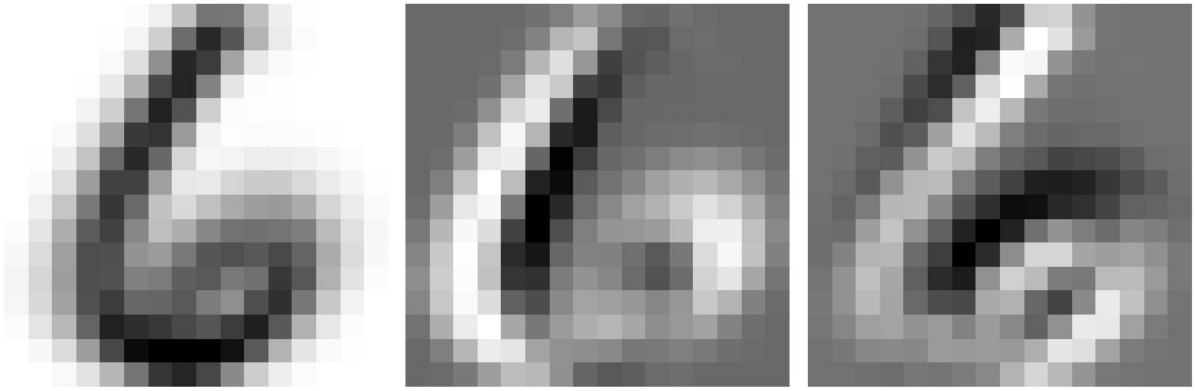
First 3 Singular Images of Digit 4



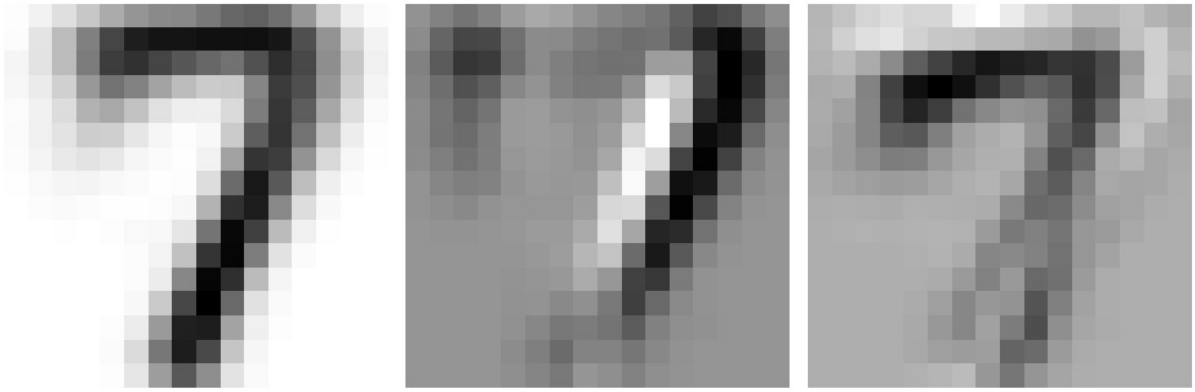
First 3 Singular Images of Digit 5



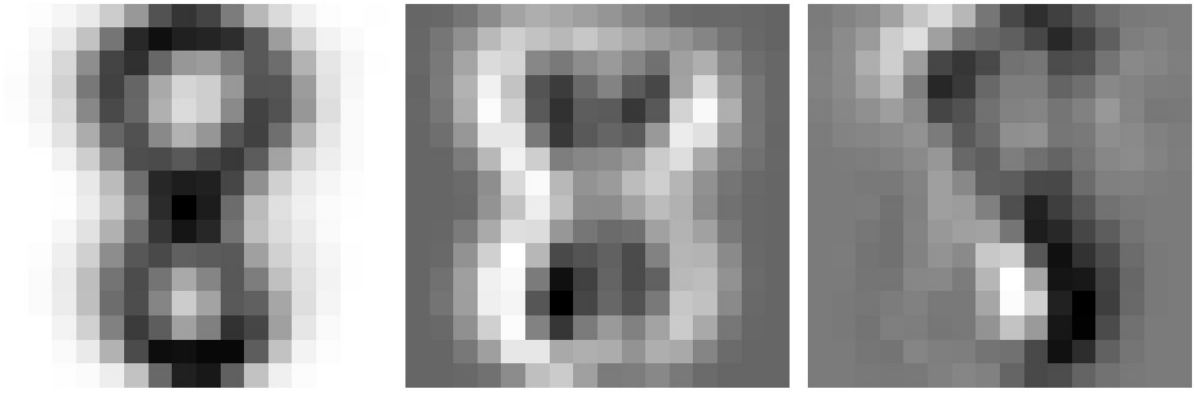
First 3 Singular Images of Digit 6



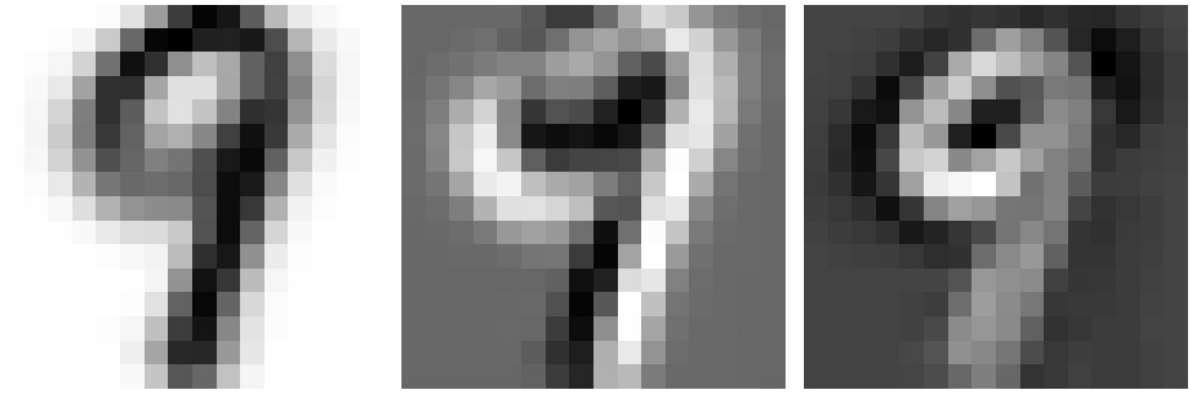
First 3 Singular Images of Digit 7



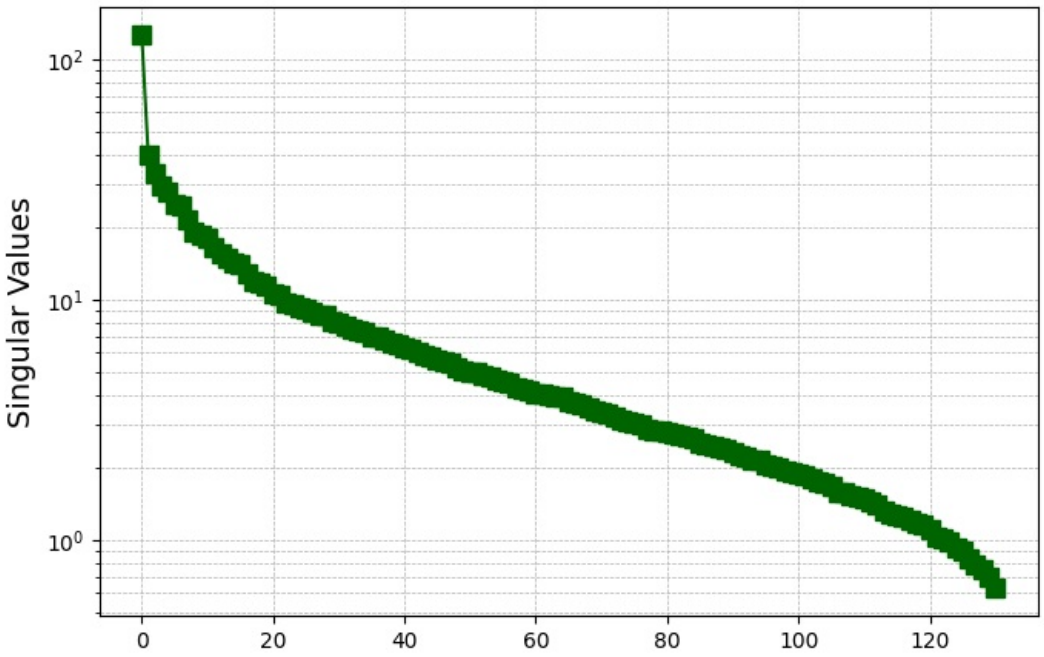
First 3 Singular Images of Digit 8

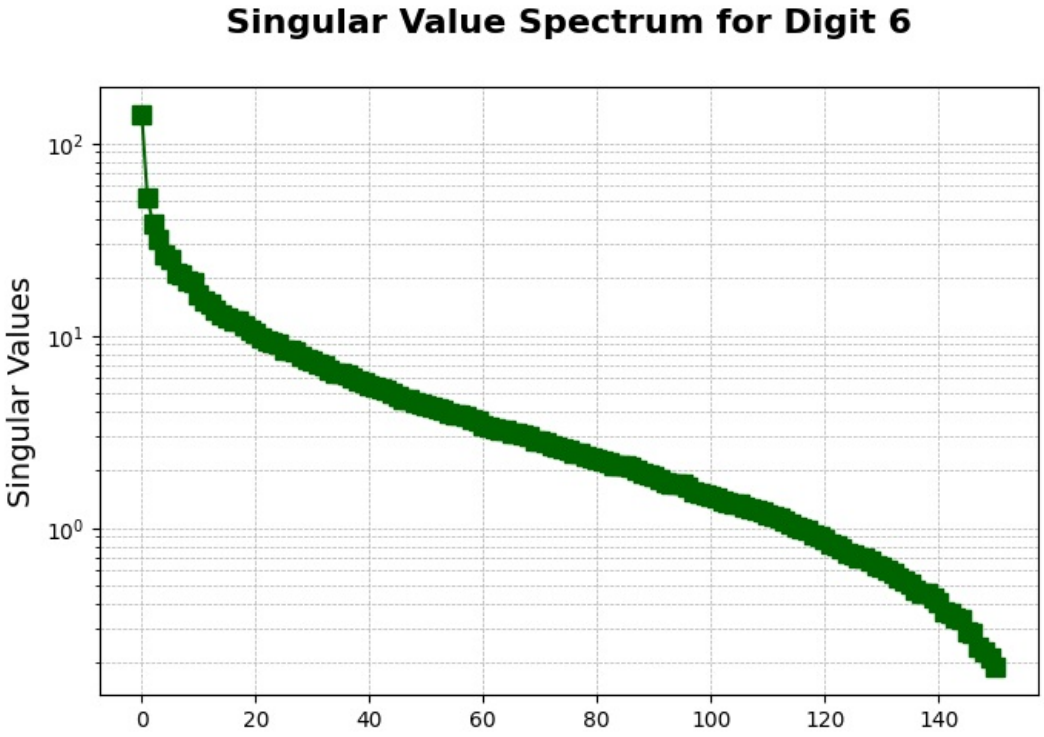


First 3 Singular Images of Digit 9

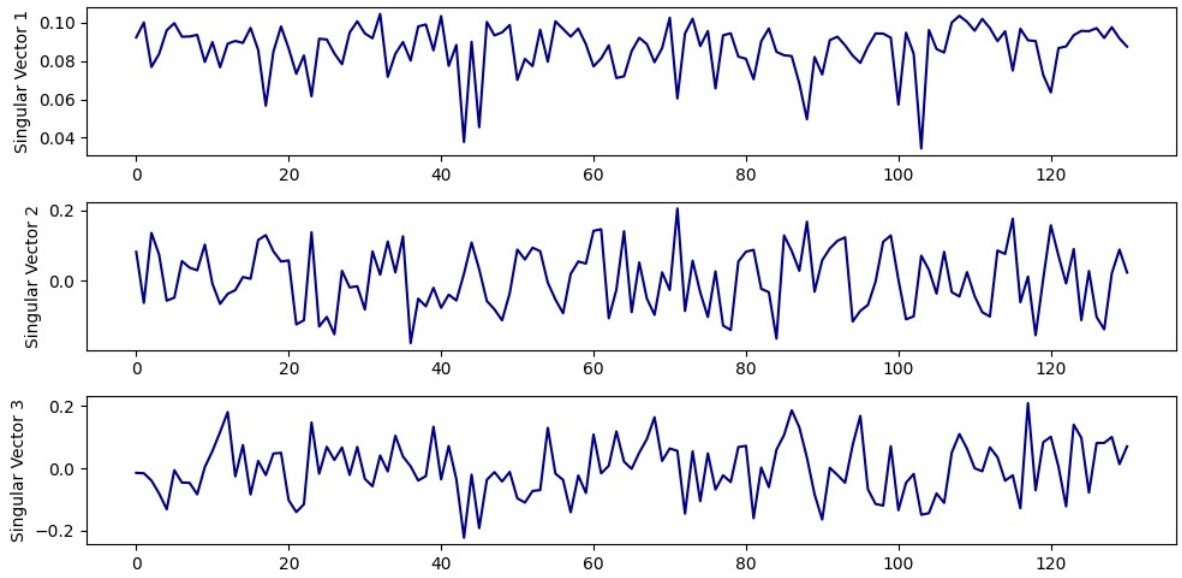


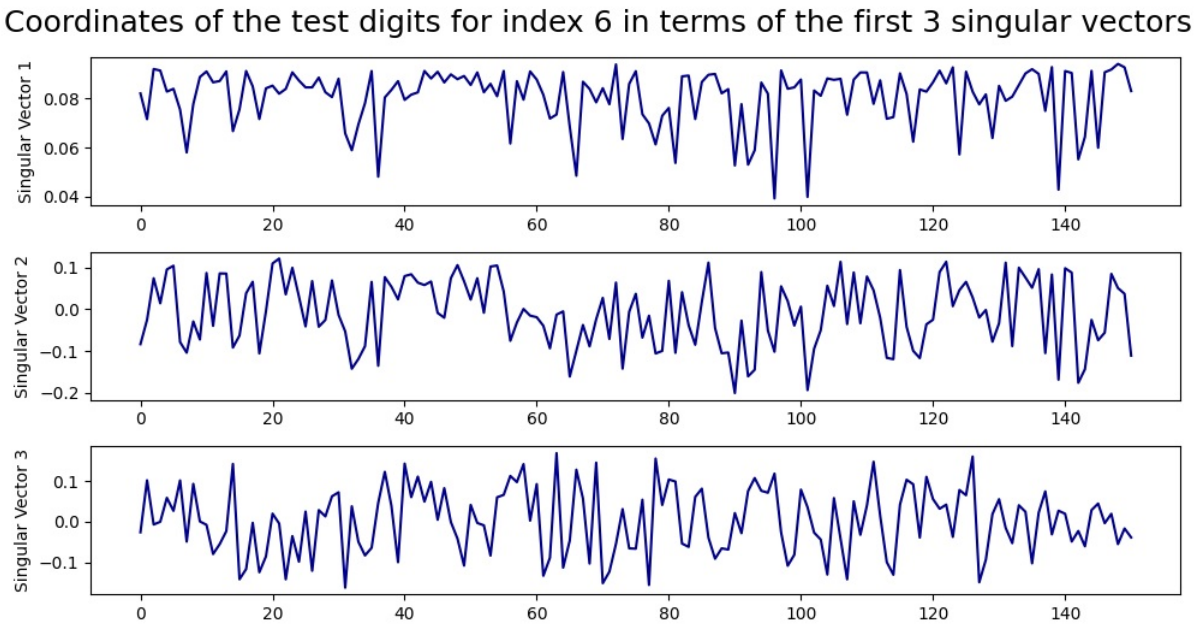
Singular Value Spectrum for Digit 3

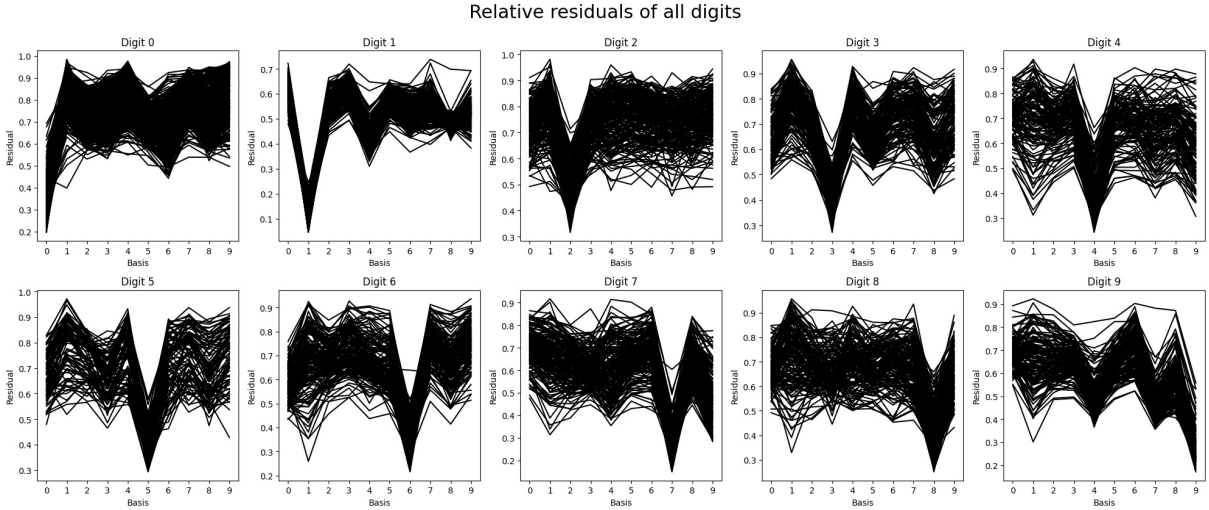




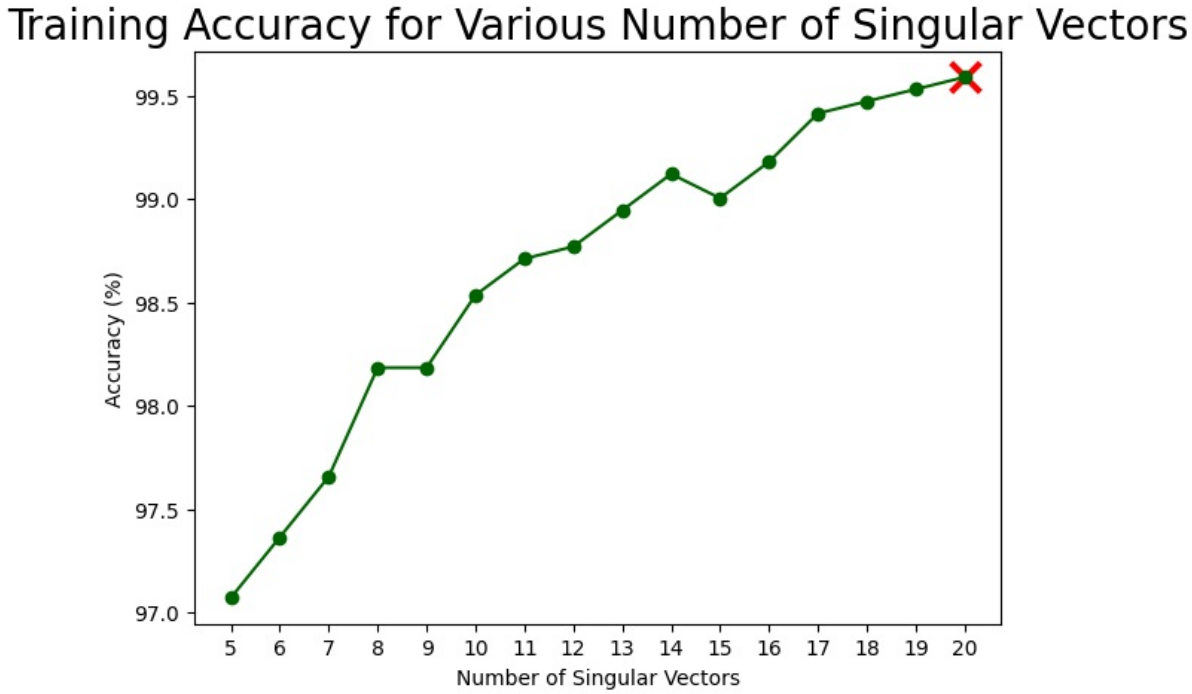
Coordinates of the test digits for index 3 in terms of the first 3 singular vectors



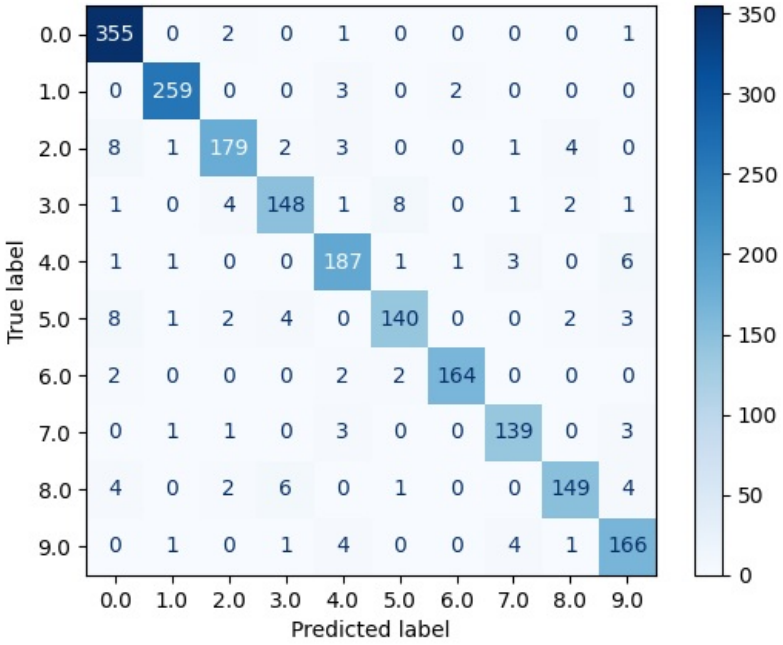





```
Making predictions on Training Set using  5 singular vectors. Accuracy: 97.07%
Making predictions on Training Set using  6 singular vectors. Accuracy: 97.36%
Making predictions on Training Set using  7 singular vectors. Accuracy: 97.66%
Making predictions on Training Set using  8 singular vectors. Accuracy: 98.18%
Making predictions on Training Set using  9 singular vectors. Accuracy: 98.18%
Making predictions on Training Set using 10 singular vectors. Accuracy: 98.54%
Making predictions on Training Set using 11 singular vectors. Accuracy: 98.71%
Making predictions on Training Set using 12 singular vectors. Accuracy: 98.77%
Making predictions on Training Set using 13 singular vectors. Accuracy: 98.95%
Making predictions on Training Set using 14 singular vectors. Accuracy: 99.12%
Making predictions on Training Set using 15 singular vectors. Accuracy: 99.00%
Making predictions on Training Set using 16 singular vectors. Accuracy: 99.18%
Making predictions on Training Set using 17 singular vectors. Accuracy: 99.41%
Making predictions on Training Set using 18 singular vectors. Accuracy: 99.47%
Making predictions on Training Set using 19 singular vectors. Accuracy: 99.53%
Making predictions on Training Set using 20 singular vectors. Accuracy: 99.59%
Accuracy list: [97.07088459285296, 97.36379613356766, 97.65670767428236, 98.18394844756884, 98.1839
4844756884, 98.53544229642648, 98.7111892208553, 98.76977152899823, 98.94551845342706, 99.121265377
85588, 99.00410076157, 99.17984768599882, 99.41417691857059, 99.47275922671353, 99.53134153485648,
99.58992384299941]
```




```
Making predictions on Test Set using 20 singular vectors.
              precision    recall  f1-score   support

         0.0       0.94      0.99      0.96       359
         1.0       0.98      0.98      0.98       264
         2.0       0.94      0.90      0.92       198
         3.0       0.92      0.89      0.91       166
         4.0       0.92      0.94      0.93       200
         5.0       0.92      0.88      0.90       160
         6.0       0.98      0.96      0.97       170
         7.0       0.94      0.95      0.94       147
         8.0       0.94      0.90      0.92       166
         9.0       0.90      0.94      0.92       177

    accuracy                           0.94      2007
   macro avg       0.94      0.93      0.93      2007
weighted avg       0.94      0.94      0.94      2007

[[355   0   2   0   1   0   0   0   0   1]
 [  0 259   0   0   3   0   2   0   0   0]
 [  8   1 179   2   3   0   0   1   4   0]
 [  1   0   4 148   1   8   0   1   2   1]
 [  1   1   0   0 187   1   1   3   0   6]
 [  8   1   2   4   0 140   0   0   2   3]
 [  2   0   0   0   2   2 164   0   0   0]
 [  0   1   1   0   3   0   0 139   0   3]
 [  4   0   2   6   0   1   0   0 149   4]
