MNIST Classification

Using KNN, Naïve Bayes, Linear Logistic Regression, MLP, Clustering, Fast Retrieval, PCA and Data Augmentation

by **Rajat Chakraborty**

1. KNN, Naïve Bayes and Linear Regression

a. Validation Error:

	KNN	Naive Bayes	Logistic Regression
Val Error (%)	3.04	16.32	7.53
Training Time (s)	0	7.85	35.56
Inference Time (ms)	166.4	7.96e-2	1.94

b. Most common mistaken label:

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Most common mistaken label	6	7	7	5	9	3	0,1	9	1,3	7
% of times assigned to that label	0.3	0.2	1.2	2.1	2.9	1.4	0.4	0.9	1.2	1

c. Plot Validation Error (%) vs. Training Size

# training samples	KNN	Naive Bayes	Logistic Regression
50	33.72	40.95	35.03
500	16.11	23.28	15.34
5,000	6.82	16.94	10.85
50,000	3.04	16.32	7.53

d. Parameter selection

	KNN (K)	Naive Bayes (alpha)	Logistic Regression (C)
Best parameter	1	0.0009	0.7
Validation error (%)	16.11	20.36	15.31
Test error (%)	16.22	19.23	14.81

e. Improvement on kNN principle common analysis (PCA) and augmentation technique

Using KNN as it provided the best performance in before with following modifications:

- Using principle common analysis (PCA), selected the most important features by trial and error. In my case, the number was 35.
- Selected the optimum K(=4).
- Used augmentation technique from internet to increase the number of training sample. This technique basically shifts the pixel to create the same number.

Validation Error: 1.66
Test Error: 1.92

2. MLPs with MNIST

a. On original data

Training and validation losses after the given number of epochs

# training epochs	Training Loss	Validation Loss
25	0.10863	0.1408
50	0.05996	0.1078
100	0.02638	0.10198

Best model parameters:

- #Hidden Layers = 3
- Activation = ReLU
- Hidden layer size = 128 each
- #epoch = 128
- Learning rate =0.0085
- Optimizer = Adam

Best model losses and errors:

Training **Loss**	Validation **Loss**
0.0115	0.3393

Training **Error (%)**	Validation **Error (%)**	Test **Error (%)**
0.168	2.17	2.08

b. MLP on data after PCA and Data Augmentation

I used the following model after applying PCA and data augmentation to the original data:

- #Hidden Layers = 3
- Activation = ReLU
- Hidden layer size = 128 each
- #epoch = 128
- Learning rate =0.0085

- Optimizer = Adam
- Batch size for training data =512, batch size for validation and test data = 128

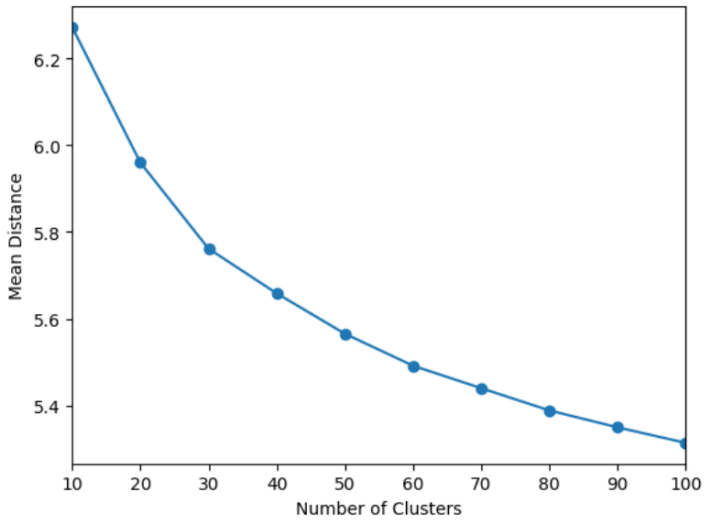
Errors:

Training **Error (%)**	Validation **Error (%)**	Test **Error (%)**
0.3372	1.61	1.58

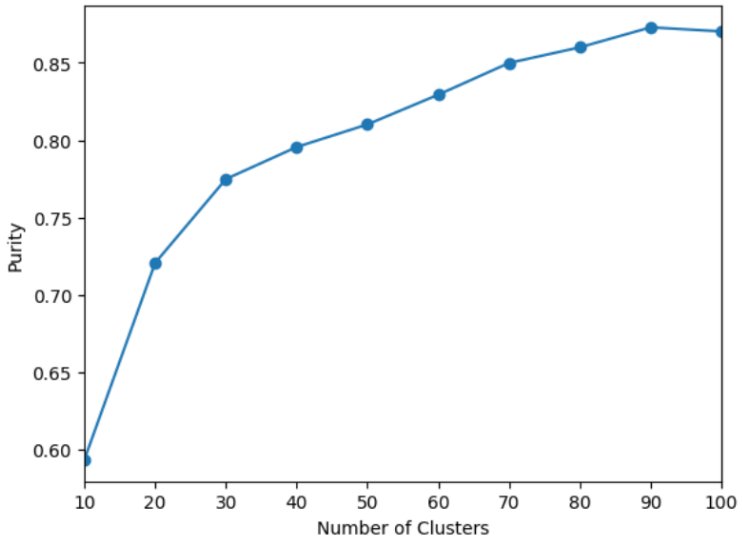
3. Clustering and FAISS Retrieval

a. Test K-Means Purity

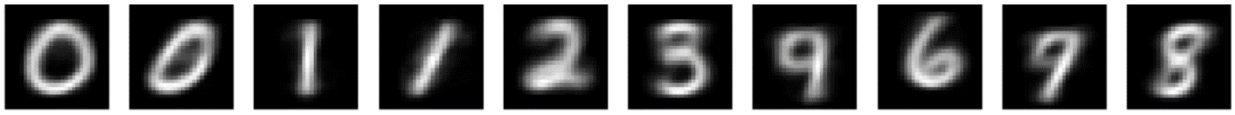
Distance:



Purity:



Immages of centroids for K = 10, K = 20, and K=30 below (three rows).
K=10:



K=20:

K=30:

Observations:

- **Purity and K:** Purity increases with more clusters (K) as it captures subtle differences, but may plateau or decrease due to overfitting.
- **Average Distance:** The average distance to the centroid decreases with iterations but stabilizes. Increasing K reduces this distance as points get closer to centroids.
- **Optimal Clustering:** K-means may not find the optimal clustering due to non-convexity and dependence on initial centroids. Multiple runs with different initializations help mitigate this.
- **Purity vs. Objective:** Lower mean squared error in K-means doesn't guarantee higher purity, as purity depends on data homogeneity, not just distance to centroids.

b. Fast Retrievals

i. Brute Force:

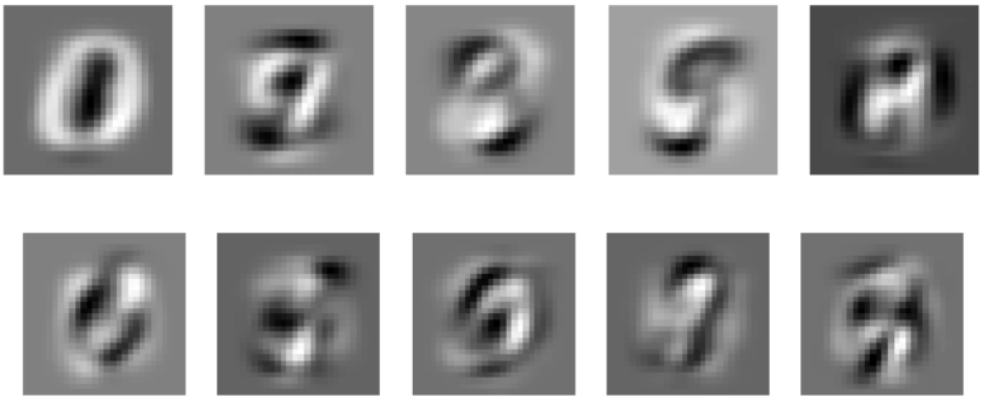
Test Error (%)	Time to Add (s)	Time to Search (s)
3.31	0.207	16.524

ii. LSH:

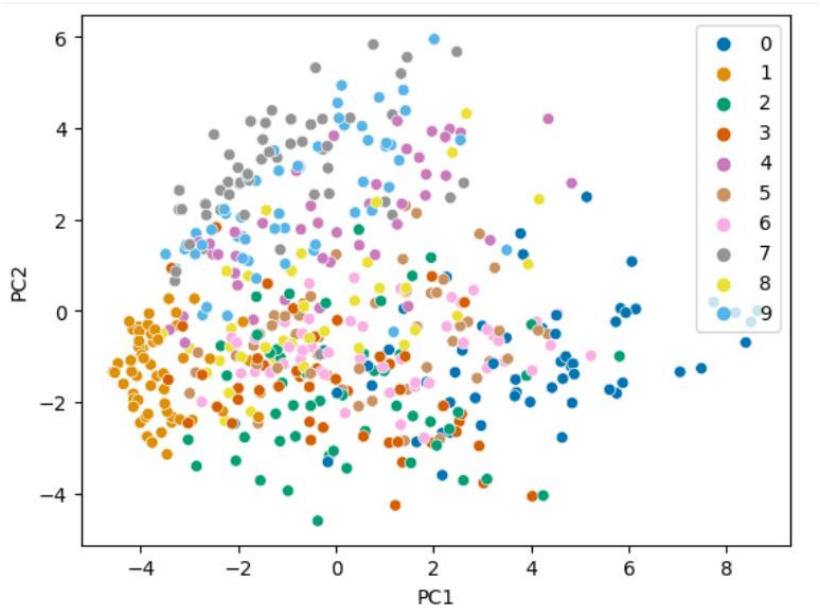
Test Error (%)	Time to Add (s)	Time to Search (s)	Nbits parameter
3.18	3.52	16.99	1568 (2*dim)