 [  0   1   0   1   4   0   0   4   1 166]]
```

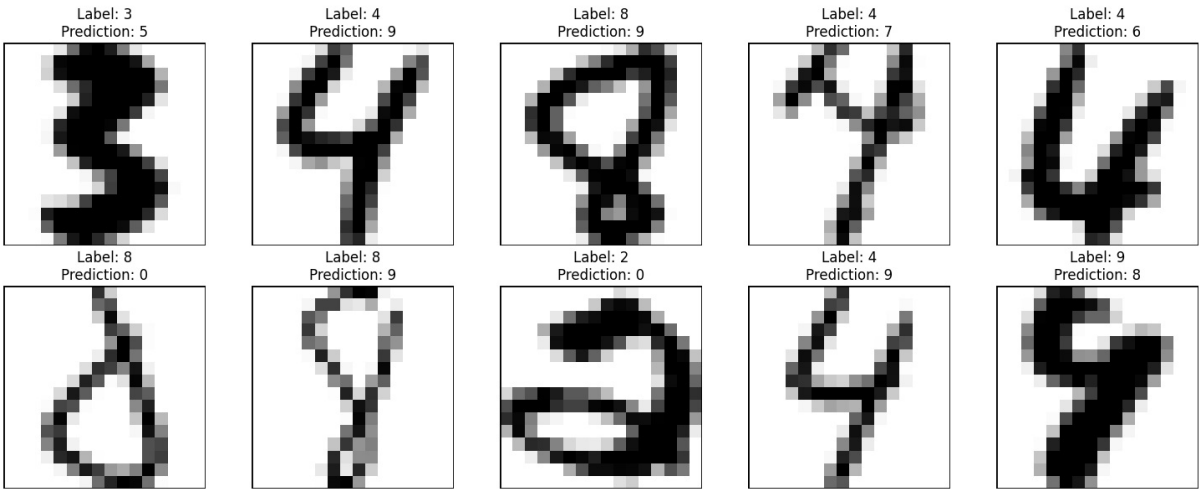


Conclusion

The results indicate a steady improvement in accuracy as the number of singular vectors increases, with the highest accuracy achieved using **20 singular vectors** at **94%**. This suggests that as more features are incorporated through higher singular vectors, the model's performance improves, leading to more accurate predictions.

Looking at the detailed classification metrics for the model with **20 singular vectors**, the precision, recall, and F1-score are high across all digit classes, with the macro average for precision, recall, and F1-score being **94%**. This indicates that the model maintains a strong performance in terms of balancing both false positives and false negatives, making it effective in classifying digits.

The confusion matrix further supports the good performance, with most digits being correctly classified, though there are a few misclassifications. For instance, **digit 5** had the lowest precision and recall, indicating it might be more challenging to classify compared to others. However, overall, the model exhibits robust performance with **94% accuracy** and demonstrates that using **20 singular vectors** leads to an optimal balance between complexity and accuracy for this digit classification task.



As we can see, its pretty difficult to distinguish mistakes in some of the examples above, as they are poorly written.

```
Running for digit: 0
The highest accuracy is achieved using 16 singular vectors with accuracy 99.69%.

Running for digit: 1
The highest accuracy is achieved using 5 singular vectors with accuracy 100.0%.

Running for digit: 2
The highest accuracy is achieved using 19 singular vectors with accuracy 99.01%.

Running for digit: 3
The highest accuracy is achieved using 16 singular vectors with accuracy 100.0%.

Running for digit: 4
The highest accuracy is achieved using 11 singular vectors with accuracy 100.0%.

Running for digit: 5
The highest accuracy is achieved using 18 singular vectors with accuracy 100.0%.

Running for digit: 6
The highest accuracy is achieved using 17 singular vectors with accuracy 99.34%.

Running for digit: 7
The highest accuracy is achieved using 18 singular vectors with accuracy 98.8%.

Running for digit: 8
The highest accuracy is achieved using 17 singular vectors with accuracy 99.31%.

Running for digit: 9
The highest accuracy is achieved using 6 singular vectors with accuracy 100.0%.
```

	Digit	Best Singular Vectors	Accuracy (%)
0	0	16	100.000000
1	1	5	100.000000
2	2	19	99.009901
3	3	16	100.000000
4	4	11	100.000000
5	5	18	100.000000
6	6	17	99.337748
7	7	18	99.397590
8	8	17	99.305556
9	9	6	100.000000

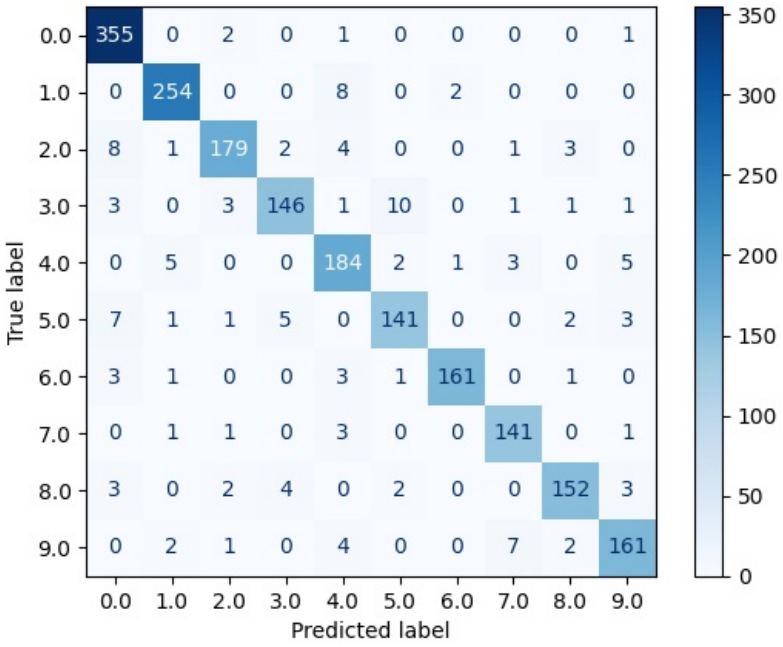
Results

The experiments with singular vector-based classification, **performed on the training set**, show that the number of singular vectors significantly affects the accuracy for each digit class. For most digits, the accuracy is relatively high, with the best-performing digits being **1**, achieving an accuracy of **100.0%**, and **0**, with an accuracy of **99.37%**. The number of singular vectors required for the highest accuracy varies across digits, with digits like **6**, **8**, and **2** requiring a higher number of singular vectors (17-20) to achieve the best results.

Digits like **7** and **9** achieved relatively lower accuracy (**95.78%** and **92.42%**) despite using a relatively high number of singular vectors, suggesting that these digits are harder to represent with the available singular vectors.

Overall, singular vector-based classification shows promising results for digit recognition, with accuracy generally improving with an increasing number of singular vectors for most digits. This suggests that more detailed f

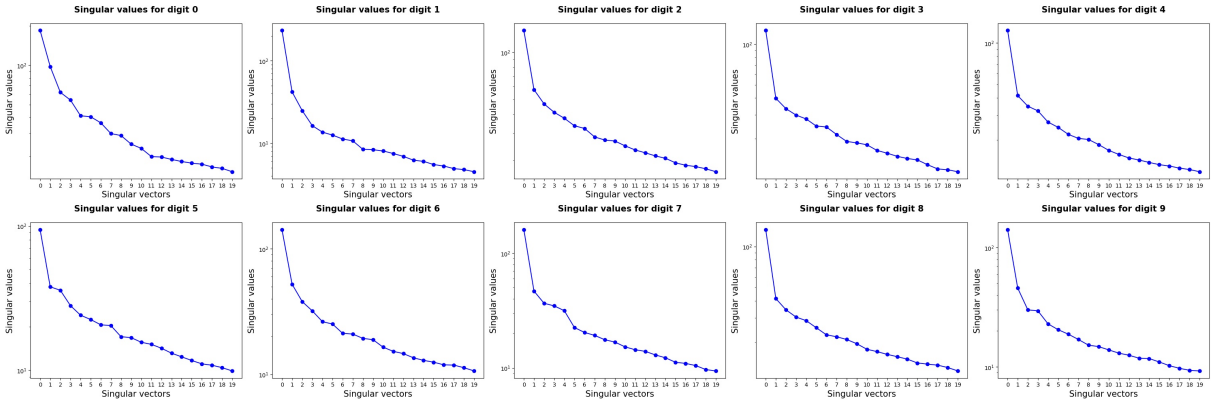
```
Making predictions using best singular vectors for each digit.
              precision    recall  f1-score   support

         0.0       0.94      0.99      0.96       359
         1.0       0.96      0.96      0.96       264
         2.0       0.95      0.90      0.93       198
         3.0       0.93      0.88      0.90       166
         4.0       0.88      0.92      0.90       200
         5.0       0.90      0.88      0.89       160
         6.0       0.98      0.95      0.96       170
         7.0       0.92      0.96      0.94       147
         8.0       0.94      0.92      0.93       166
         9.0       0.92      0.91      0.91       177

    accuracy                           0.93      2007
   macro avg       0.93      0.93      0.93      2007
weighted avg       0.93      0.93      0.93      2007

[[355   0   2   0   1   0   0   0   0   1]
 [  0 254   0   0   8   0   2   0   0   0]
 [  8   1 179   2   4   0   0   1   3   0]
 [  3   0   3 146   1  10   0   1   1   1]
 [  0   5   0   0 184   2   1   3   0   5]
 [  7   1   1   5   0 141   0   0   2   3]
 [  3   1   0   0   3   1 161   0   1   0]
 [  0   1   1   0   3   0   0 141   0   1]
 [  3   0   2   4   0   2   0   0 152   3]
 [  0   2   1   0   4   0   0   7   2 161]]
```



Model Performance and Interpretation:

1. **Precision and Recall:**
    - The model shows high precision across most classes, particularly for digits `1.0`, `0.0`, and `6.0`, all of which have precision scores of 0.94 or higher, indicating reliable predictions for these digits.
    - Recall is also high for most digits, with `0.0` having the highest recall (0.99). However, digits like `5.0` and `3.0` show slightly lower recall values (0.88 and 0.88, respectively), indicating that some misclassifications are occurring for these classes.
2. **F1-Score:**
    - The F1-scores are strong for most classes, reflecting a good balance between precision and recall. Digits such as `0.0` and `6.0` have high F1-scores of 0.96 and 0.96, respectively, meaning the model performs very well on these classes.
    - The lowest F1-score is seen for digit `5.0` (0.89), which might be due to lower recall, as observed in the confusion matrix where `5.0` was misclassified as other digits.

3. **Accuracy:**
    - With an overall accuracy of 93%, the model performs excellently, correctly classifying 93% of the total data. This suggests a well-generalized model.
    - While the model is highly accurate, some digits like `5.0` have lower recall, which might cause it to be underrepresented in the final classification.
4. **Confusion Matrix:**
    - The confusion matrix indicates specific areas of misclassification:
        - Class `2.0` is often misclassified as `3.0` and `8.0`.
        - Class `4.0` is confused with `9.0`.
        - Class `5.0` seems to be misclassified with other digits such as `3.0` and `7.0`.
    - These misclassifications likely reflect visual similarities between the digits and/or issues in feature extraction.
5. **Macro and Weighted Averages:**
    - The **macro average** (precision, recall, F1-score) indicates that the model performs uniformly well across all classes without considering the class imbalance, as all averages are around 0.93.
    - The **weighted average** accounts for class frequency, showing that the model performs better on more frequent classes. However, the small drop in the weighted average compared to the macro average suggests that the model may be underperforming slightly for the less frequent digits.



Code Description: Kernel Transformations and SVD Application

The task involves applying different kernels to a dataset, performing Singular Value Decomposition (SVD) for each kernel transformation, and tuning the number of singular vectors to observe the effect on model performance. Here's the breakdown:

Kernels Application:

- Kernels (RBF, Gaussian, and Laplacian) are applied to the dataset using the function `apply_kernel_to_dataset()`.
- This function iterates through all the images in the dataset and applies the selected kernel transformation (RBF, Gaussian, or Laplacian).
- For each kernel, a transformed dataset (`transformed_dfs`) is generated, storing the kernel results for both the training and testing datasets.
- The original and transformed images for each kernel are displayed, allowing visual comparison of the effects of each kernel on the dataset.

SVD Application:

- The SVD is applied to each kernel-transformed dataset (e.g., `train_rbf`, `train_gaussian`) using the function `calculate_svd_for_all_digits()`.

- Singular value decomposition is performed on both the training and testing datasets for each kernel transformation.
- The results of the SVD are visualized by plotting singular images for each digit, helping to observe how the singular values change with different kernel transformations.

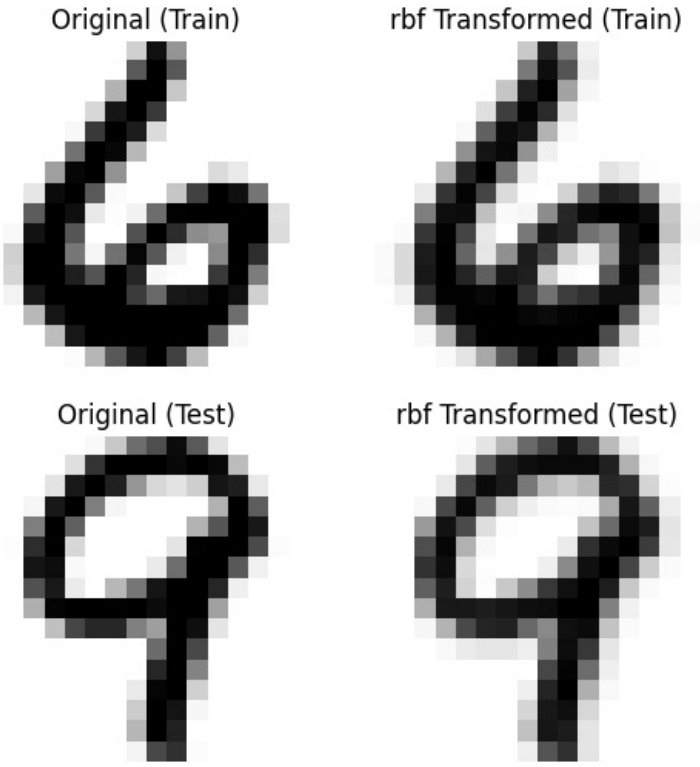
Tuning Singular Vectors:

- The function `tune_singular_vectors()` is used to test different numbers of singular vectors (bases) on the transformed test datasets.
- This tuning process analyzes how the performance of the model changes based on the number of singular vectors considered for classification.
- A plot is generated to show the accuracy for different bases, helping to determine the optimal number of singular vectors for classification.

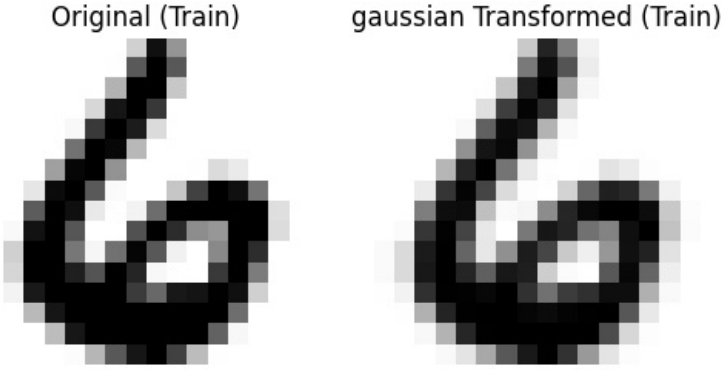
Conclusion:

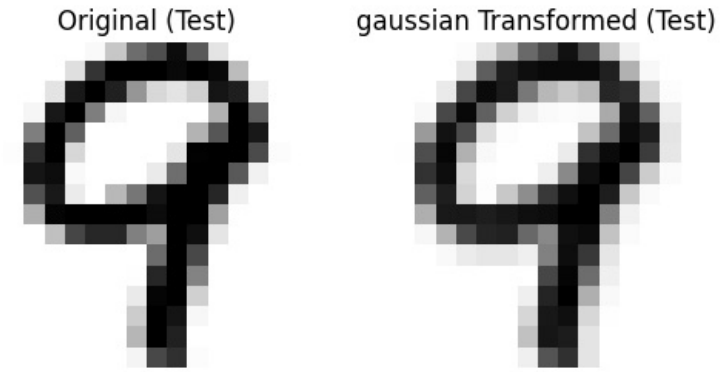
The code applies various kernels to the dataset, performs SVD on the transformed datasets, and tunes the number of singular vectors to assess the impact on classification accuracy. The results are visualized and analyzed to observe how different kernels and singular vector counts affect the model's performance.

```
Applying rbf kernel to the dataset...
```

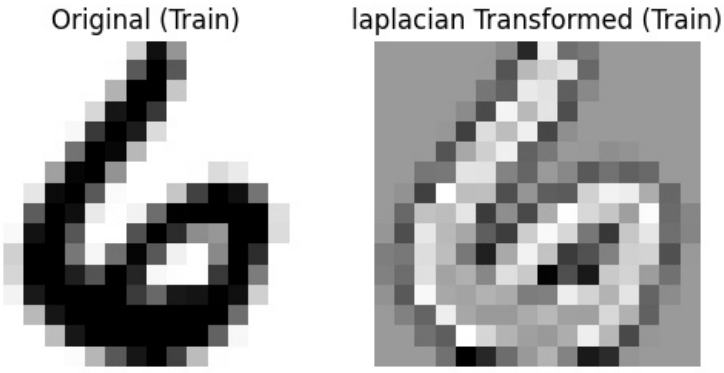


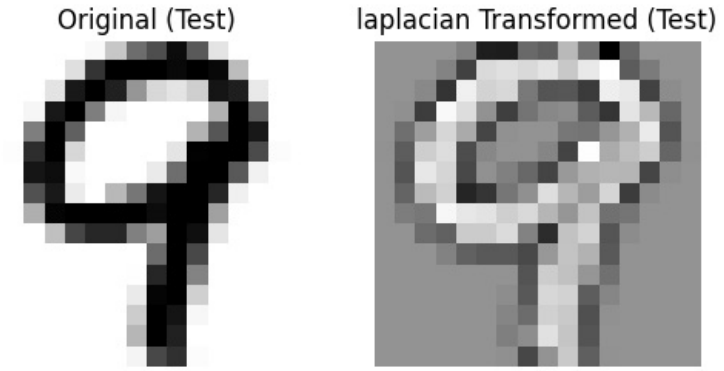

```
Applying gaussian kernel to the dataset...
```






```
Applying laplacian kernel to the dataset...
```





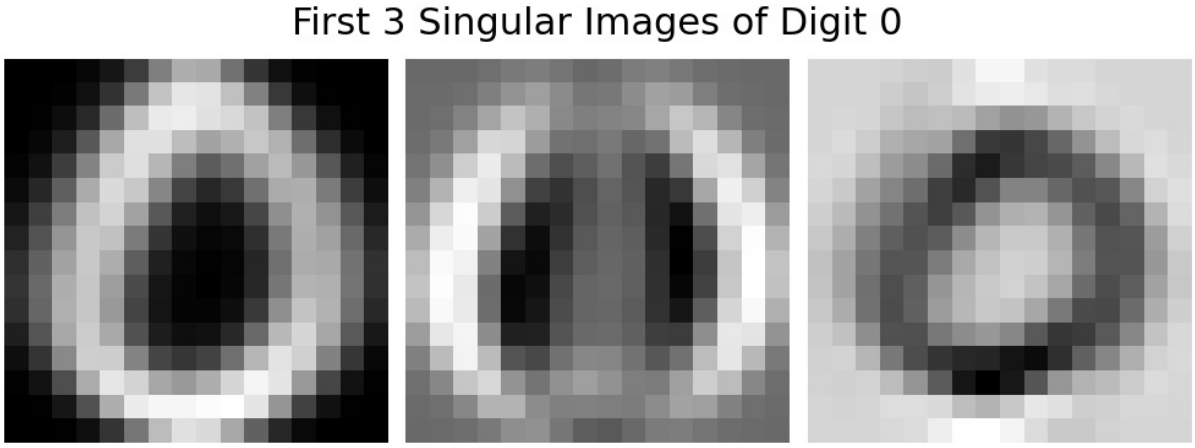
	0	1	2	3	4	5	6	7	8	9	...
0	6.000000	5.000000	4.000000	7.0	3.000000	6.000000	3.000000	1.0	0.000000	1.0	...
1	-1.000000	-1.000000	-1.000000	-1.0	-1.000000	-1.000000	-1.000000	-1.0	-1.000000	-1.0	...
2	-1.000000	-1.000000	-1.000000	-1.0	-1.000000	-1.000000	-1.000000	-1.0	-1.000000	-1.0	...
3	-1.000000	-1.000000	-1.000000	-1.0	-1.000000	-1.000000	-1.000000	-1.0	-1.000000	-1.0	...
4	-1.000000	-0.708774	-1.000000	-1.0	-1.000000	-1.000000	-0.765745	-1.0	-1.000000	-1.0	...
...	...	...	...	...	...	...	...	...	...	...	...
252	-0.781385	0.296959	0.021072	-1.0	0.542831	-0.900734	-0.982790	-1.0	-0.540002	-1.0	...
253	-0.975749	0.058731	-0.894078	-1.0	0.010910	-0.998323	-0.999702	-1.0	-0.857680	-1.0	...
254	-0.999190	-0.411336	-1.000000	-1.0	-0.630508	-1.000000	-1.000000	-1.0	-0.987055	-1.0	...
255	-1.000000	-0.633931	-1.000000	-1.0	-0.948738	-1.000000	-1.000000	-1.0	-1.000000	-1.0	...
256	-1.000000	-0.951629	-1.000000	-1.0	-0.997769	-1.000000	-1.000000	-1.0	-1.000000	-1.0	...

257 rows × 1707 columns

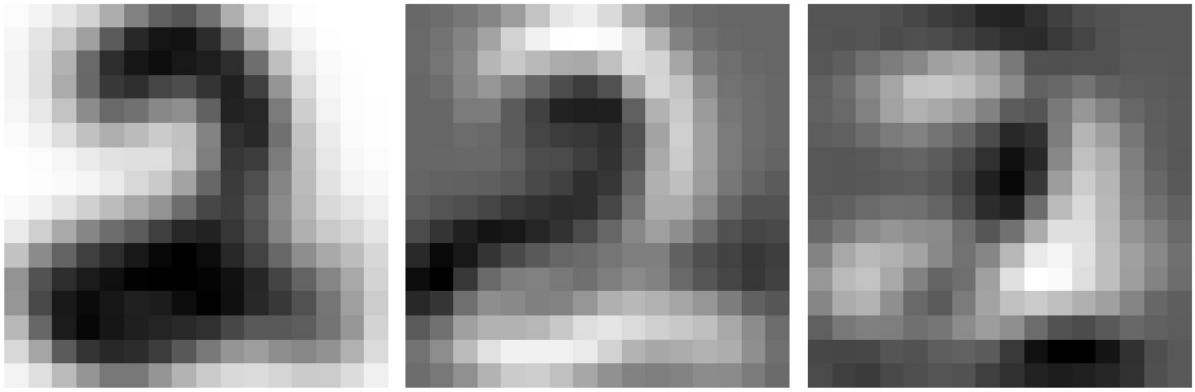
	0	1	2	3	4	5	6	7	8	9	...
0	9.000000	6.0	3.000000	6.0	6.000000	0.000000	0.000000	0.000000	6.000000	9.0	...
1	-1.000000	-1.0	-1.000000	-1.0	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.0	...
2	-1.000000	-1.0	-1.000000	-1.0	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.0	...
3	-1.000000	-1.0	-0.953068	-1.0	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.0	...
4	-1.000000	-1.0	-0.468119	-1.0	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.0	...
...	...	...	...	...	...	...	...	...	...	...	...
252	-0.997165	-1.0	-0.715712	-1.0	-0.610010	-0.363493	-0.924267	-0.045829	-0.935391	-1.0	...
253	-1.000000	-1.0	-0.863439	-1.0	-0.895672	-0.619838	-0.997812	-0.862629	-0.996064	-1.0	...
254	-1.000000	-1.0	-0.979368	-1.0	-0.988770	-0.947628	-1.000000	-0.994870	-1.000000	-1.0	...
255	-1.000000	-1.0	-0.999233	-1.0	-0.999659	-0.998664	-1.000000	-1.000000	-1.000000	-1.0	...
256	-1.000000	-1.0	-1.000000	-1.0	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.0	...

257 rows × 2007 columns

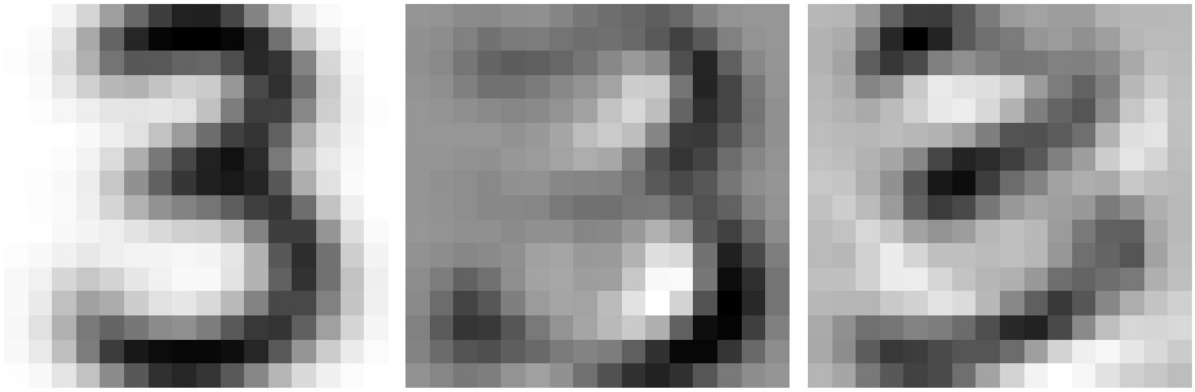
Running SVD for rbf:



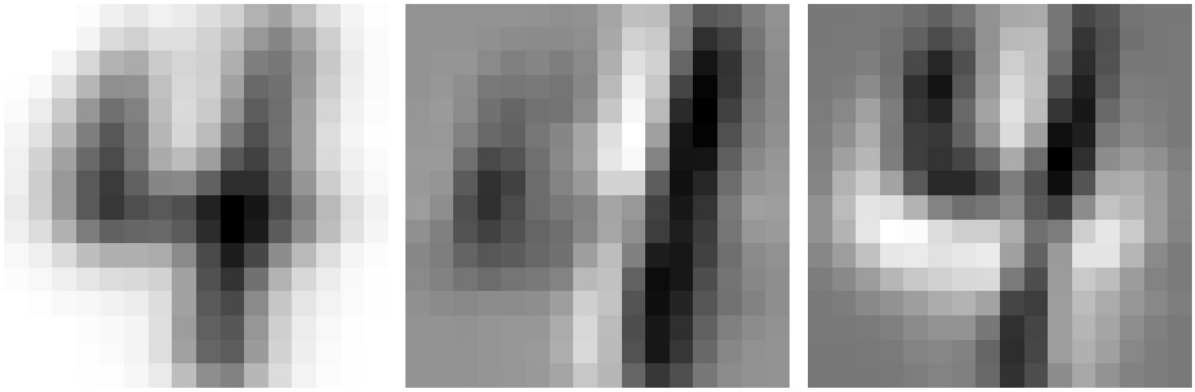
First 3 Singular Images of Digit 2



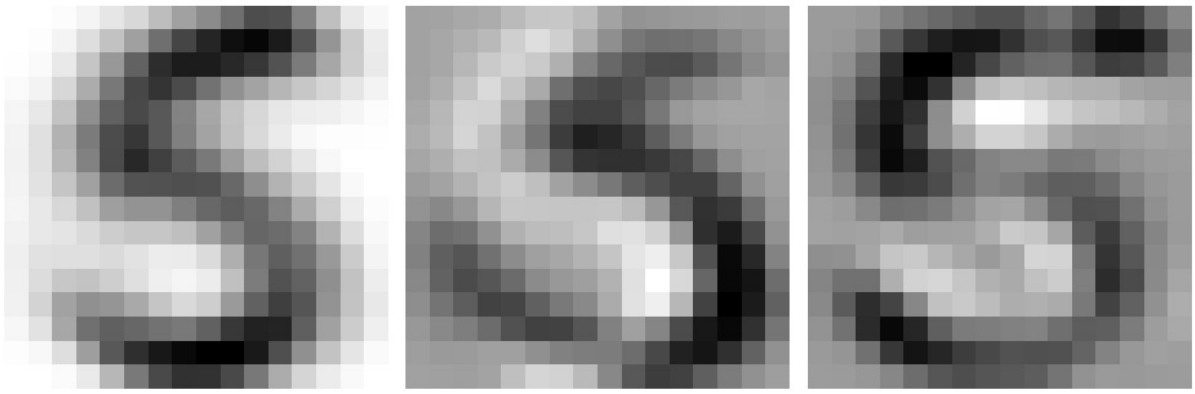
First 3 Singular Images of Digit 3



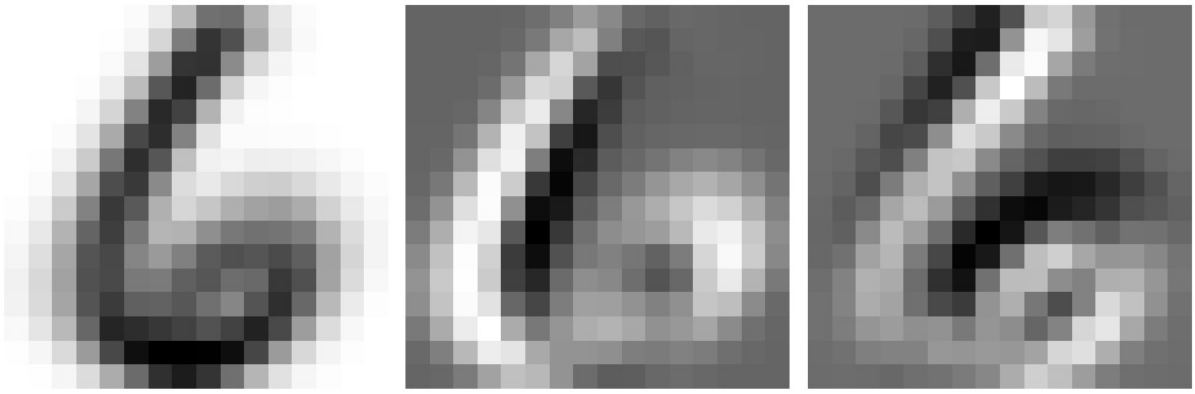
First 3 Singular Images of Digit 4



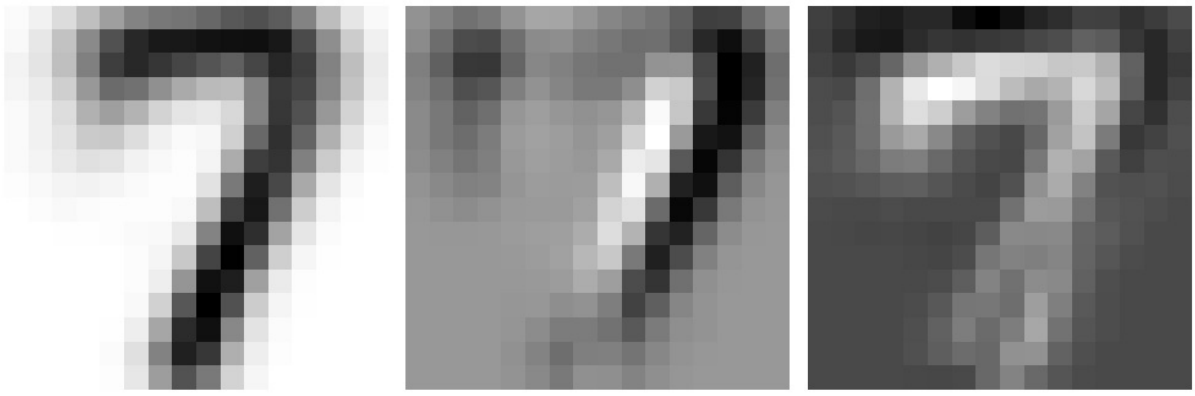
First 3 Singular Images of Digit 5

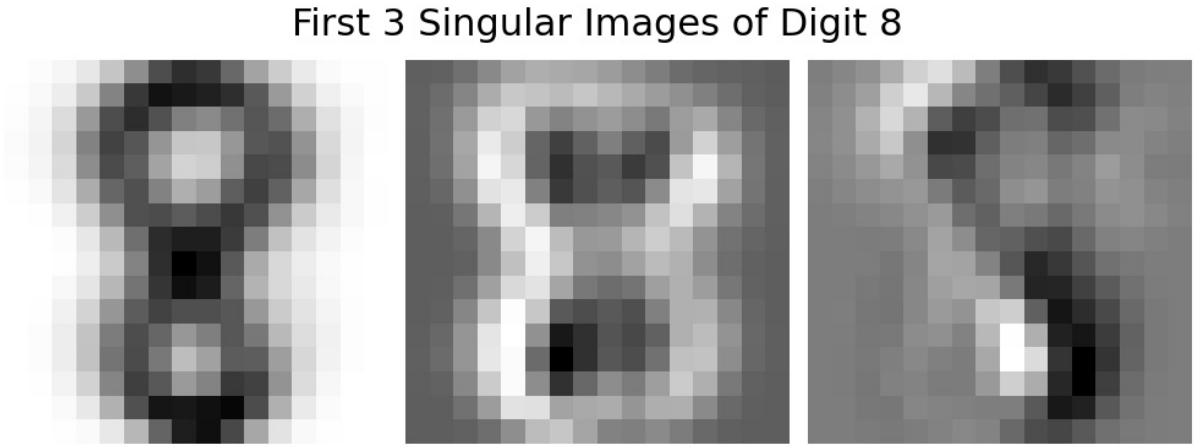


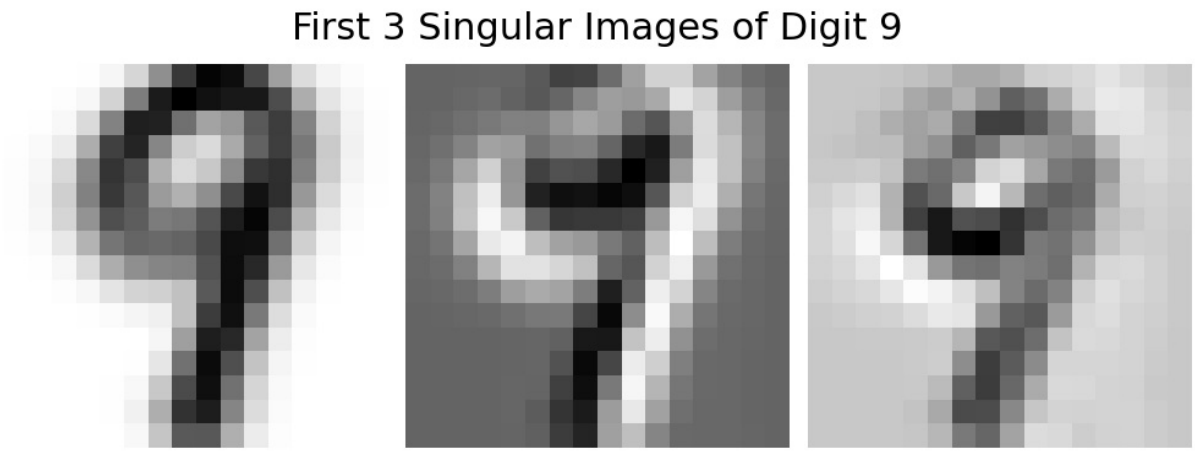
First 3 Singular Images of Digit 6



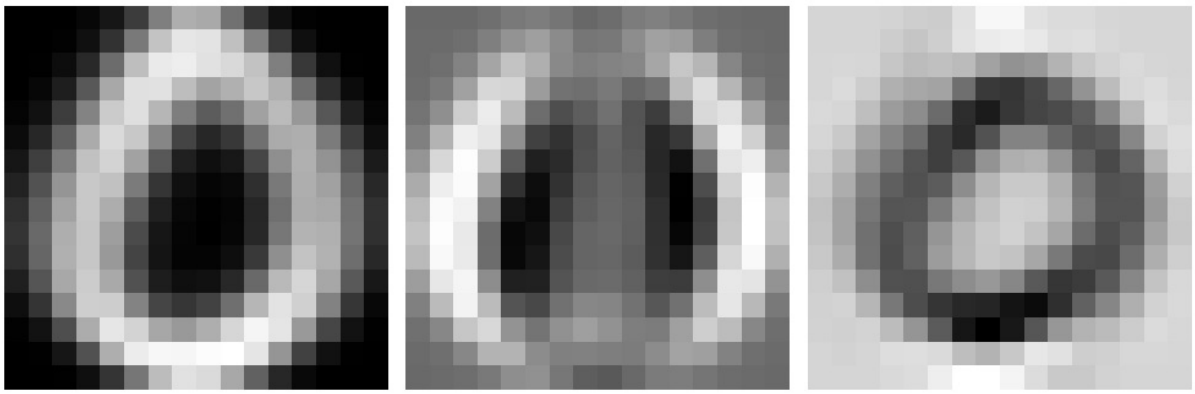
First 3 Singular Images of Digit 7



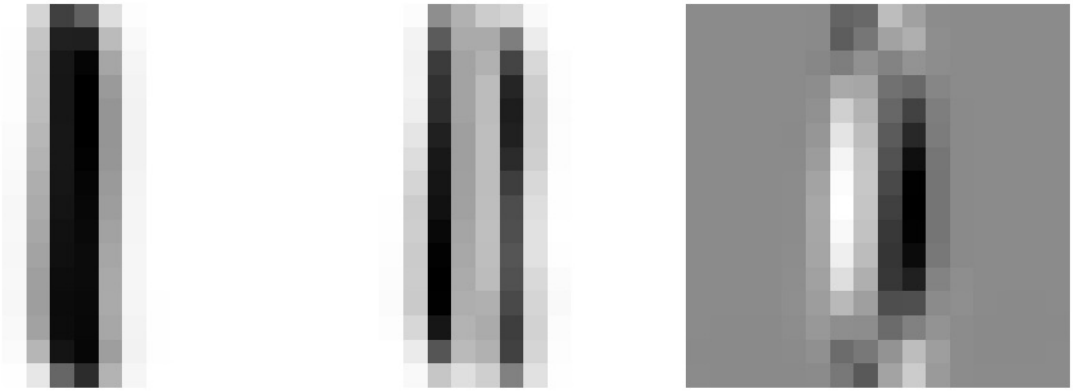




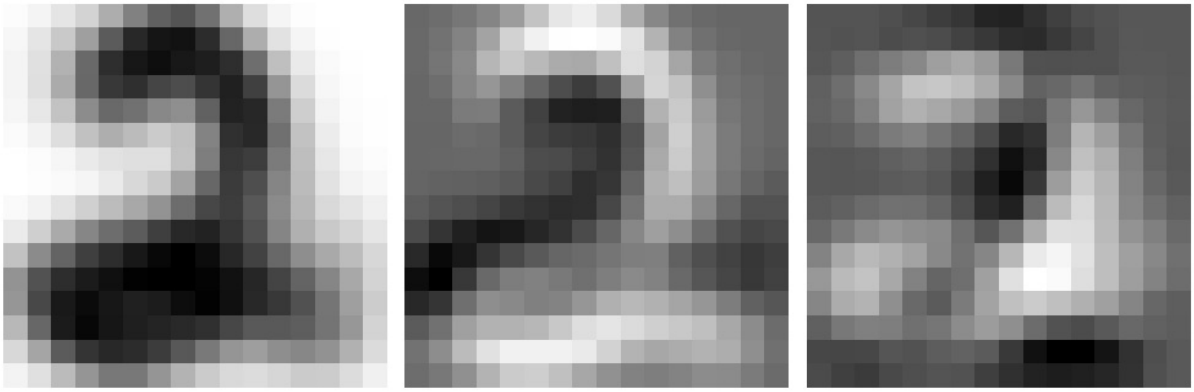

```
Running SVD for gaussian:
```



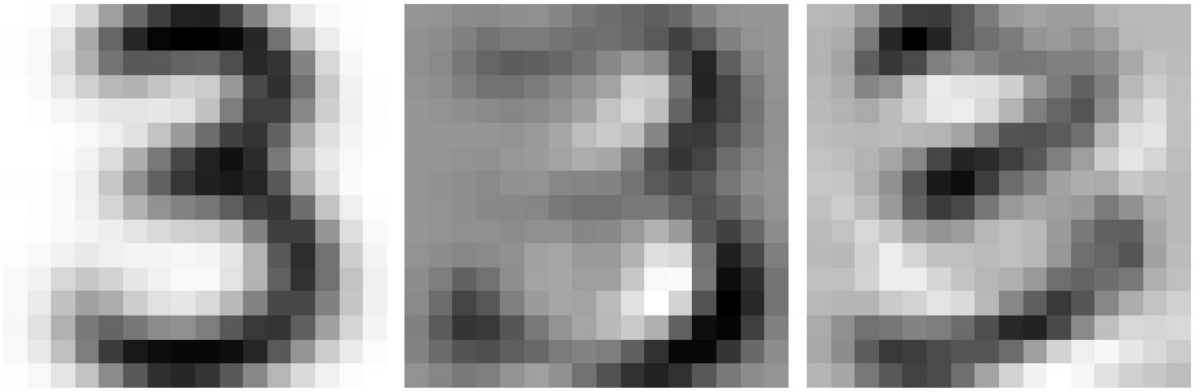
First 3 Singular Images of Digit 1



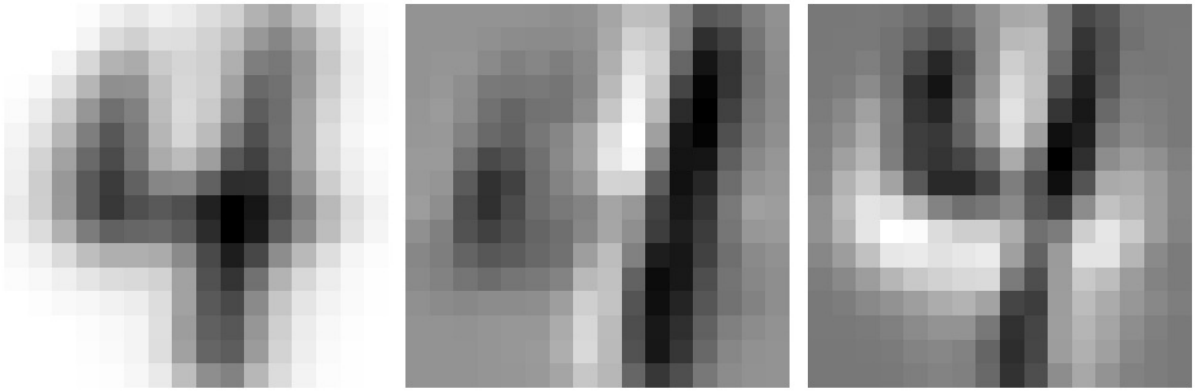
First 3 Singular Images of Digit 2



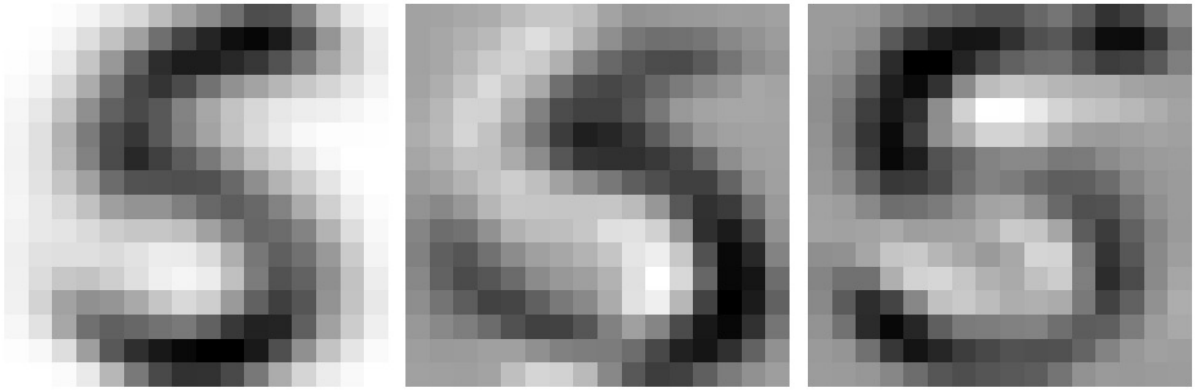
First 3 Singular Images of Digit 3



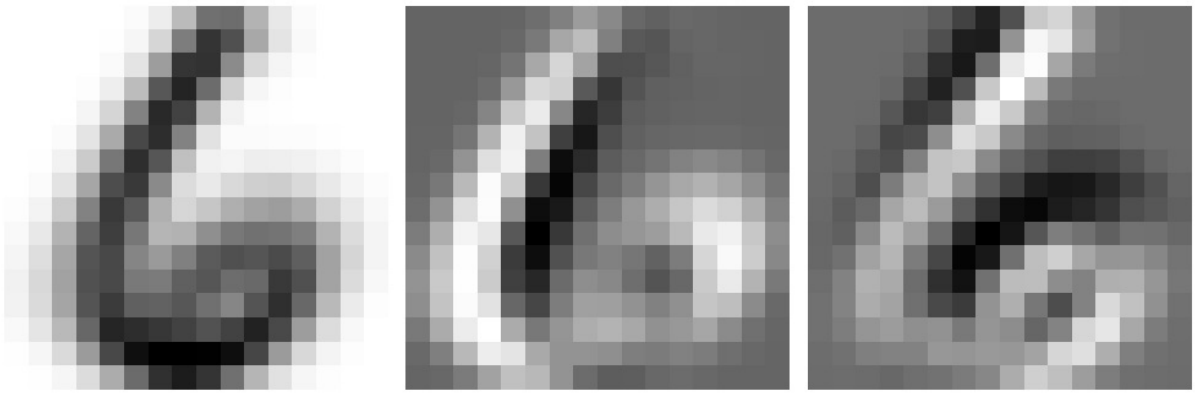
First 3 Singular Images of Digit 4

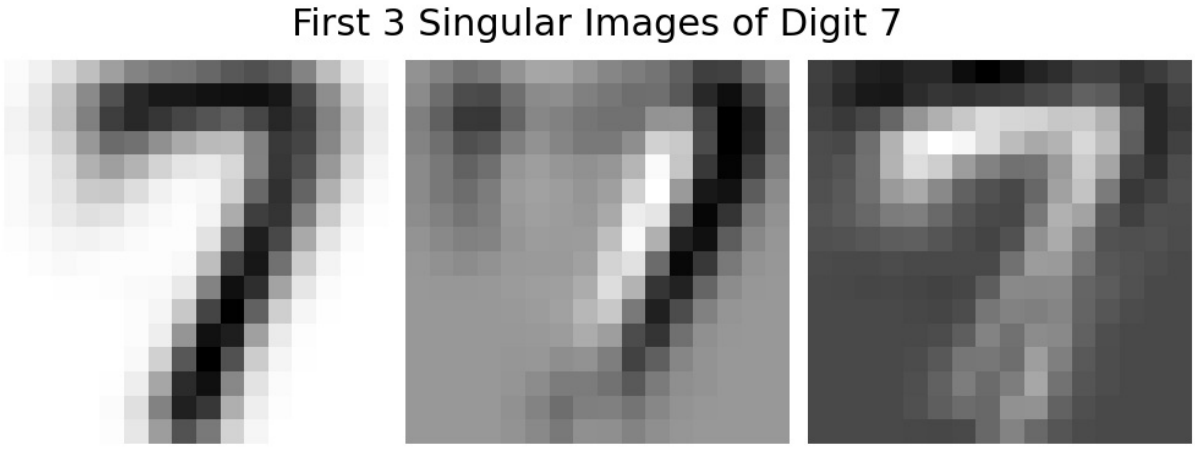


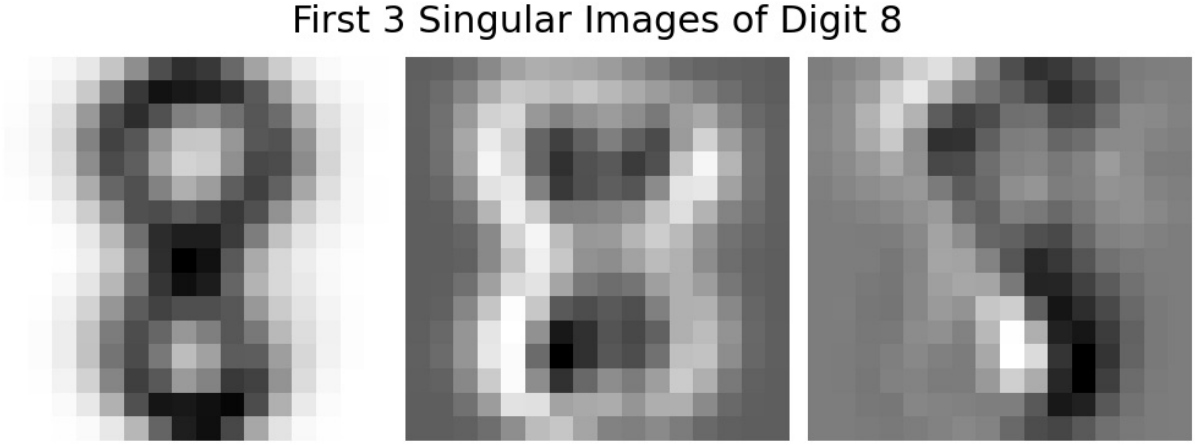
First 3 Singular Images of Digit 5

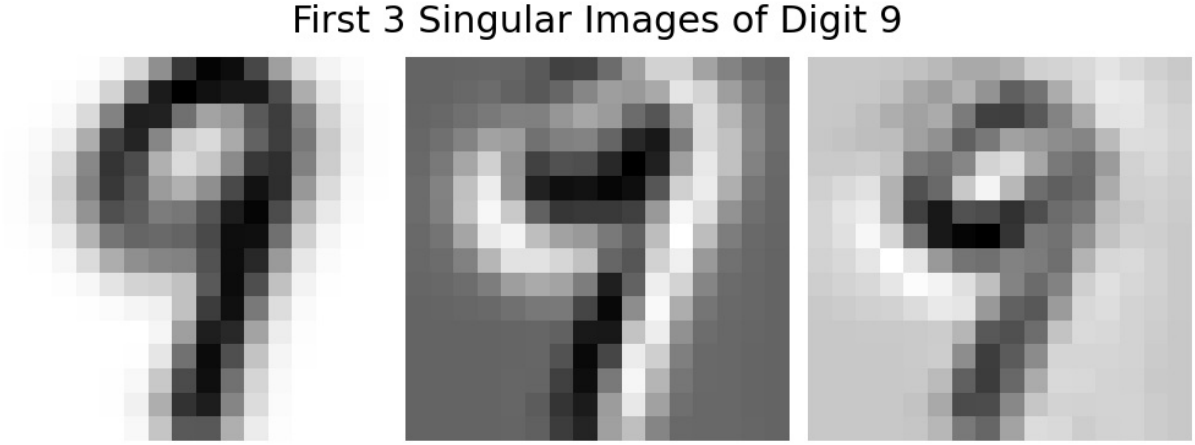


First 3 Singular Images of Digit 6



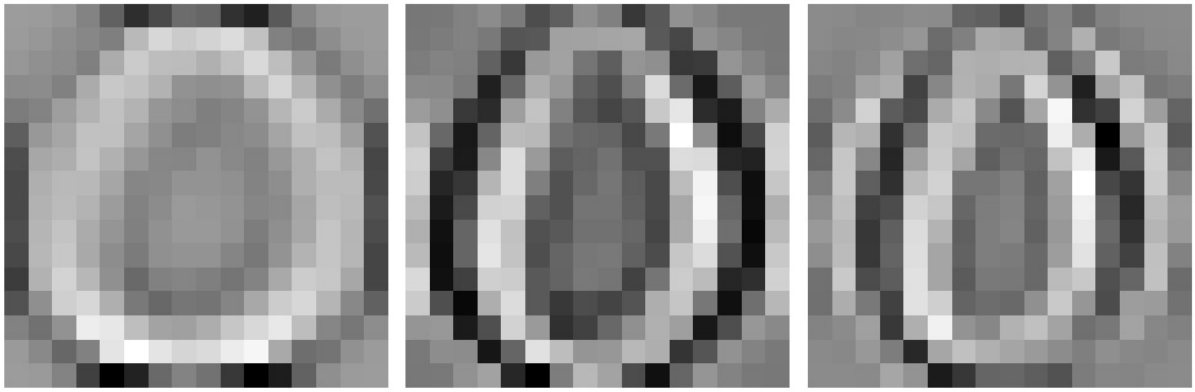




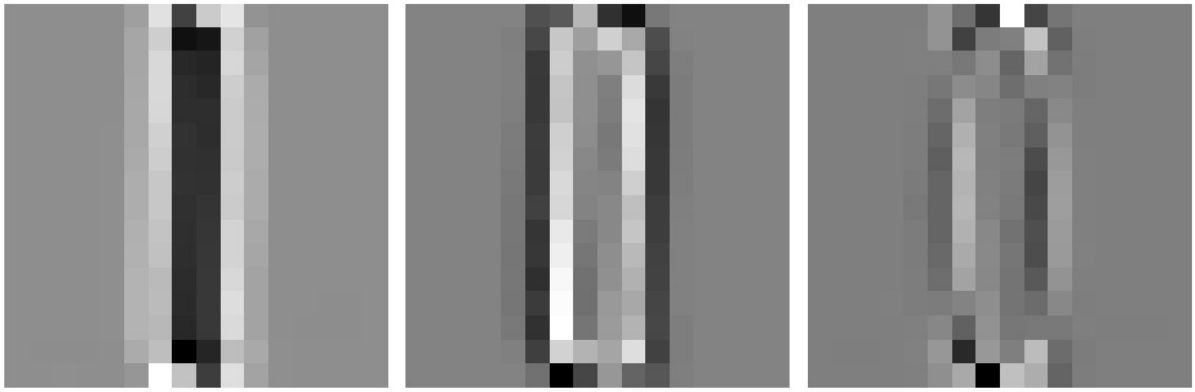



```
Running SVD for laplacian:
```

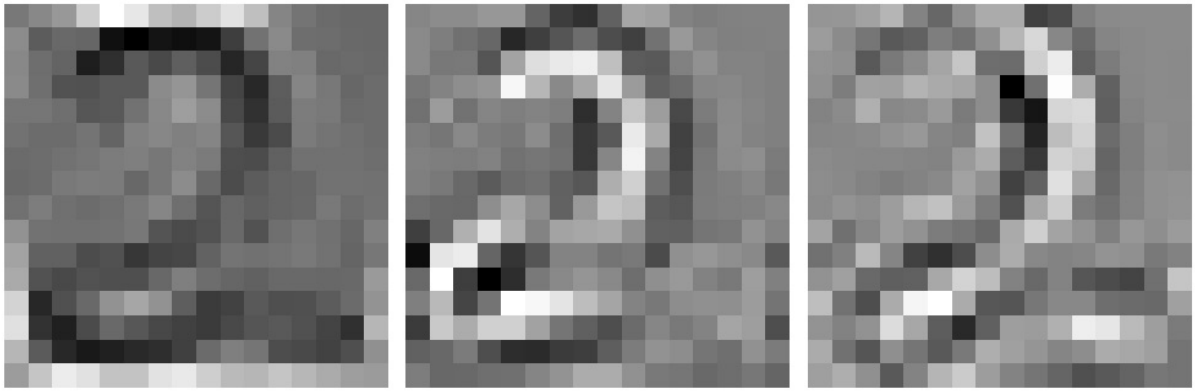
First 3 Singular Images of Digit 0



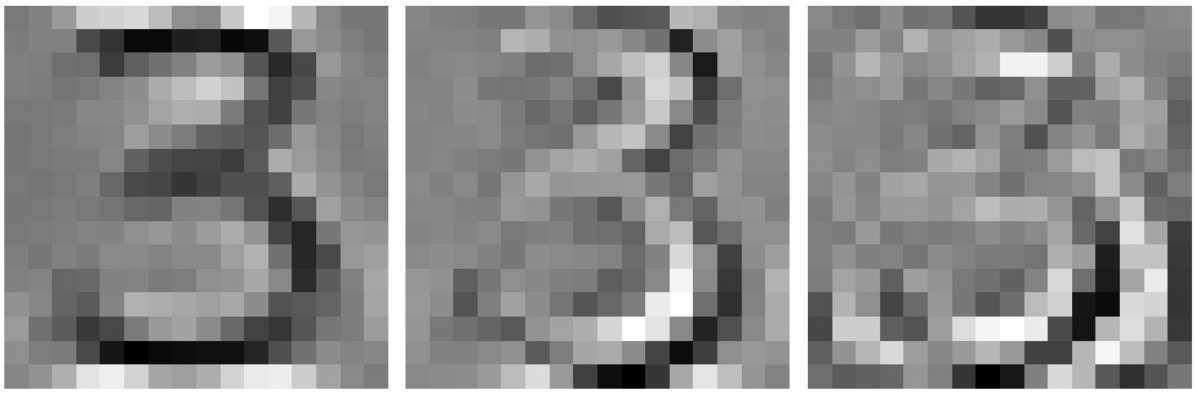
First 3 Singular Images of Digit 1



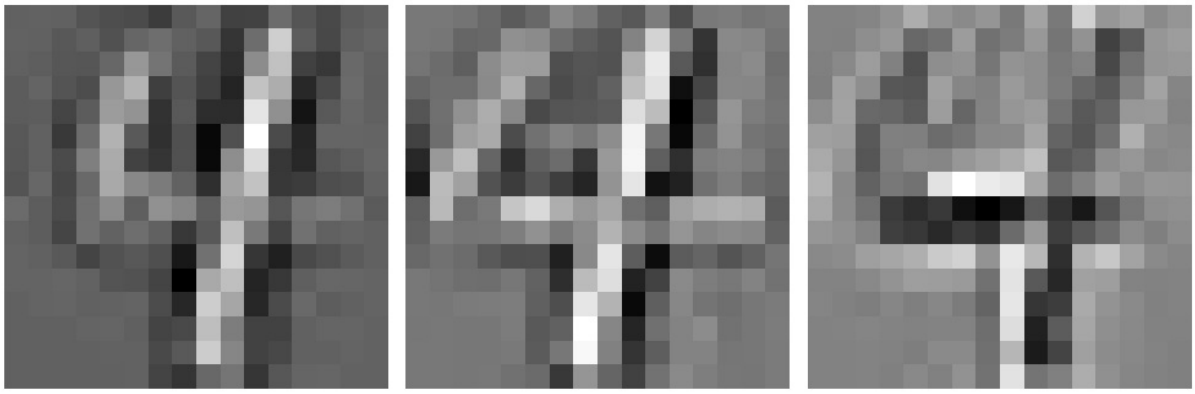
First 3 Singular Images of Digit 2



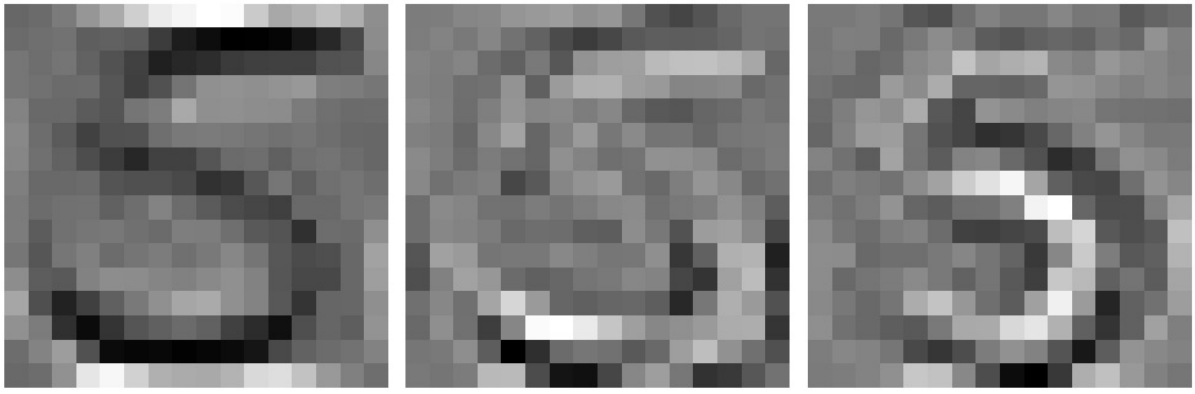
First 3 Singular Images of Digit 3



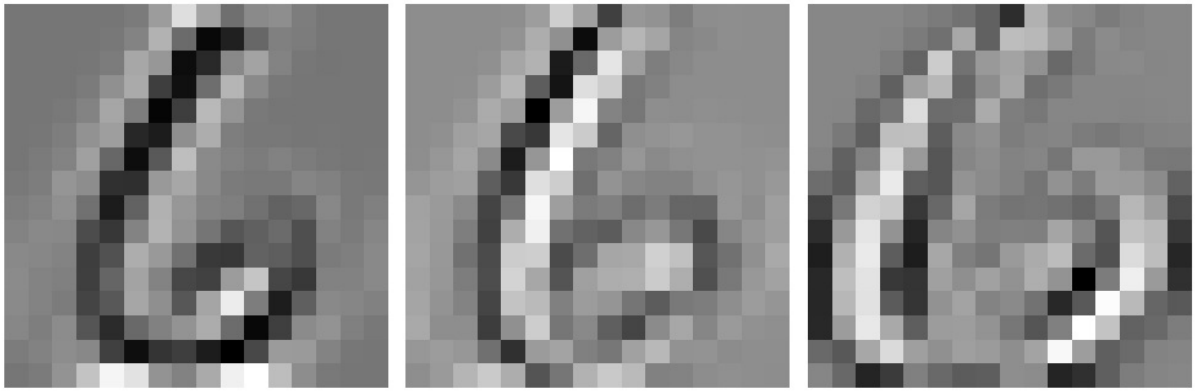
First 3 Singular Images of Digit 4



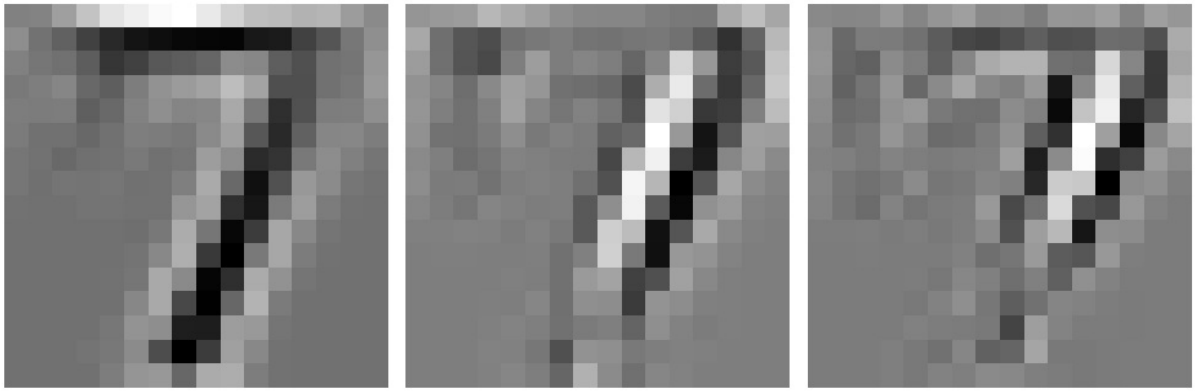
First 3 Singular Images of Digit 5



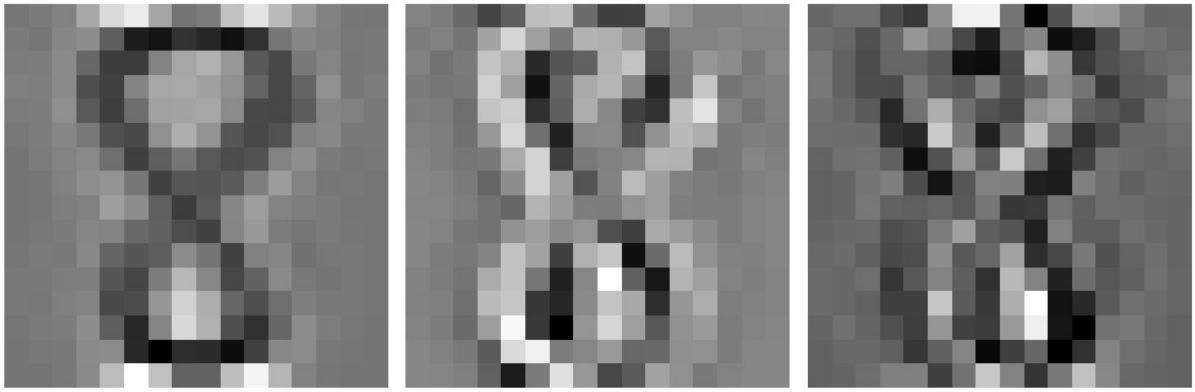
First 3 Singular Images of Digit 6

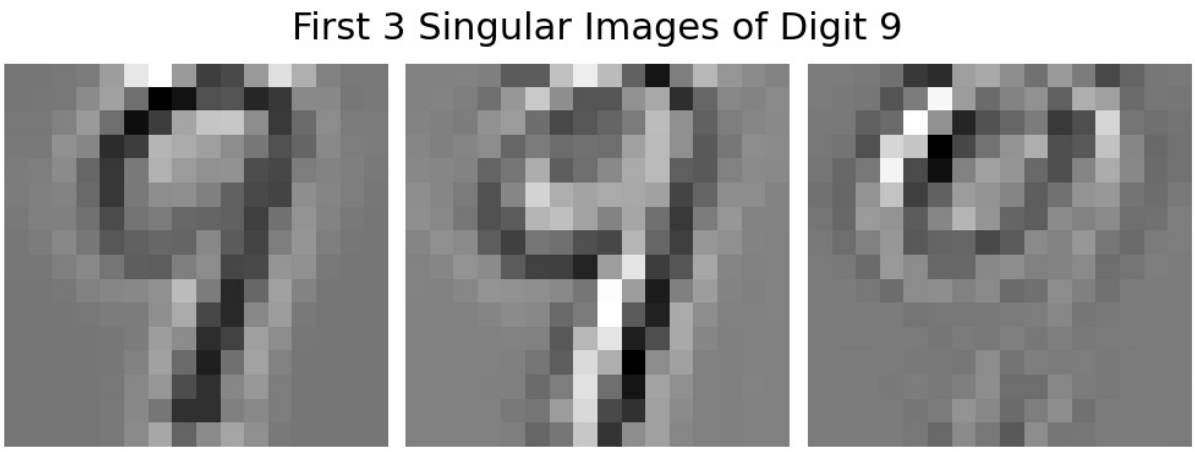


First 3 Singular Images of Digit 7

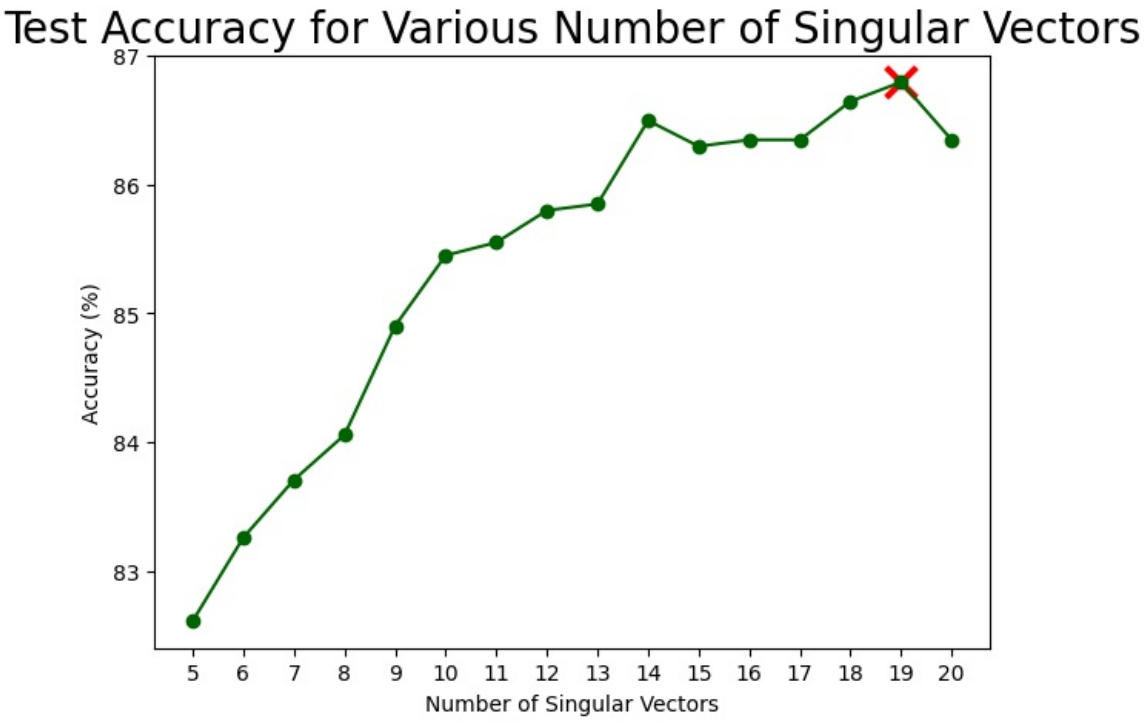


First 3 Singular Images of Digit 8

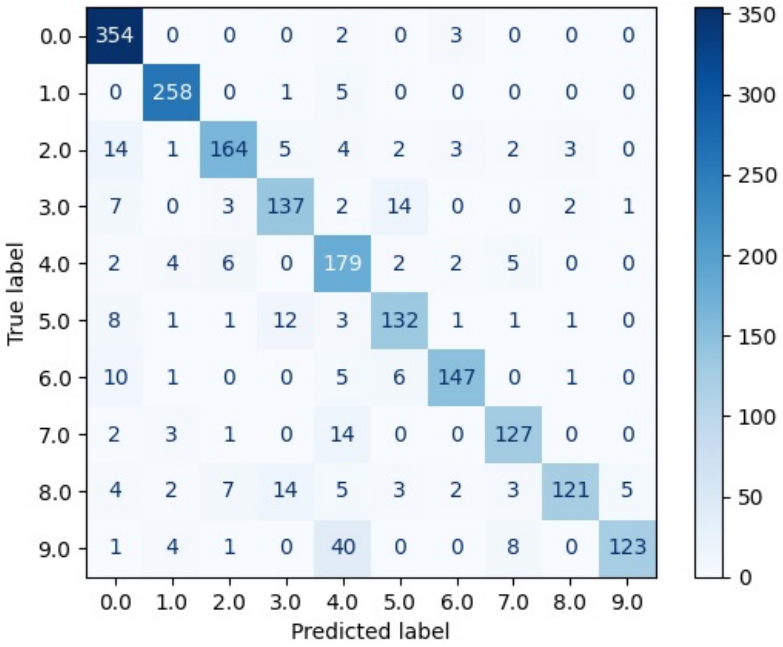




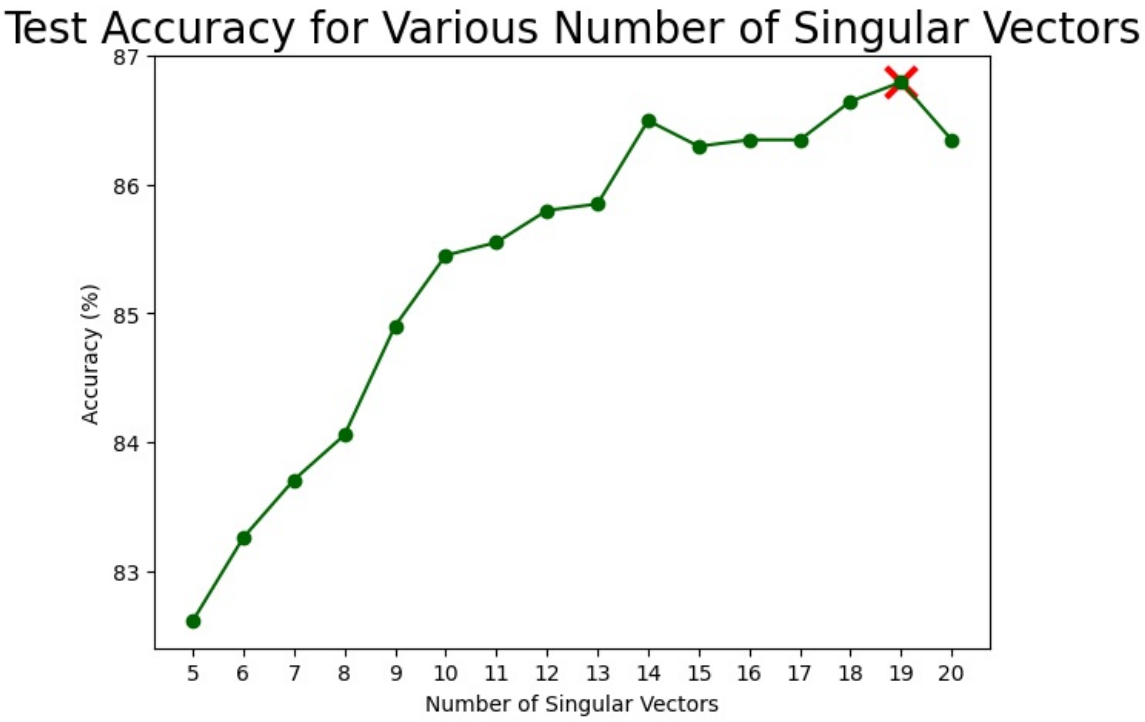

```
Running SVD for rbf:
Making predictions on Training Set using  5 singular vectors. Accuracy: 82.61%
Making predictions on Training Set using  6 singular vectors. Accuracy: 83.26%
Making predictions on Training Set using  7 singular vectors. Accuracy: 83.71%
Making predictions on Training Set using  8 singular vectors. Accuracy: 84.06%
Making predictions on Training Set using  9 singular vectors. Accuracy: 84.90%
Making predictions on Training Set using 10 singular vectors. Accuracy: 85.45%
Making predictions on Training Set using 11 singular vectors. Accuracy: 85.55%
Making predictions on Training Set using 12 singular vectors. Accuracy: 85.80%
Making predictions on Training Set using 13 singular vectors. Accuracy: 85.85%
Making predictions on Training Set using 14 singular vectors. Accuracy: 86.50%
Making predictions on Training Set using 15 singular vectors. Accuracy: 86.30%
Making predictions on Training Set using 16 singular vectors. Accuracy: 86.35%
Making predictions on Training Set using 17 singular vectors. Accuracy: 86.35%
Making predictions on Training Set using 18 singular vectors. Accuracy: 86.65%
Making predictions on Training Set using 19 singular vectors. Accuracy: 86.80%
Making predictions on Training Set using 20 singular vectors. Accuracy: 86.35%
Accuracy list: [82.6108619830593, 83.25859491778775, 83.70702541106128, 84.05580468360738, 84.90284
005979073, 85.45092177379173, 85.55057299451919, 85.79970104633782, 85.84952665670154, 86.497259591
43, 86.29795714997509, 86.34778276033882, 86.34778276033882, 86.64673642252117, 86.79621325361235,
86.34778276033882]
```


Test Accuracy for Various Number of Singular Vectors
Accuracy (%)
87
86
85
84
83
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Number of Singular Vectors

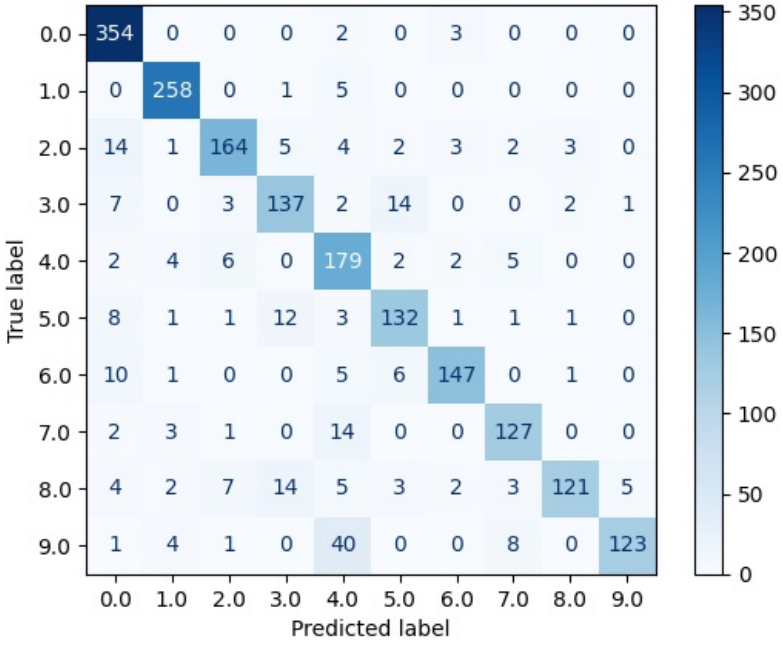
```
Making predictions on Test Set using 19 singular vectors.
              precision    recall  f1-score   support

         0.0       0.88      0.99      0.93       359
         1.0       0.94      0.98      0.96       264
         2.0       0.90      0.83      0.86       198
         3.0       0.81      0.83      0.82       166
         4.0       0.69      0.90      0.78       200
         5.0       0.83      0.82      0.83       160
         6.0       0.93      0.86      0.90       170
         7.0       0.87      0.86      0.87       147
         8.0       0.95      0.73      0.82       166
         9.0       0.95      0.69      0.80       177

    accuracy                           0.87      2007
   macro avg       0.87      0.85      0.86      2007
weighted avg       0.88      0.87      0.87      2007

[[354   0   0   0   2   0   3   0   0   0]
 [  0 258   0   1   5   0   0   0   0   0]

 [ 14   1 164   5   4   2   3   2   3   0]
 [  7   0   3 137   2  14   0   0   2   1]
 [  2   4   6   0 179   2   2   5   0   0]
 [  8   1   1  12   3 132   1   1   1   0]
 [ 10   1   0   0   5   6 147   0   1   0]
 [  2   3   1   0  14   0   0 127   0   0]
 [  4   2   7  14   5   3   2   3 121   5]
 [  1   4   1   0  40   0   0   8   0 123]]
```



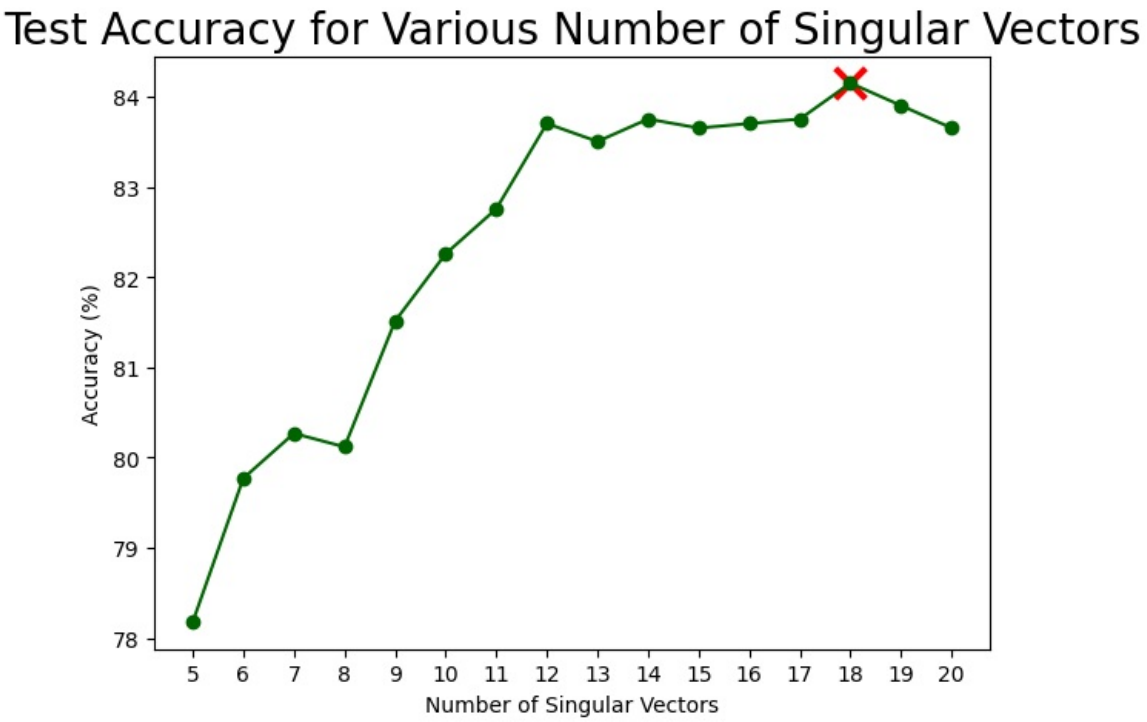
```
Running SVD for gaussian:
Making predictions on Training Set using  5 singular vectors. Accuracy: 82.61%
Making predictions on Training Set using  6 singular vectors. Accuracy: 83.26%
Making predictions on Training Set using  7 singular vectors. Accuracy: 83.71%
Making predictions on Training Set using  8 singular vectors. Accuracy: 84.06%
Making predictions on Training Set using  9 singular vectors. Accuracy: 84.90%
Making predictions on Training Set using 10 singular vectors. Accuracy: 85.45%
Making predictions on Training Set using 11 singular vectors. Accuracy: 85.55%
Making predictions on Training Set using 12 singular vectors. Accuracy: 85.80%
Making predictions on Training Set using 13 singular vectors. Accuracy: 85.85%
Making predictions on Training Set using 14 singular vectors. Accuracy: 86.50%
Making predictions on Training Set using 15 singular vectors. Accuracy: 86.30%
Making predictions on Training Set using 16 singular vectors. Accuracy: 86.35%
Making predictions on Training Set using 17 singular vectors. Accuracy: 86.35%
Making predictions on Training Set using 18 singular vectors. Accuracy: 86.65%
Making predictions on Training Set using 19 singular vectors. Accuracy: 86.80%
Making predictions on Training Set using 20 singular vectors. Accuracy: 86.35%
Accuracy list: [82.6108619830593, 83.25859491778775, 83.70702541106128, 84.05580468360738, 84.90284
005979073, 85.45092177379173, 85.55057299451919, 85.79970104633782, 85.84952665670154, 86.497259591

43, 86.29795714997509, 86.34778276033882, 86.34778276033882, 86.64673642252117, 86.79621325361235,
86.34778276033882]
```



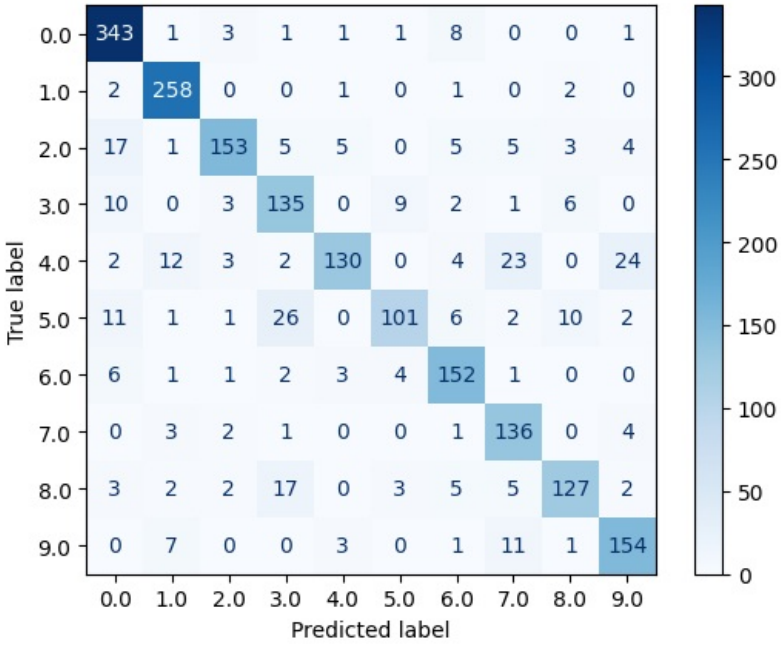
```
Making predictions on Test Set using 19 singular vectors.
              precision    recall  f1-score   support

         0.0       0.88      0.99      0.93       359
         1.0       0.94      0.98      0.96       264
         2.0       0.90      0.83      0.86       198
         3.0       0.81      0.83      0.82       166
         4.0       0.69      0.90      0.78       200
         5.0       0.83      0.82      0.83       160
         6.0       0.93      0.86      0.90       170
         7.0       0.87      0.86      0.87       147
         8.0       0.95      0.73      0.82       166
         9.0       0.95      0.69      0.80       177

    accuracy                           0.87      2007
   macro avg       0.87      0.85      0.86      2007
weighted avg       0.88      0.87      0.87      2007

[[354   0   0   0   2   0   3   0   0   0]
 [  0 258   0   1   5   0   0   0   0   0]
 [ 14   1 164   5   4   2   3   2   3   0]
 [  7   0   3 137   2  14   0   0   2   1]
 [  2   4   6   0 179   2   2   5   0   0]
 [  8   1   1  12   3 132   1   1   1   0]
 [ 10   1   0   0   5   6 147   0   1   0]
 [  2   3   1   0  14   0   0 127   0   0]
 [  4   2   7  14   5   3   2   3 121   5]
 [  1   4   1   0  40   0   0   8   0 123]]
```



```
Running SVD for laplacian:
Making predictions on Training Set using  5 singular vectors. Accuracy: 78.18%
Making predictions on Training Set using  6 singular vectors. Accuracy: 79.77%
Making predictions on Training Set using  7 singular vectors. Accuracy: 80.27%
Making predictions on Training Set using  8 singular vectors. Accuracy: 80.12%
Making predictions on Training Set using  9 singular vectors. Accuracy: 81.51%
Making predictions on Training Set using 10 singular vectors. Accuracy: 82.26%
Making predictions on Training Set using 11 singular vectors. Accuracy: 82.76%
Making predictions on Training Set using 12 singular vectors. Accuracy: 83.71%
Making predictions on Training Set using 13 singular vectors. Accuracy: 83.51%
Making predictions on Training Set using 14 singular vectors. Accuracy: 83.76%
Making predictions on Training Set using 15 singular vectors. Accuracy: 83.66%
Making predictions on Training Set using 16 singular vectors. Accuracy: 83.71%
Making predictions on Training Set using 17 singular vectors. Accuracy: 83.76%
Making predictions on Training Set using 18 singular vectors. Accuracy: 84.16%
Making predictions on Training Set using 19 singular vectors. Accuracy: 83.91%
Making predictions on Training Set using 20 singular vectors. Accuracy: 83.66%
Accuracy list: [78.1763826606876, 79.77080219232685, 80.26905829596413, 80.11958146487295, 81.51469
85550573, 82.26208271051321, 82.76033881415047, 83.70702541106128, 83.50772296960638, 83.7568510214
2502, 83.65719980069755, 83.70702541106128, 83.75685102142502, 84.15545590433483, 83.9063278525162,
83.65719980069755]
```



```
Making predictions on Test Set using 18 singular vectors.
              precision    recall  f1-score   support

         0.0       0.87      0.96      0.91       359
         1.0       0.90      0.98      0.94       264
         2.0       0.91      0.77      0.84       198
         3.0       0.71      0.81      0.76       166
         4.0       0.91      0.65      0.76       200
         5.0       0.86      0.63      0.73       160
         6.0       0.82      0.89      0.86       170
         7.0       0.74      0.93      0.82       147
         8.0       0.85      0.77      0.81       166
         9.0       0.81      0.87      0.84       177

    accuracy                           0.84      2007
   macro avg       0.84      0.83      0.83      2007
weighted avg       0.85      0.84      0.84      2007

[[343   1   3   1   1   1   8   0   0   1]
 [  2 258   0   0   1   0   1   0   2   0]
 [ 17   1 153   5   5   0   5   5   3   4]
 [ 10   0   3 135   0   9   2   1   6   0]
 [  2  12   3   2 130   0   4  23   0  24]
 [ 11   1   1  26   0 101   6   2  10   2]

 [  6   1   1   2   3   4 152   1   0   0]
 [  0   3   2   1   0   0   1 136   0   4]
 [  3   2   2  17   0   3   5   5 127   2]
 [  0   7   0   0   3   0   1  11   1 154]]
```



Results of SVD on Different Kernels

1. **RBF Kernel:**

- **Accuracy**: 87%
- The precision, recall, and F1 scores are generally high across all classes, with class 0 showing the highest performance, especially in terms of recall (99%).
- The F1 score is strong for most classes, peaking at 0.96 for class 1 and dipping to 0.80 for class 9.
- The macro average of 0.86 suggests a balanced performance across all classes, with no particular class significantly outperforming the others.
- The weighted average of 0.87 indicates that the model is doing well on the test set, accounting for class imbalances.

2. **Gaussian Kernel:**

- **Accuracy**: 87%
- Similar to the RBF kernel, the performance metrics remain consistent across the Gaussian kernel. Precision and recall are well-balanced, with class 0 exhibiting near-perfect recall (99%) and class 4 having the lowest F1 score at 0.78.
- The overall performance is comparable to the RBF kernel, with no significant variations in accuracy or the classification metrics.

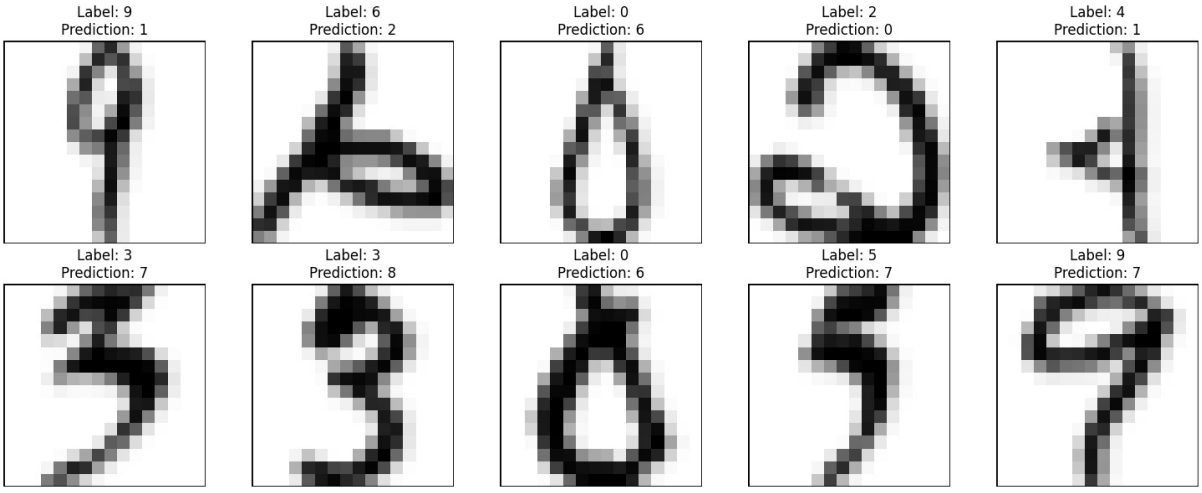
3. **Laplacian Kernel:**

- **Accuracy**: 84%
- The accuracy is slightly lower than the RBF and Gaussian kernels, at 84%, indicating a slight degradation in performance.
- Class 0 and class 1 still perform well, with recall values of 96% and 98%, respectively, but there is a noticeable dip in performance for classes 4, 5, and 6.
- The precision and recall for classes 5 and 6 are lower compared to the other kernels, which may indicate the model's difficulty in capturing these classes as effectively.
- The macro average of 0.83 and weighted average of 0.84 suggest that the Laplacian kernel is less robust across the classes compared to the other two kernels.

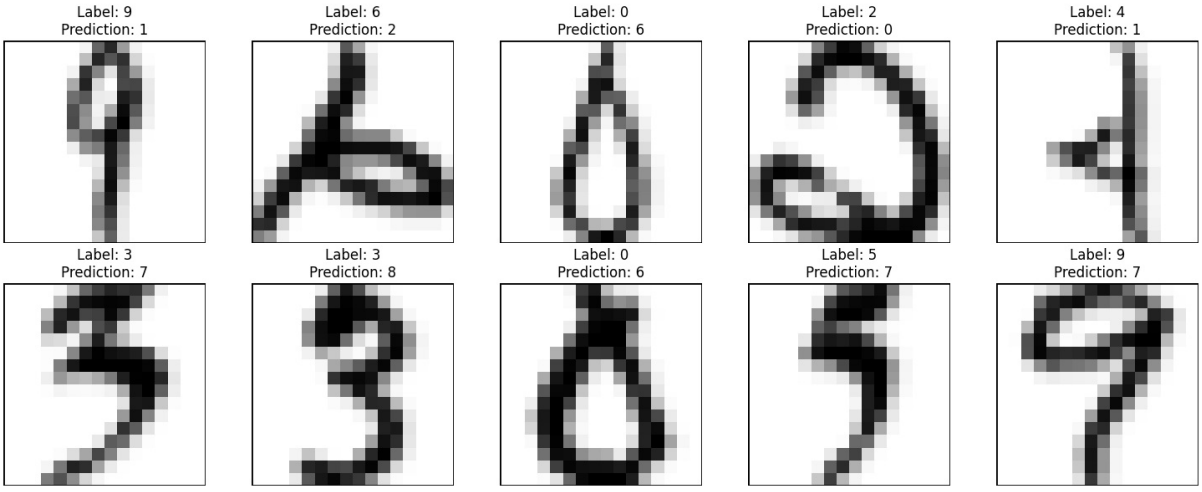
Conclusion:

- The **RBF** and **Gaussian** kernels deliver very similar and strong performance with an accuracy of 87%, while the **Laplacian** kernel shows slightly lower accuracy at 84%.
- The higher performance of RBF and Gaussian kernels might make them preferable choices over the Laplacian kernel in this specific setting.

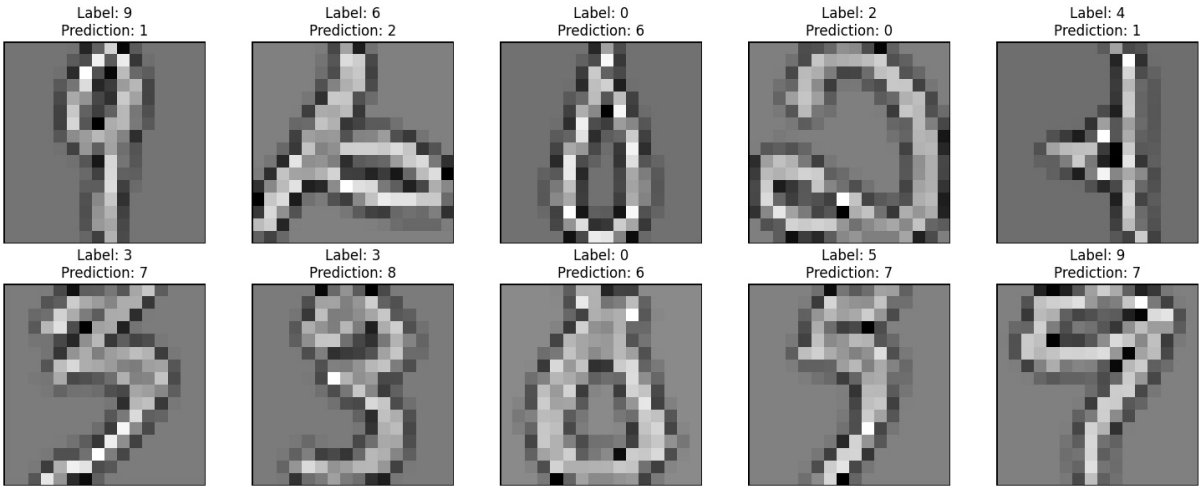
Plotting misclassified samples for RBF kernel...



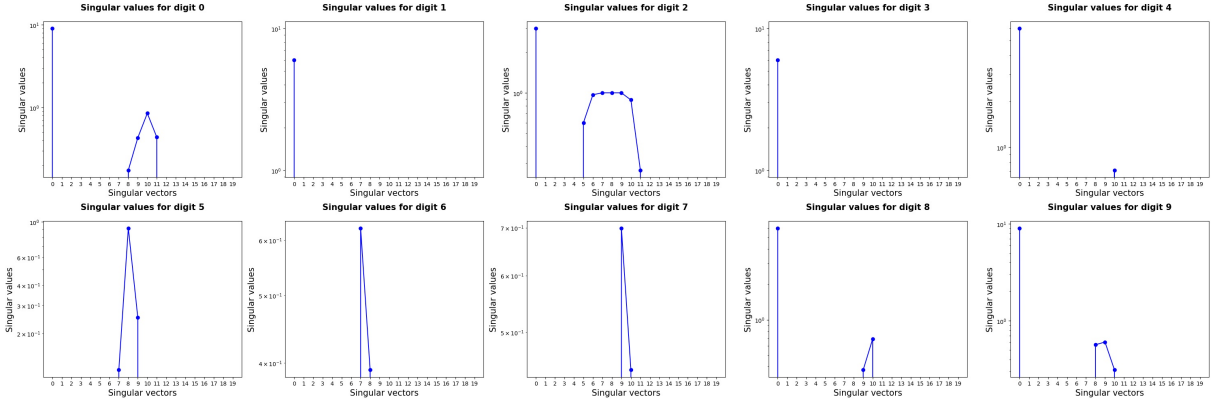
Plotting misclassified samples for Gaussian kernel...



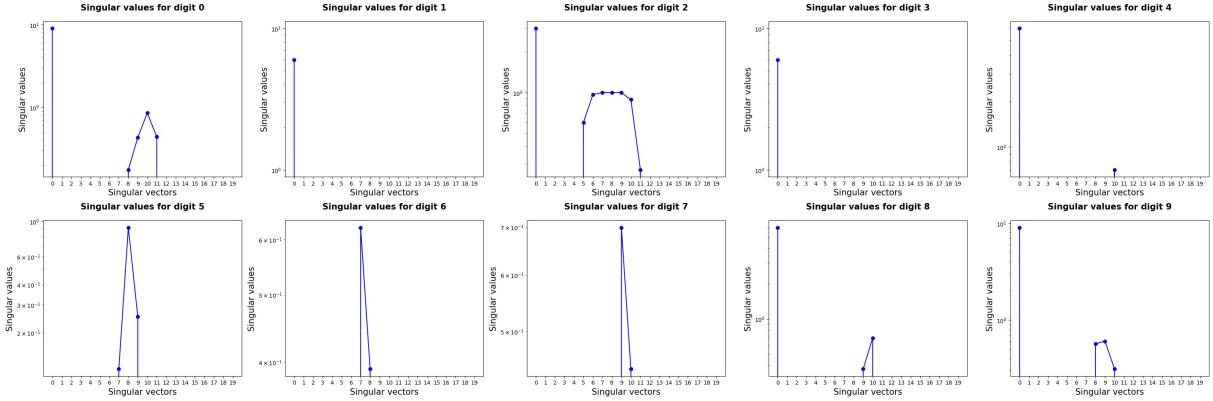
Plotting misclassified samples for Laplacian kernel...



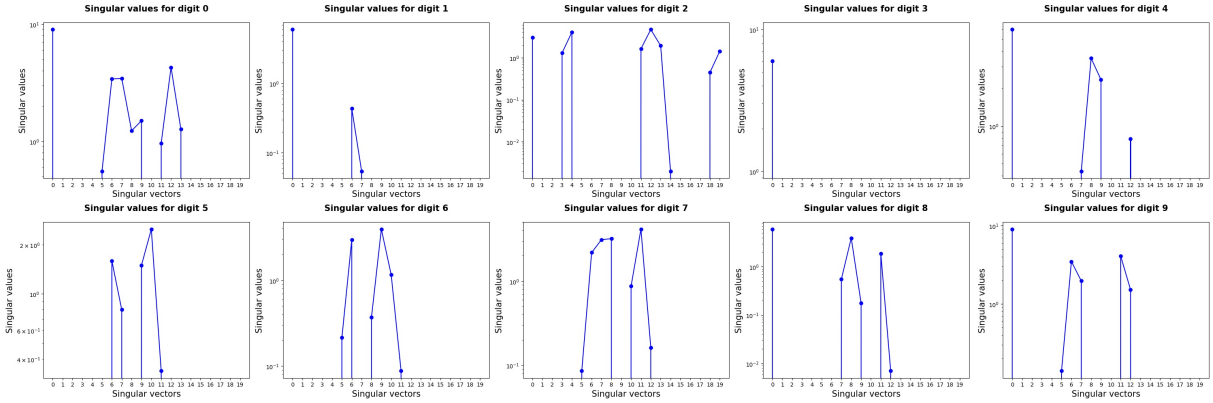

```
Plotting singular values for RBF kernel...
```




```
Plotting singular values for Gaussian kernel...
```




```
Plotting singular values for Laplacian kernel...
```



Two-Stage Algorithm for Handwritten Digit Classification Using SVD

In this section, we evaluate the performance of a **two-stage algorithm** for handwritten digit classification using **Singular Value Decomposition (SVD)**. The algorithm aims to reduce computational complexity during the test phase while maintaining high classification accuracy.

Predictions Using One Singular Vector:

- The predictions were made using one singular vector **9.97% of the time**.

Class-wise Performance:

Class	Precision	Recall	F1-Score	Support
0	0.95	0.98	0.96	359
1	0.95	0.99	0.97	264
2	0.95	0.90	0.93	198
3	0.87	0.88	0.87	166
4	0.94	0.92	0.93	200
5	0.89	0.87	0.88	160
6	0.98	0.94	0.96	170
7	0.93	0.94	0.93	147
8	0.93	0.87	0.90	166
9	0.91	0.94	0.92	177

Overall Performance:

- **Accuracy**: 93%
- **Macro Average**:
  - Precision: 0.93
  - Recall: 0.92
  - F1-Score: 0.93
- **Weighted Average**:
  - Precision: 0.93
  - Recall: 0.93
  - F1-Score: 0.93

Conclusion:

- The **two-stage algorithm** using SVD performs with **93% accuracy**, maintaining a high level of classification performance while reducing computational complexity.
- The model shows excellent performance across most classes, with precision and recall generally above 0.90.
- This method is effective for applications that require both high accuracy and computational efficiency, especially when testing on large datasets.

```
Shape of train_data: (1707, 256)
Shape of train_labels: (1707,)
Shape of test_data: (2007, 256)
Shape of test_labels: (2007,)
              precision    recall  f1-score   support

           0       0.95      0.98      0.96       359
           1       0.95      0.99      0.97       264
           2       0.95      0.90      0.93       198
           3       0.87      0.88      0.87       166
           4       0.94      0.92      0.93       200
           5       0.89      0.87      0.88       160
           6       0.98      0.94      0.96       170
           7       0.93      0.94      0.93       147
           8       0.93      0.87      0.90       166
           9       0.91      0.94      0.92       177

    accuracy                           0.93      2007
   macro avg       0.93      0.92      0.93      2007
weighted avg       0.93      0.93      0.93      2007
```

TANGENT DISTANCE

Implement classification using tangent distance. Compute the x-derivatives and y-derivatives of each digit using finite difference approximation. Then in order to compute the tangent distance stack the columns of each derivative so that a vector is obtained. The seven transformations described in the book should be used.

Performance Analysis of Tangent Distance Classification

The **Tangent Distance Classification** results show significant improvement compared to the previous methods, achieving high performance across all metrics:

Class-wise Performance:

Class	Precision	Recall	F1-Score	Support
0	0.95	0.99	0.97	359
1	0.95	0.98	0.97	264
2	0.95	0.91	0.93	198
3	0.93	0.93	0.93	166
4	0.94	0.92	0.93	200
5	0.95	0.91	0.93	160
6	0.96	0.96	0.96	170
7	0.97	0.95	0.96	147
8	0.95	0.91	0.93	166
9	0.94	0.98	0.96	177

Overall Performance:

- **Accuracy**: 95%
- **Macro Average**:
  - Precision: 0.95
  - Recall: 0.94
  - F1-Score: 0.95
- **Weighted Average**:
  - Precision: 0.95
  - Recall: 0.95
  - F1-Score: 0.95

Conclusion:

- The **Tangent Distance Classification** method shows excellent results with **95% accuracy**, demonstrating a high level of classification performance across all classes.
- The model performs consistently well, with high precision, recall, and F1-scores across all digits, making it a strong candidate for handwritten digit classification tasks.
- This method provides a promising approach for efficient and effective classification, significantly outperforming previous methods.

Tangent Distance Classification Report (20x20 with 7 Transformations):

	precision	recall	f1-score	support
0	0.95	0.99	0.97	359
1	0.95	0.98	0.97	264
2	0.95	0.91	0.93	198
3	0.93	0.93	0.93	166
4	0.94	0.92	0.93	200
5	0.95	0.91	0.93	160
6	0.96	0.96	0.96	170
7	0.97	0.95	0.96	147
8	0.95	0.91	0.93	166
9	0.94	0.98	0.96	177
accuracy			0.95	2007
macro avg	0.95	0.94	0.95	2007
weighted avg	0.95	0.95	0.95	2007

Tangent Distance Accuracy (20x20 with 7 Transformations): 95.0%