4. PCA and Data Compression

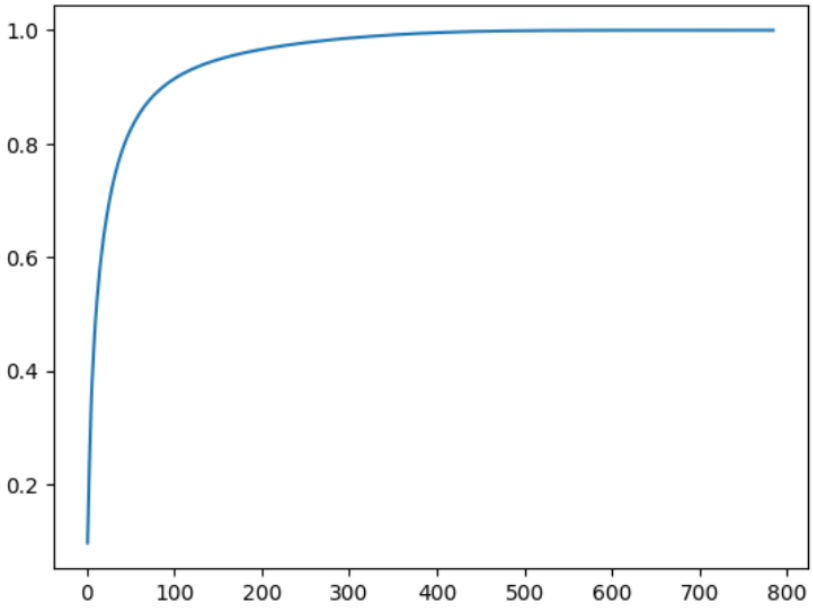
First 10 principal components:



Scatterplot:



Cumulative explained Variance:



Faiss:

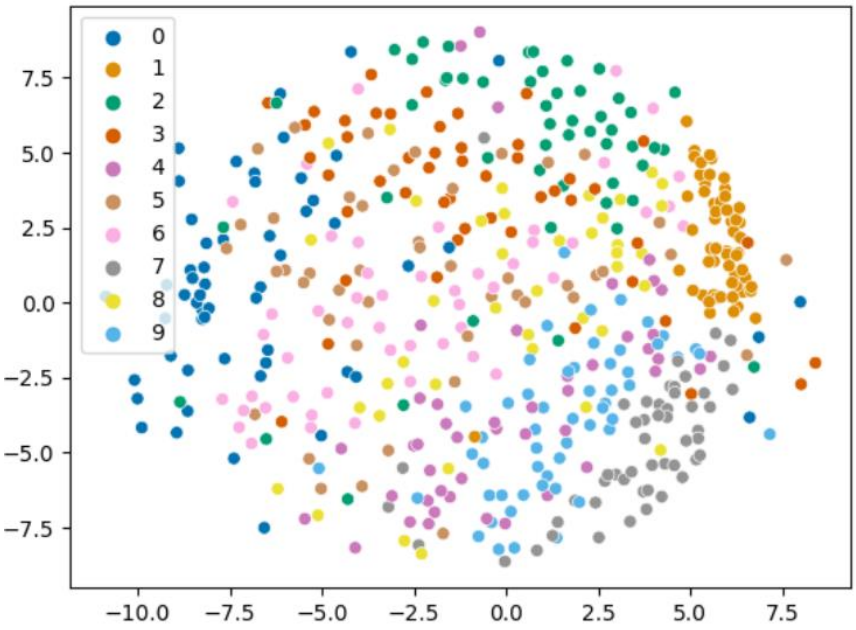
	Total Time (s)	Test Error (%)	Dimensions
Brute Force (PCA)	2.77433	2.91	88
Brute Force	16.70556	3.31	784
LSH	22.796769	3.18	784

The time has been significantly reduced. The accuracy of the method has also improved.

MDS and t-SNE:

For MDS and t-SNE, I have performed 30 components PCA transform before applying the mentioned methods.

MDS:



t-SNE: