Assignment 2

Machine Learning

Submitted By

Ratnesh Kumar Tiwari
M23MAC011

To

Dr. Deepak Mishra

Task 1:
Please visit the code:
https://colab.research.google.com/drive/1wRKRi3rr-iLAE9nM3T5OtM5AtLHku1W1?usp=sharing

a) Perform K-means clustering on MNIST data from scratch. Instead of using Euclidian distance as distance metric use Cosine Similarity as distance metric. Clustering should be done in 10, 7, and 4 clusters.

K Means Clustering:
K-means clustering is a popular unsupervised machine learning algorithm for grouping data into clusters. It works by selecting initial cluster centers, assigning data points to the nearest cluster, updating the cluster centers, and repeating until convergence. K-means is efficient and versatile but sensitive to initial conditions and assumes spherical clusters. It's a useful tool for data segmentation and exploration.

In the implementation, I had firstly separated the given MNIST dataset into two part, the first attribute as y, i.e. label, and the rest all as X i.e. feature.
In this assignment, we only needed the X portion because of supervised learning.

After that, I have normalized the whole dataset by dividing it with 255 (as 255 is the highest value any of the data points can take). This will ensure that our data should fall in range 0 to 1 for faster computation.

The K Means algorithm suggests us to initialize the centroid as a random point. But for faster conversion, I took the K random data points from X and initialized them as K centroids. This will ensure that the initial point is not abruptly away from all the data points.

The maximum iteration for checking the convergence of centroid is set to 1000. It is a random guess and can be different for different datasets. It is believed that centroid will converge within this many iterations.

The distance is the important factor in the K -means clustering algorithm. This is the deciding factor to assign the datapoint in a particular cluster. There are several way to calculate the distance. Some common are euclidean, cosine, manhattan distance etc. In this assignment, I have used cosine distance as mentioned specifically.

Cosine Distance:

$$d_{cos}(x,y) = 1 - \frac{\sum_{i=0}^{n-1} x_i y_i}{\sqrt{\sum_{i=0}^{n-1} x_i^2} \times \sqrt{\sum_{i=0}^{n-1} y_i^2}}$$

Algorithm of K- Means:

Input:

$D = \{d_1, d_2, ..., d_N\}$	//set of N data items.
K	// Number of desired cluster

Output:

A set of K clusters.

Step:

```
1.  Arbitrarily choose & data-items from D as initial centroids;
2.  Repeat
        Assign each item $d_i$ to the cluster which has the closest centroid;
        Calculate new mean for each cluster;
    Until convergence criteria is met.
```

Time Complexity of K Means Clustering Algorithm:

The time complexity of the algorithm is a NP Hard problem because the number of iteration is not fixed for every centroid to converge. However if we consider a fixed upper bound in number of iterations (in our case 1000), the time complexity comes out to be **O(T*K*N*D).**

Here,

T is maximum number of iterations

K is the number of clusters

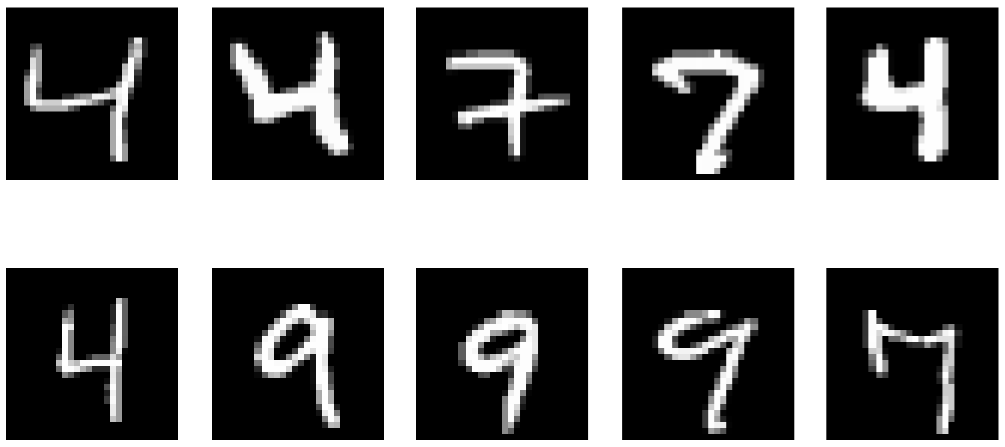
N is the number of datapoints

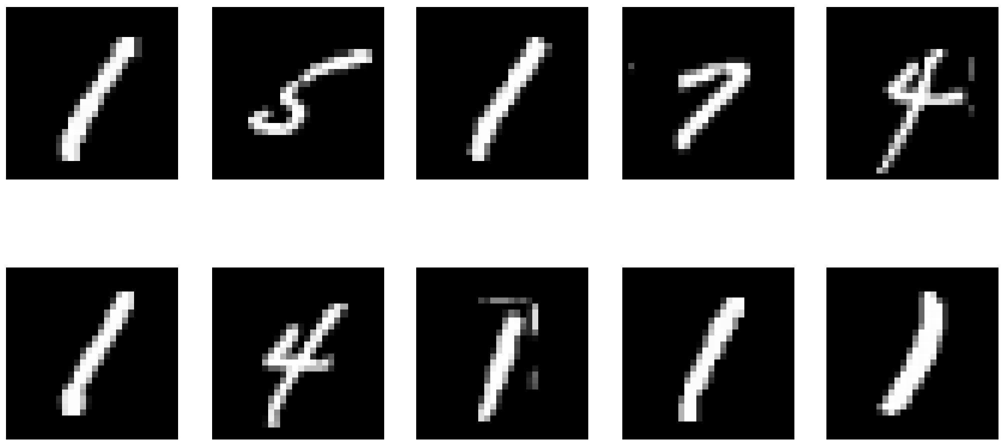
D is the number of attributes in the dataset.

b.) Visualize the images getting clustered in different clusters.

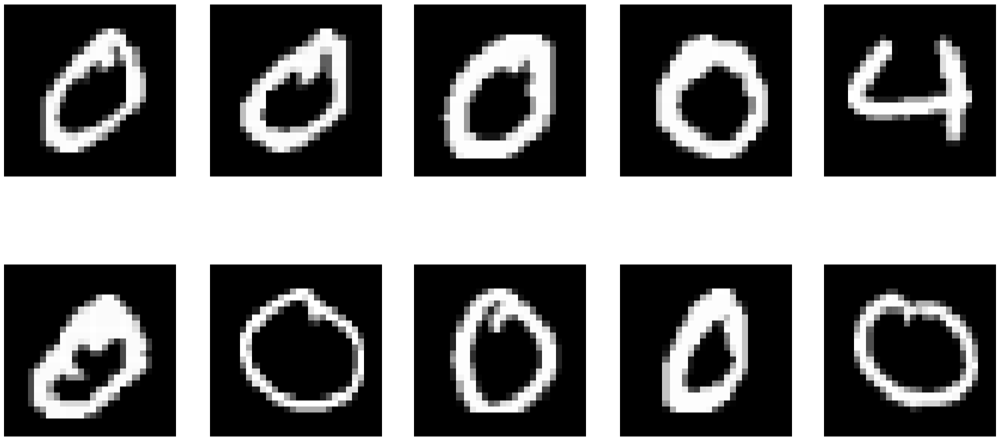
The output is the first 10 images from each cluster.

1. **When the total number of clusters is 10:**

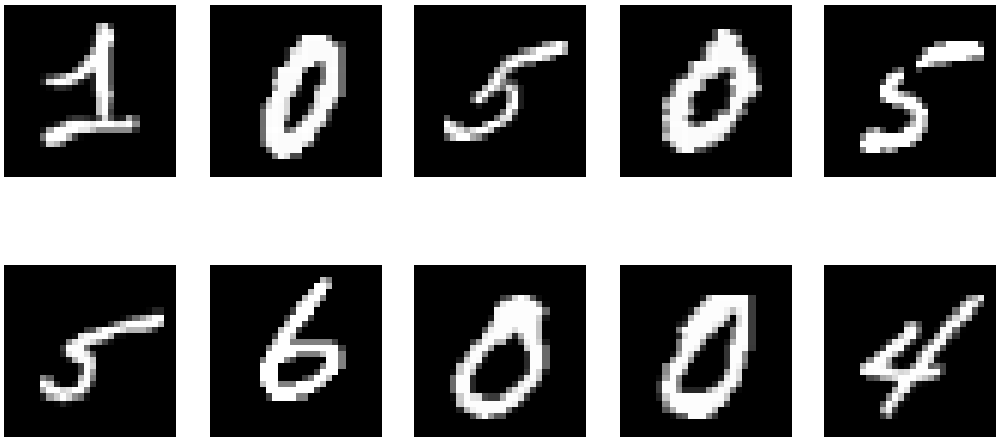

Samples of Cluster 1


Samples of Cluster 2

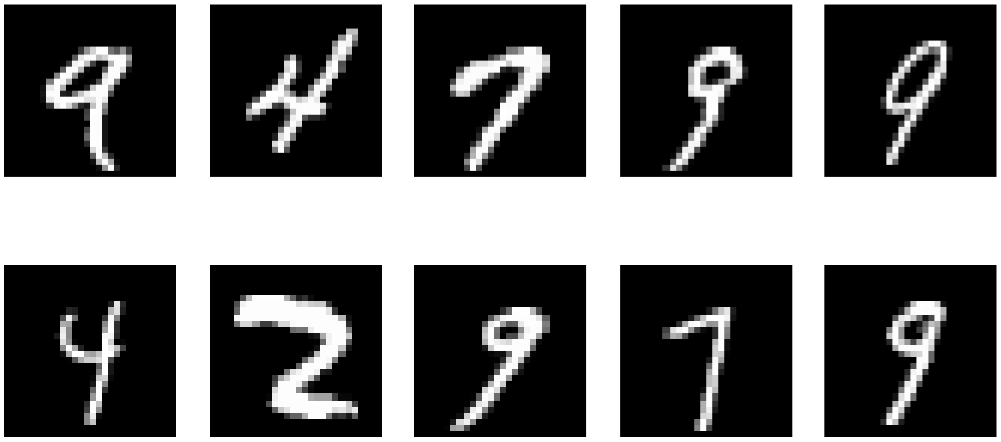
Samples of Cluster 3



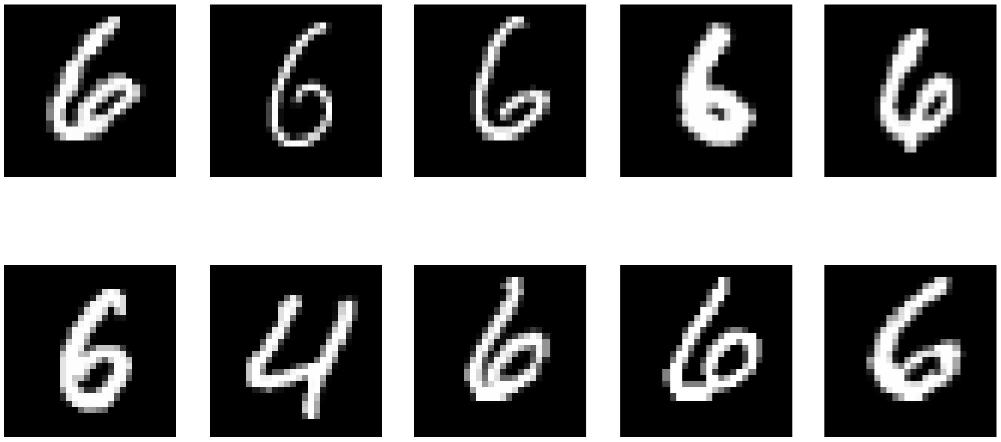
Samples of Cluster 4



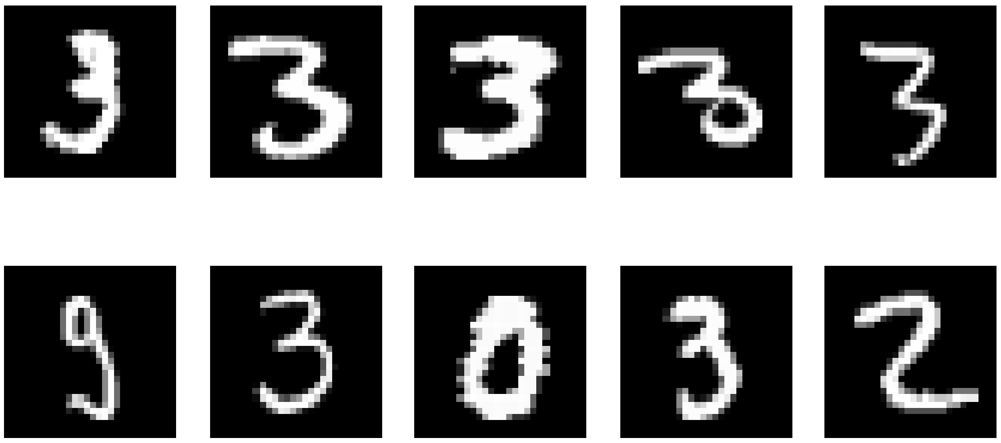
Samples of Cluster 5



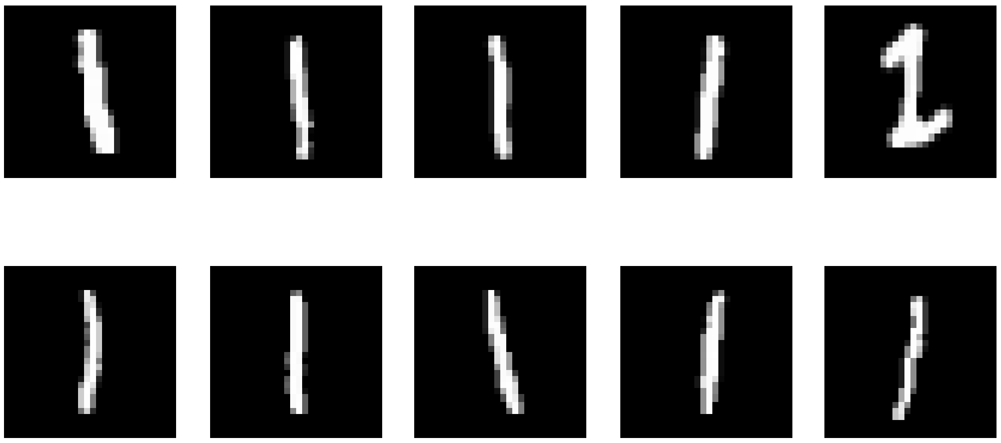
Samples of Cluster 6



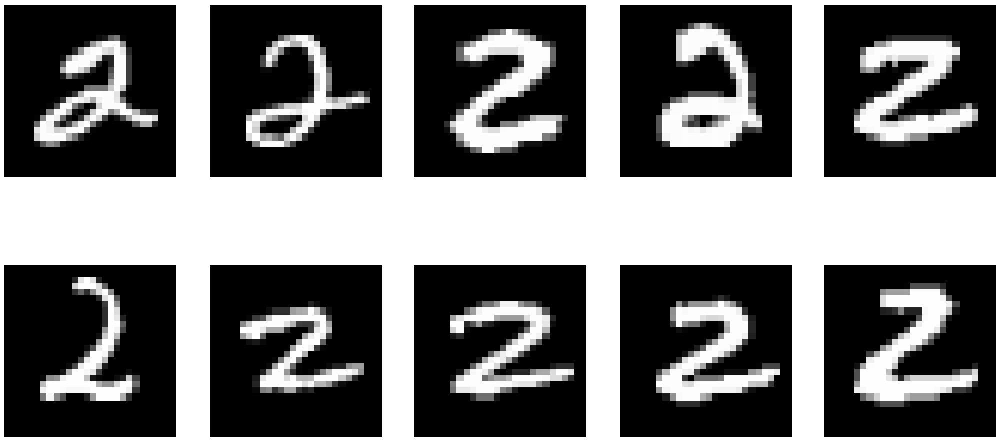
Samples of Cluster 7



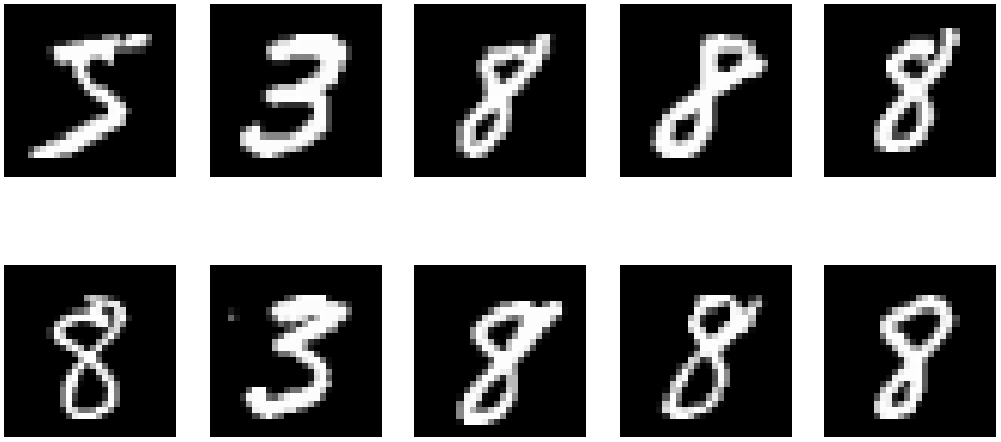
Samples of Cluster 8



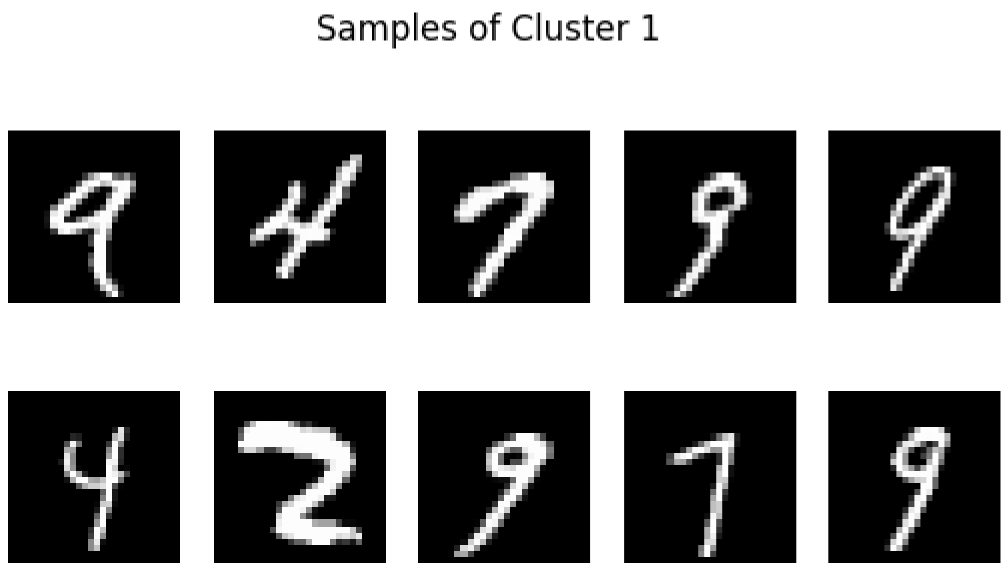
Samples of Cluster 9

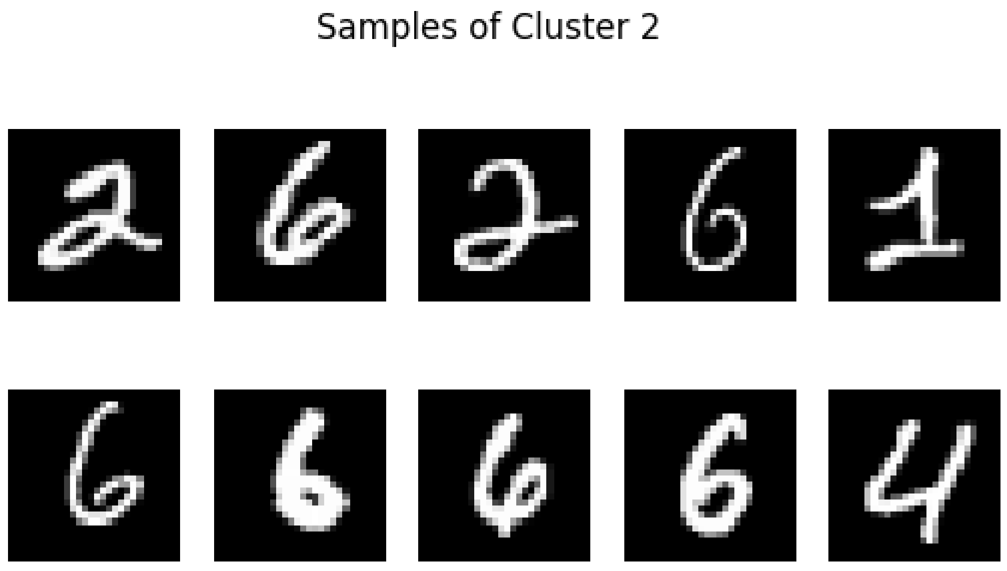


Samples of Cluster 10

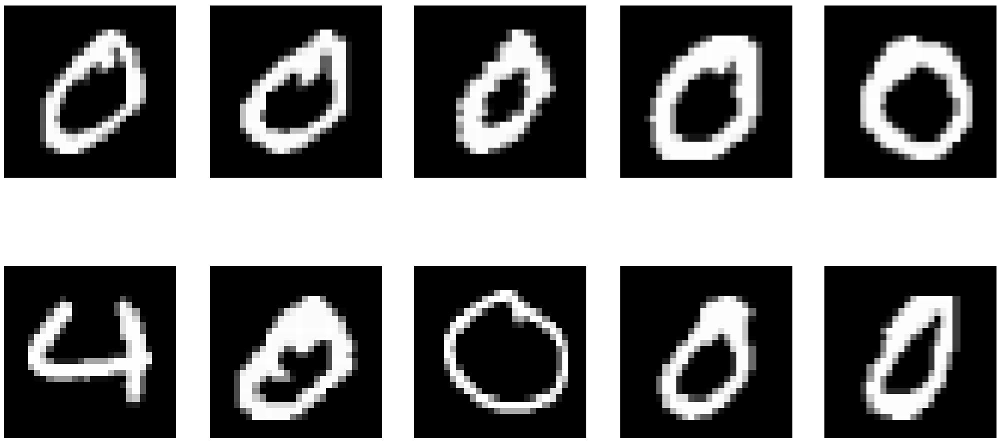


2. **When total number of cluster is 7:**



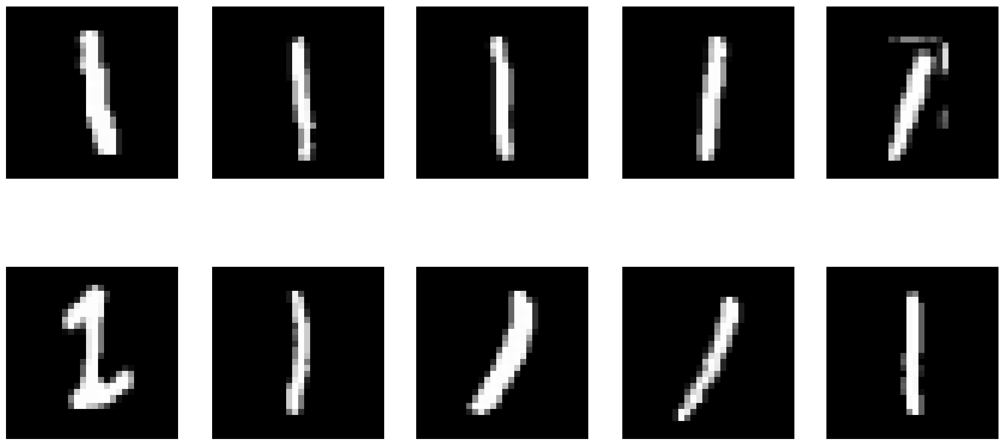


Samples of Cluster 3

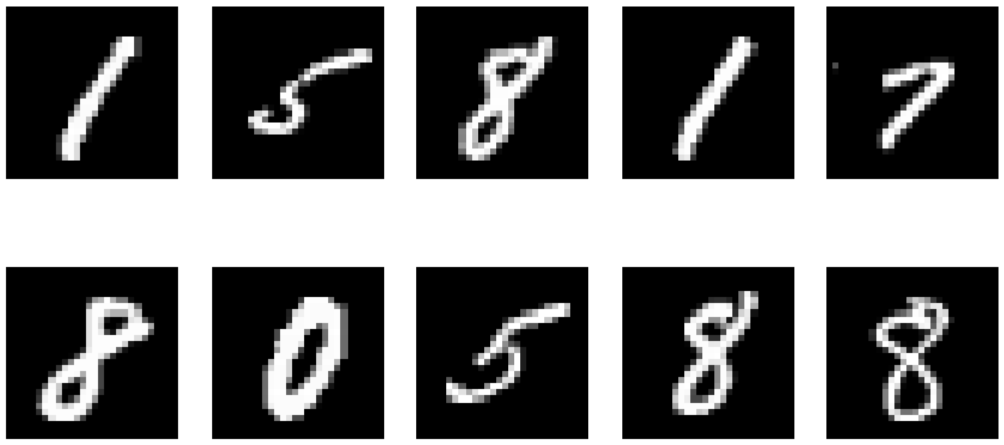


Samples of Cluster 4

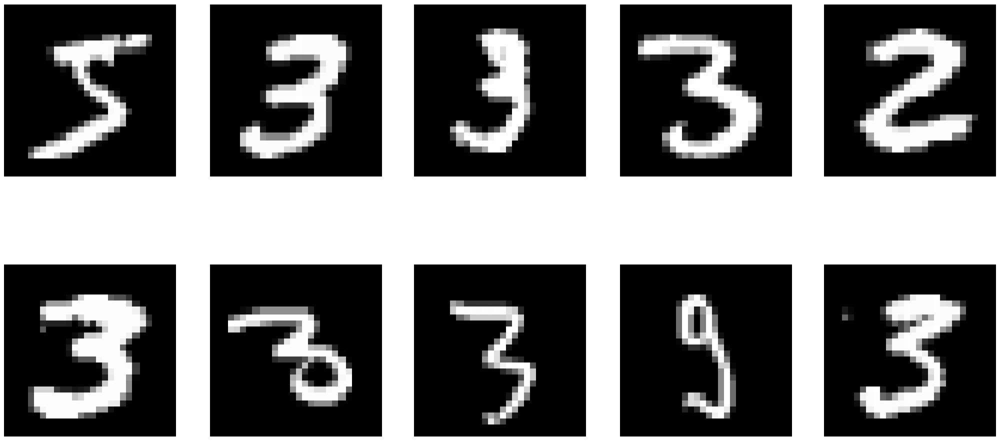
Samples of Cluster 5



Samples of Cluster 6

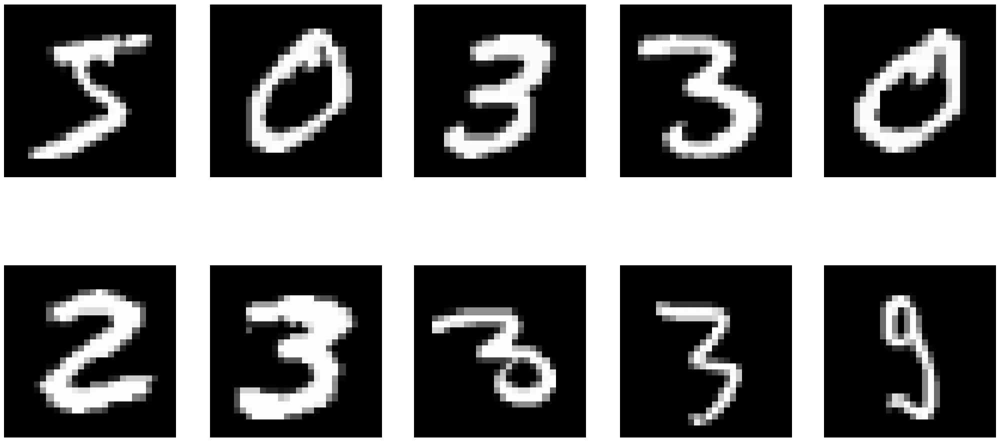


Samples of Cluster 7

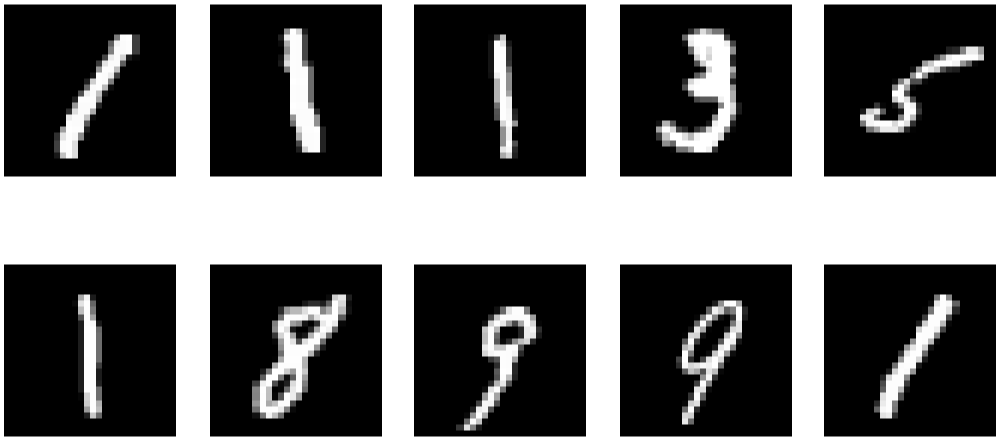


3. **When total number of cluster is 4:**

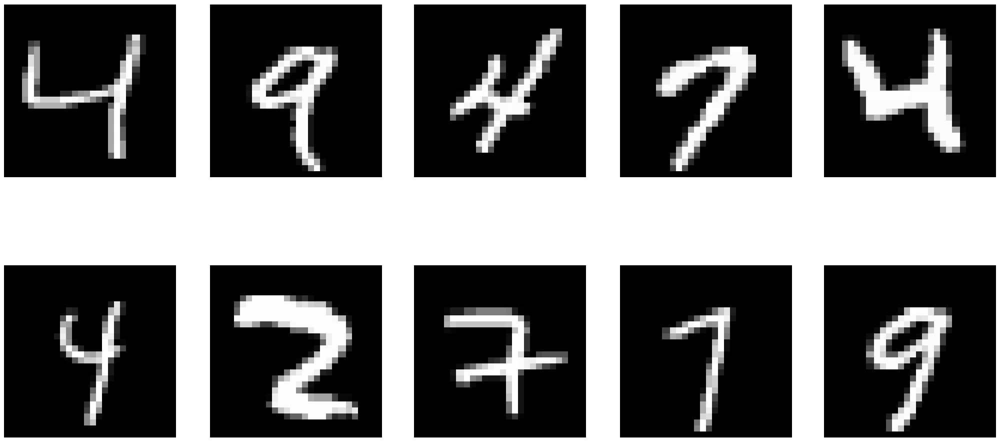
Samples of Cluster 1

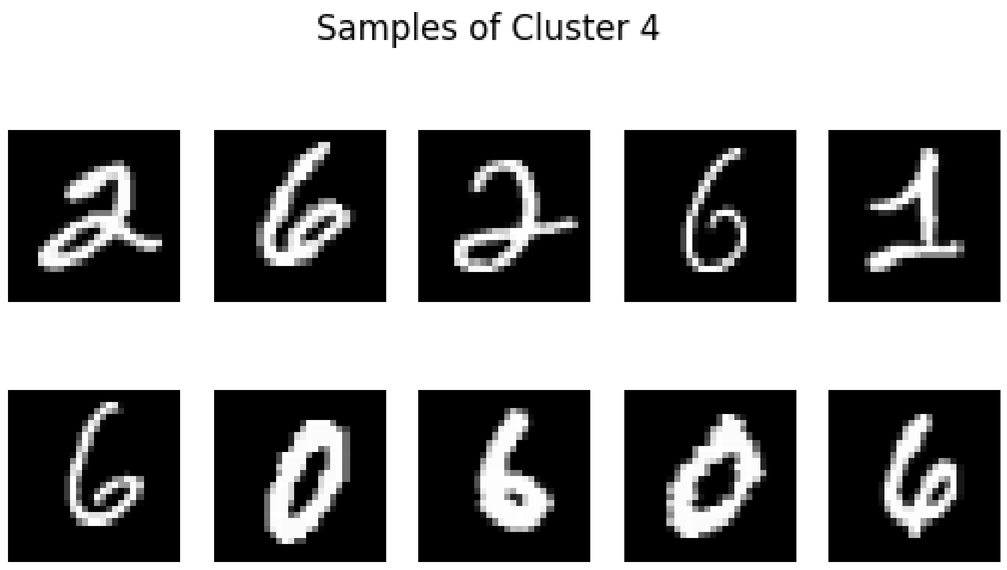


Samples of Cluster 2



Samples of Cluster 3





c.) Please comment on the cluster characteristics.
The characteristics of the clusters may differ depending on the number of clusters chosen.

K = 10:
With K set to 10, the algorithm is attempting to cluster the data into 10 different clusters. In the context of the MNIST dataset, this likely corresponds to grouping handwritten digits (0 to 9) into 10 clusters.
Each cluster would ideally represent one of the digits (0-9), and the characteristics of these clusters can be assessed by examining the average appearance of each digit within its assigned cluster.

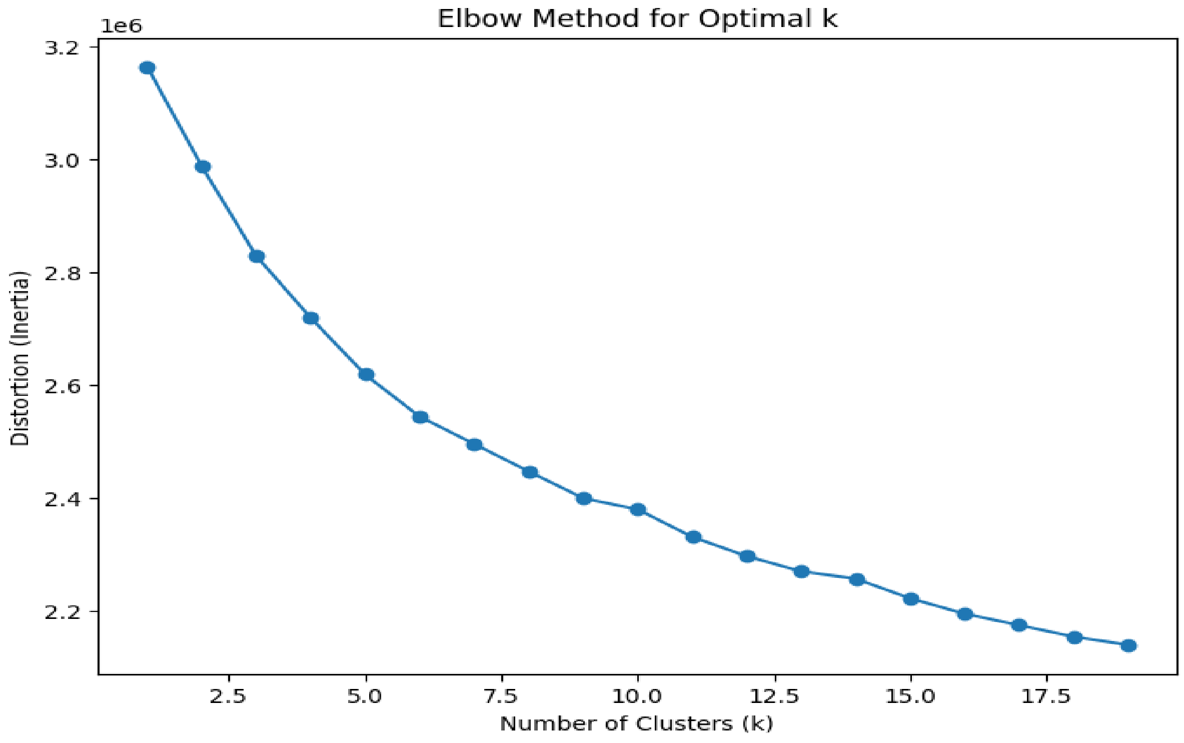
K = 7:
With K set to 7, the algorithm is aiming to find 7 different clusters in the MNIST dataset. The cluster characteristics may not directly correspond to the individual digits (0-9) but are based on the natural patterns and groupings present in the data.

K = 4:
When K is set to 4, the algorithm is looking for 4 clusters within the MNIST dataset. The cluster characteristics in this case will depend on the patterns and similarities among the data points. The characteristics may be more abstract and may not directly correspond to digits as in the case of K = 10.

d.) Try to write a python function which finds the optimal number of clusters for this dataset.
The elbow method is a popular technique for determining the appropriate number of clusters for k-means clustering. It does so by calculating the inertia, which represents the sum of squared distances between data points and their assigned cluster centroids for various values of k. The code starts by iteratively applying k-means clustering for k values ranging from 1 to m, calculating the inertia for each k. The results are then plotted on a graph, with k on the x-axis and inertia on the y-axis. The "elbow point" in the graph is the value of k where the curve starts to flatten out, indicating that increasing the number of clusters does not significantly reduce inertia. This "elbow" in the graph represents the optimal number of clusters for the dataset, as it strikes a balance between minimizing intra-cluster distance and avoiding excessive cluster fragmentation.

However this method has very heavy computational cost on a large dataset such as the dataset used in this assignment. This is so because this method iteratively runs K Means for ‘m’ different values of K to calculate the inertia. So time complexity turns out to be **O(M*T*K*N*D).**
Here,
M is the number of different values of K being tried out
T is maximum number of iterations
K is the number of clusters
N is the number of datapoints
D is the number of attributes in the dataset.

The curve obtained is as follow:



Task 2:
Please visit the code:
https://colab.research.google.com/drive/1B8TBQqZJnxhcTUHrrTNjtjF8r-95DzpJ?usp=sharing

a.) Perform PCA on MNIST and then perform GMM clustering. (Library can be used for SVD and GMM) but PCA should be from scratch. PCA should be done for 32, 64 and 128 components.Clustering should be done in 10, 7, and 4 clusters.
In PCA, the first step is to perform Singular Value Decomposition (SVD) on the input data. SVD decomposes the data matrix into three separate matrices: U, S, and V^T. These matrices represent the left singular vectors, singular values, and right singular vectors, respectively. The code uses the ***linalg.svd*** function, which is part of the ***SciPy library***.

After performing SVD, the code selects the top 'K' principal components from the V^T matrix. In the code, it slices the V^T matrix to retain only the first K columns. These top principal components capture the most significant variations in the data.

To reduce the dimensionality of the data, the code then projects the original data onto the selected principal components. This is done using matrix multiplication (dot product) between the original data and the transposed top principal components matrix.
Finally, the function returns the projected data, which has been reduced to 'K' dimensions while retaining the most important information from the original dataset.

Now this reduced data is used to perform the clustering using Gaussian Mixture Model. The gaussian mixture model is implemented using the sklearn library.

Algorithm of PCA using SVD:
Principal Component Analysis (PCA) Algorithm:

Input:
- **data:** The input data matrix
- **num_components:** The number of principal components to retain

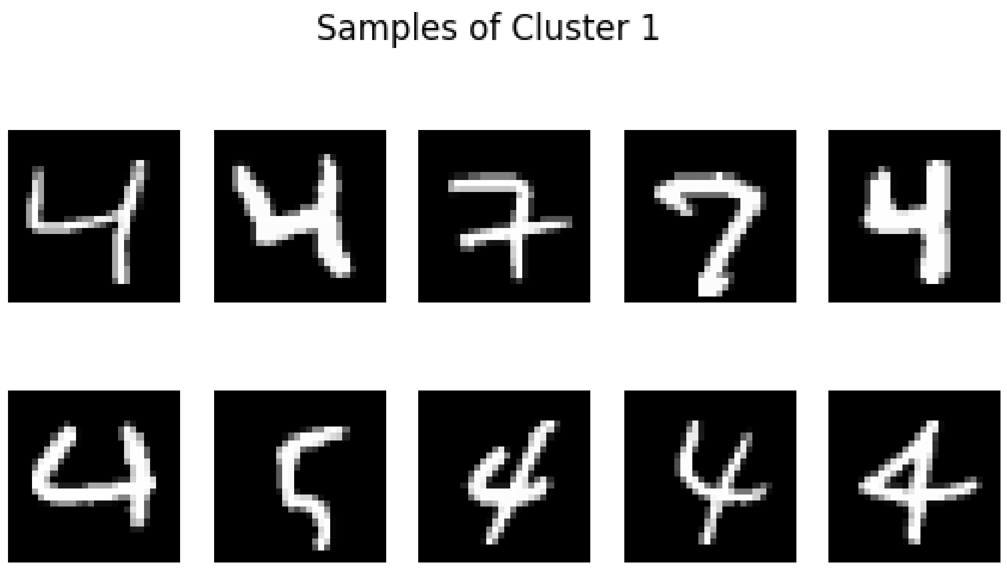
Output:
- Return the **'projected_data'**

.
Steps:
1. Perform Singular Value Decomposition (SVD):
2. Select the Top Principal Components:
3. Project the Data:

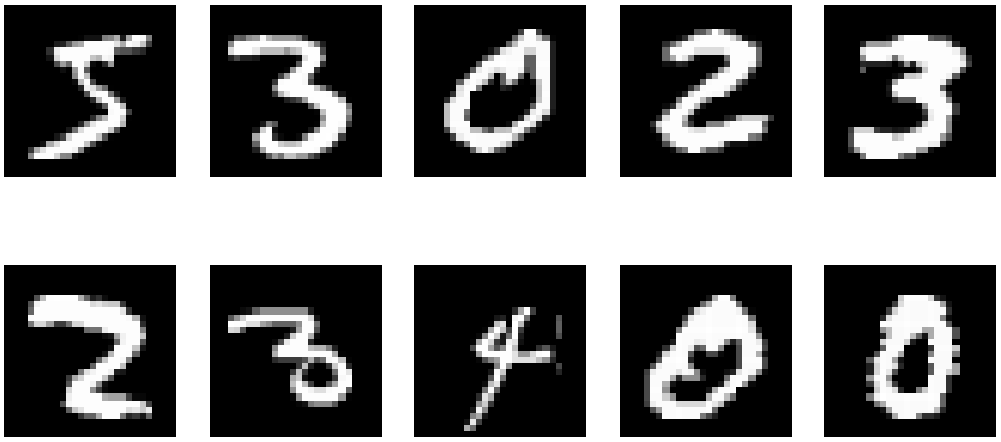
Time complexity of PCA using SVD:
The overall time complexity of the PCA algorithm is dominated by the SVD step, which is typically the most time-consuming part. The complexity is roughly O(n * m^2).
Here,
m is the number of features
n is the number of datapoints

b.) Visualize the images getting clustered in different clusters.

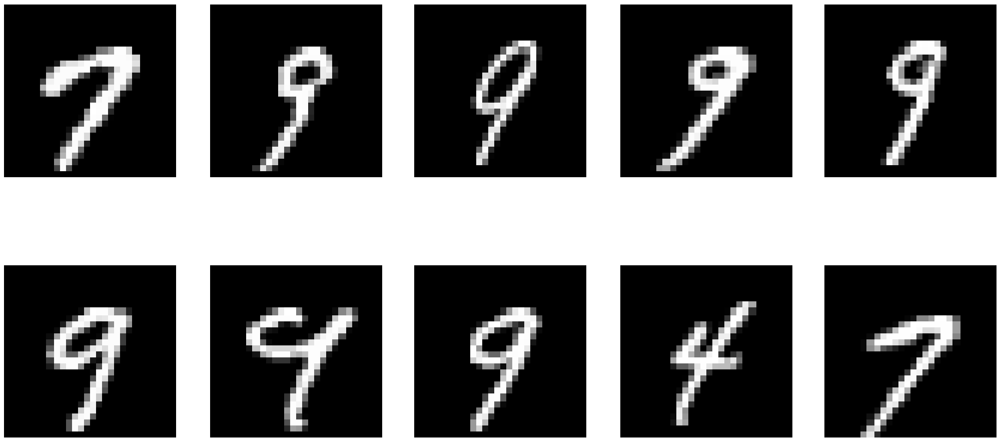
1. **When Number of PCA Component is 128 and Number of cluster is 10:**



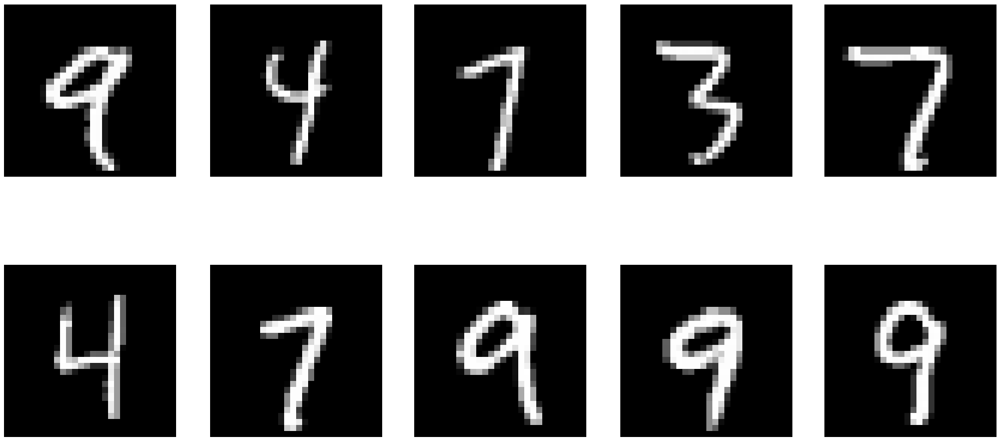
Samples of Cluster 2



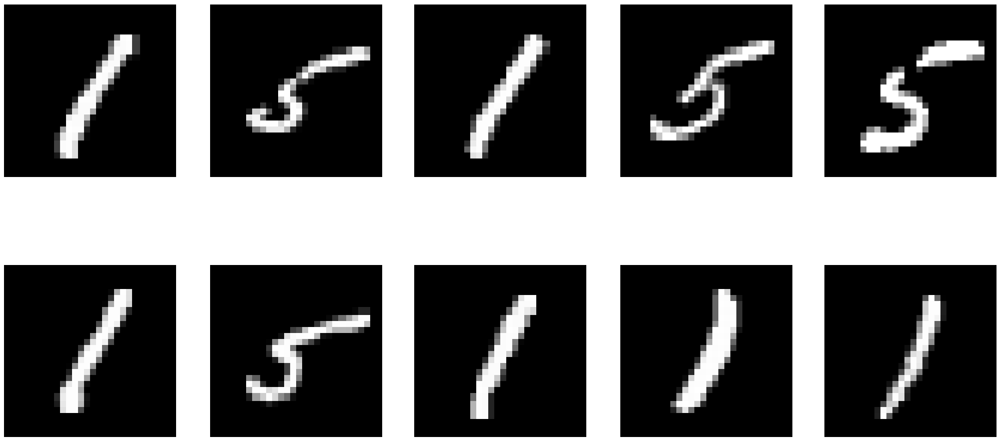
Samples of Cluster 3



Samples of Cluster 4

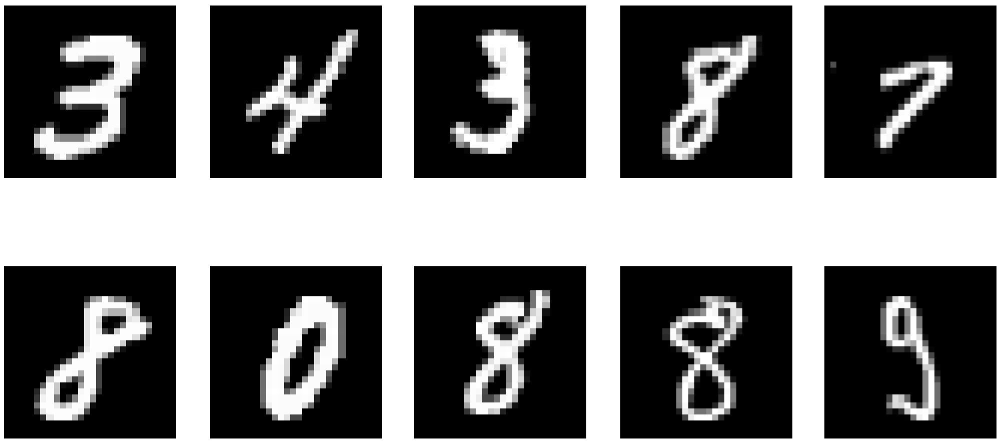


Samples of Cluster 5

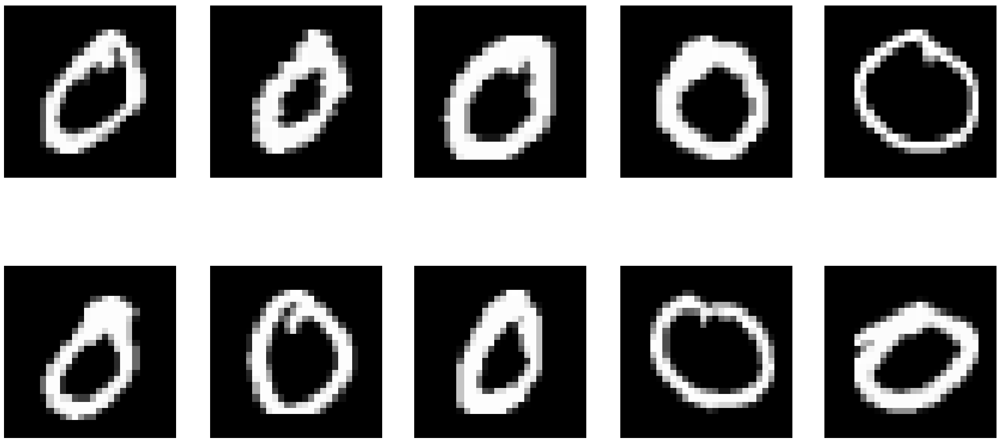


Samples of Cluster 6

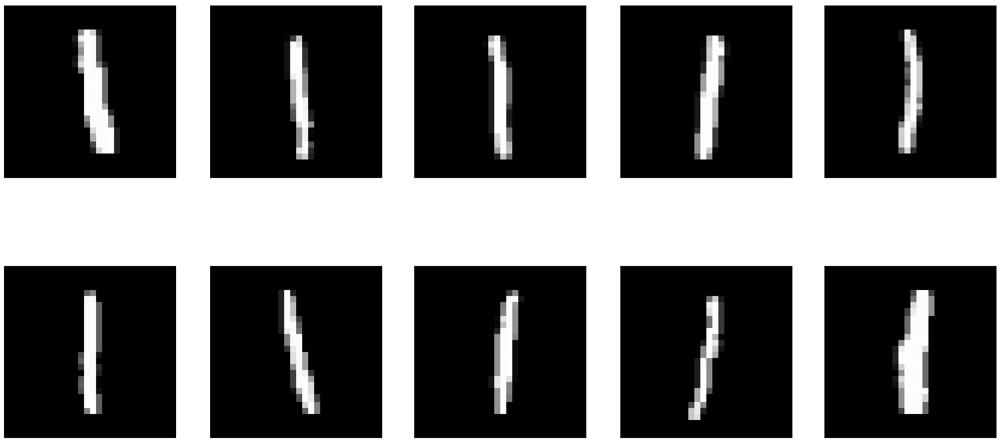
Samples of Cluster 7



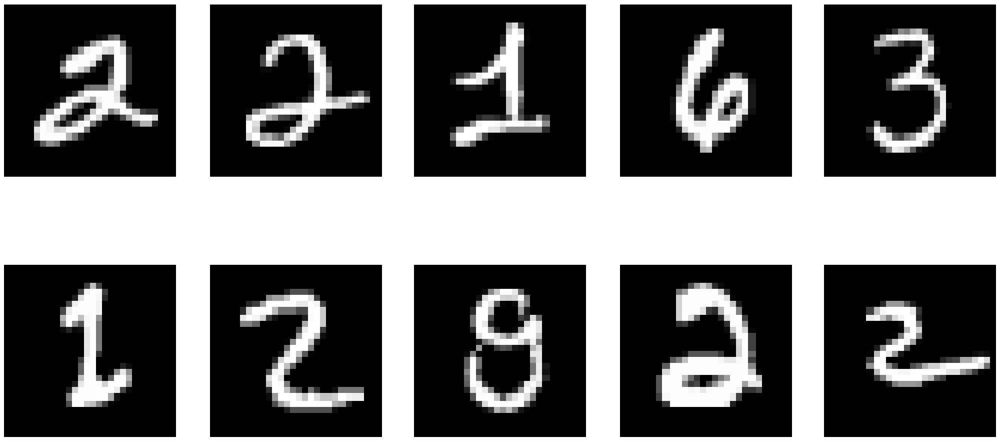
Samples of Cluster 8



Samples of Cluster 9

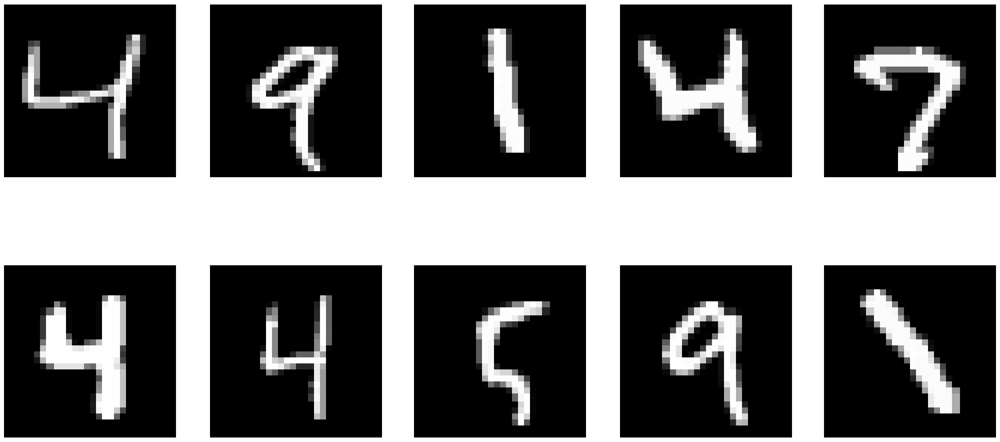


Samples of Cluster 10

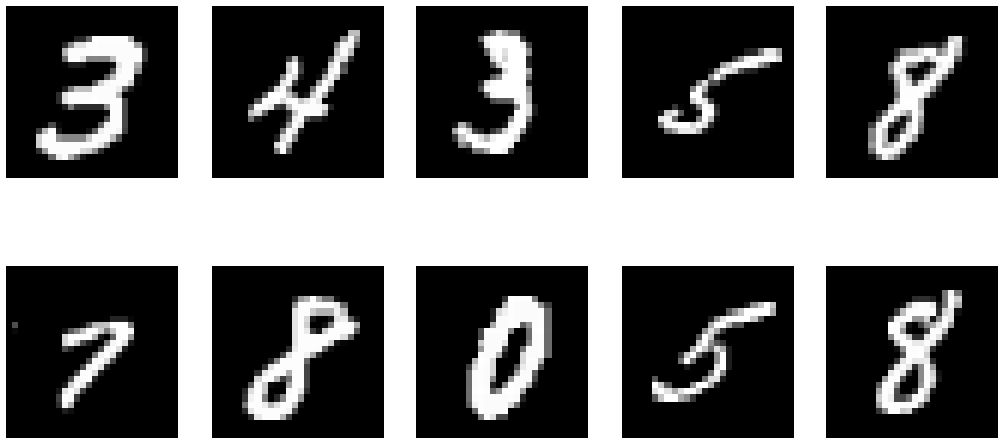


2. **When Number of PCA Component is 128 and Number of cluster is 7:**

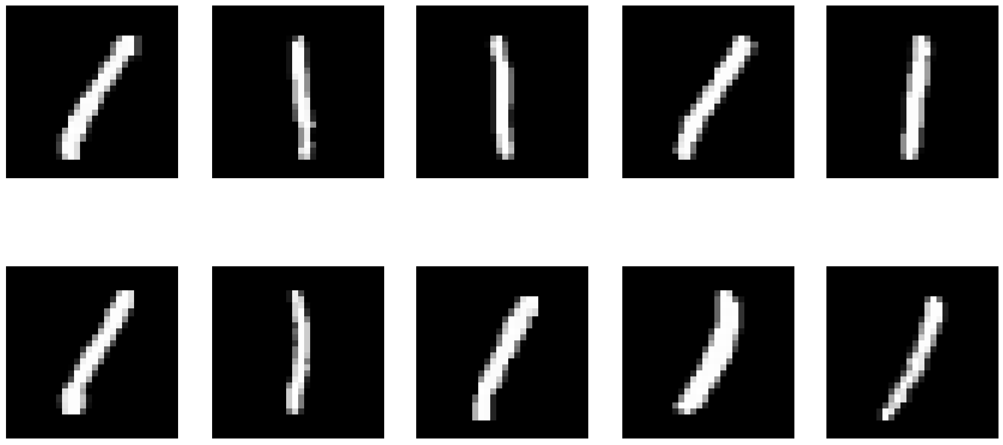
Samples of Cluster 1



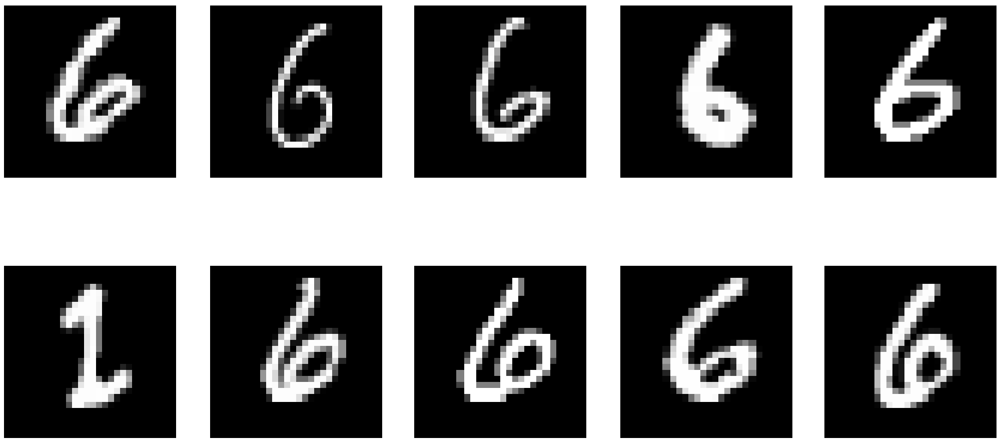
Samples of Cluster 2



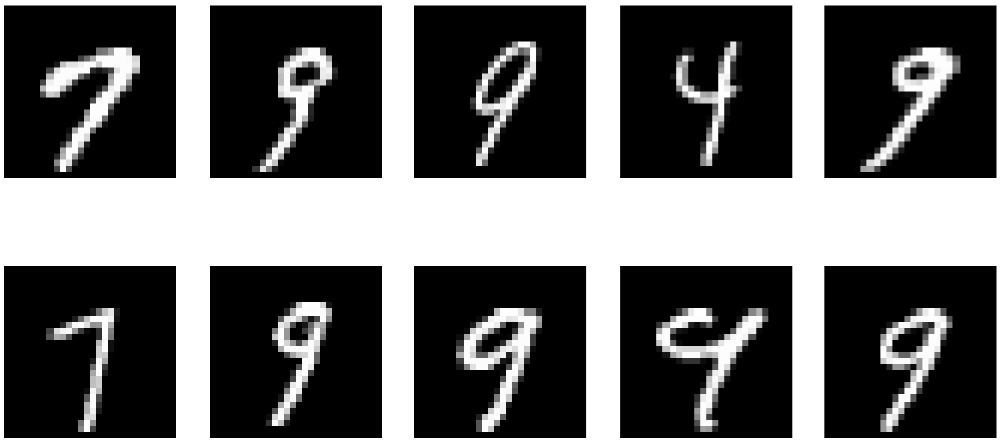
Samples of Cluster 3



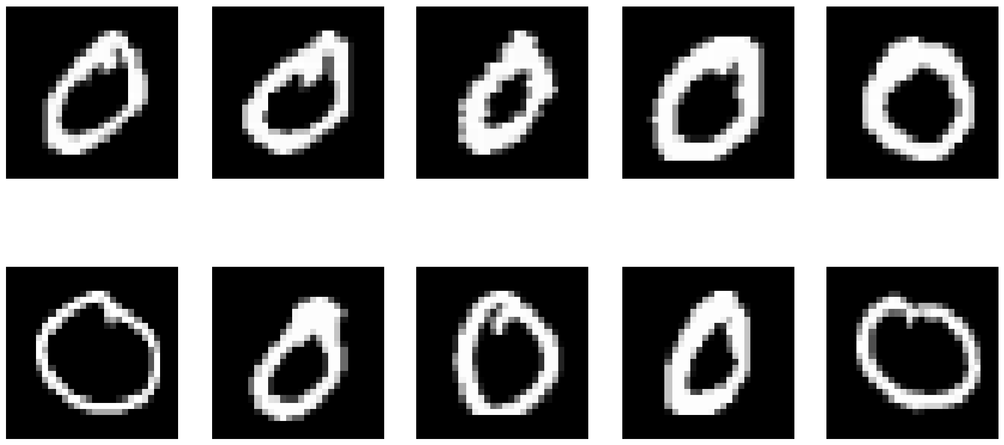
Samples of Cluster 4



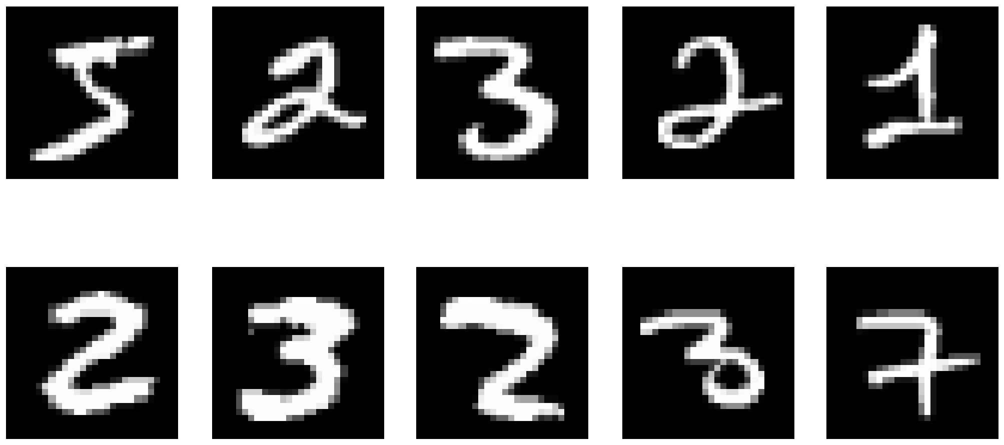
Samples of Cluster 5



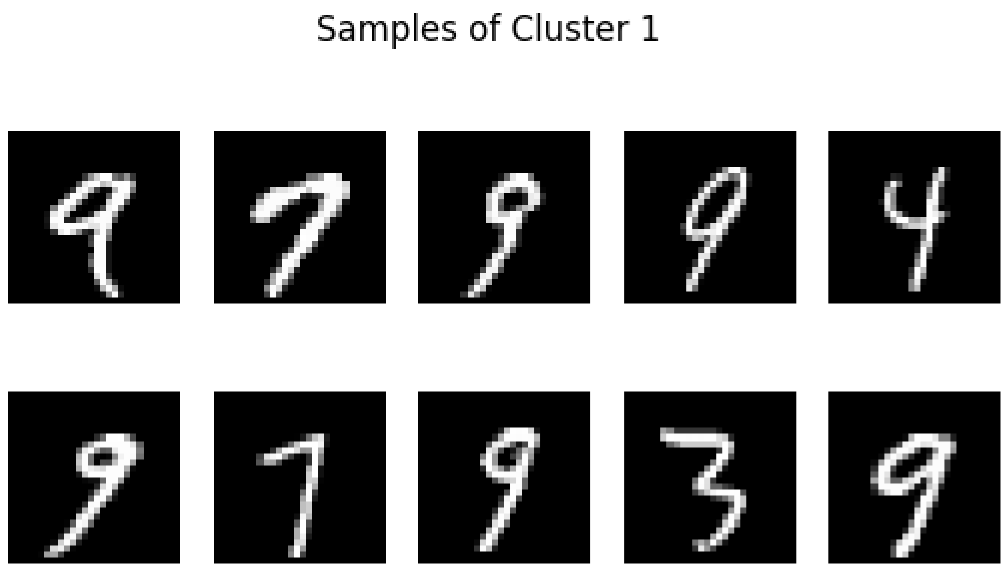
Samples of Cluster 6

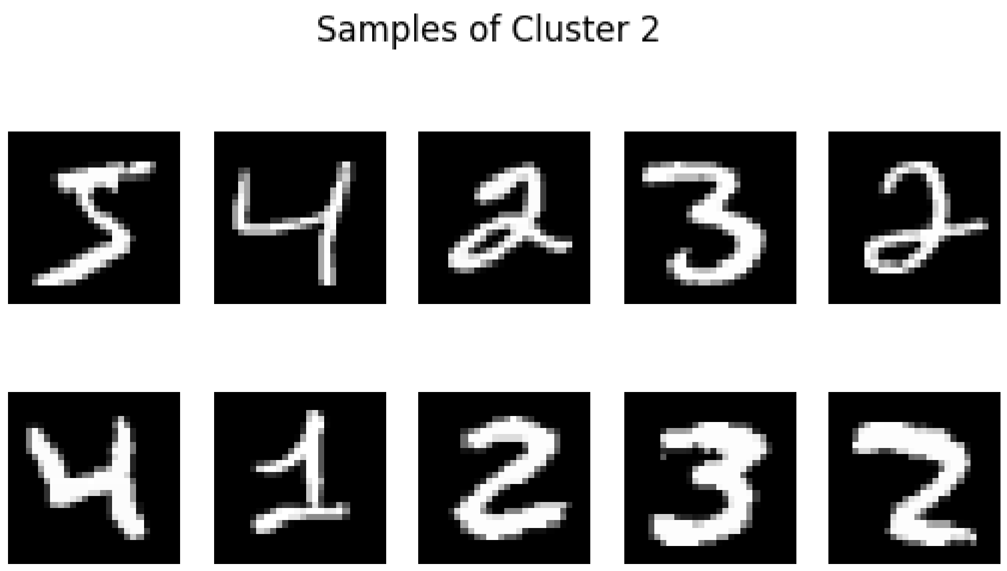


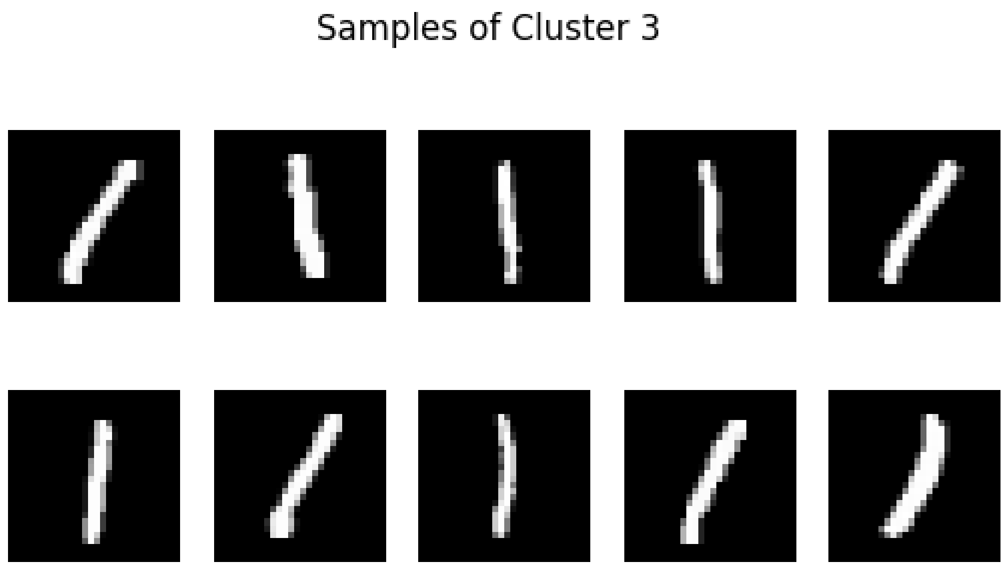
Samples of Cluster 7

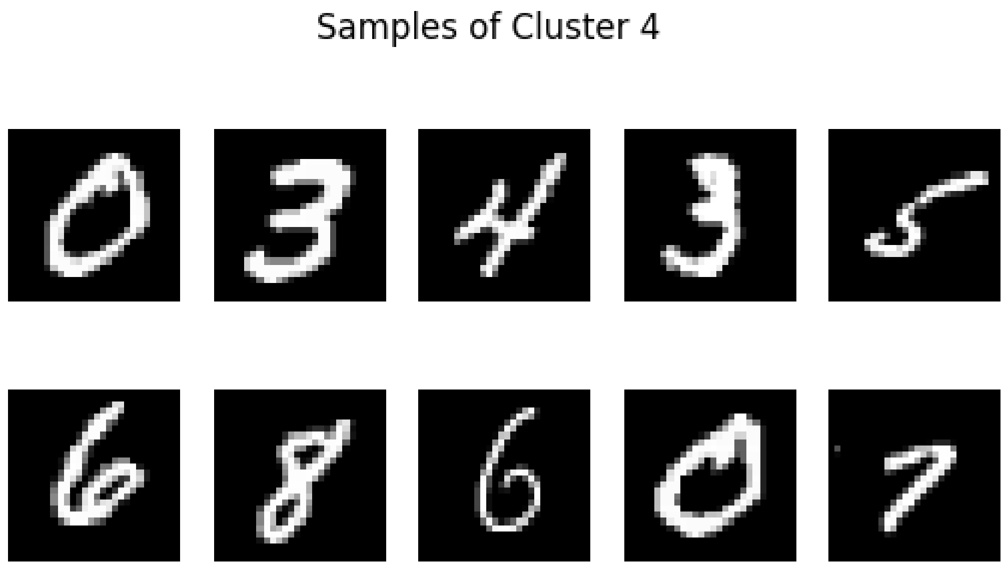


3. **When Number of PCA Component is 128 and Number of cluster is 4:**



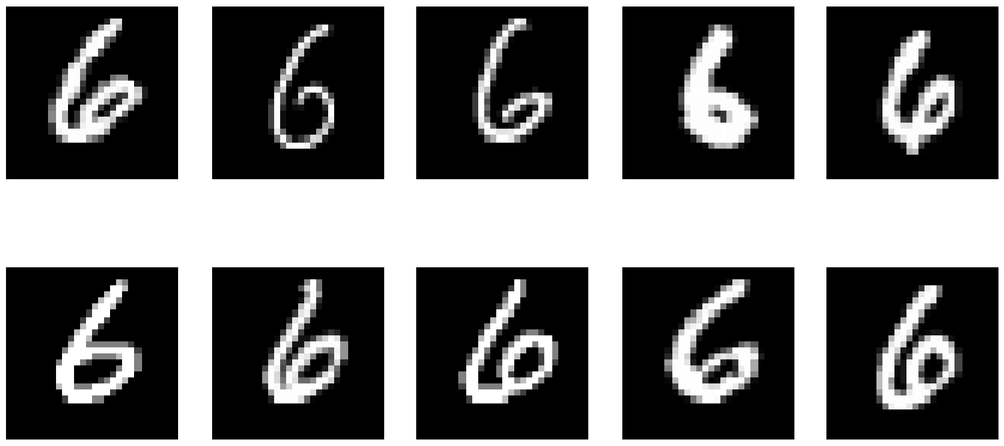



Samples of Cluster 3

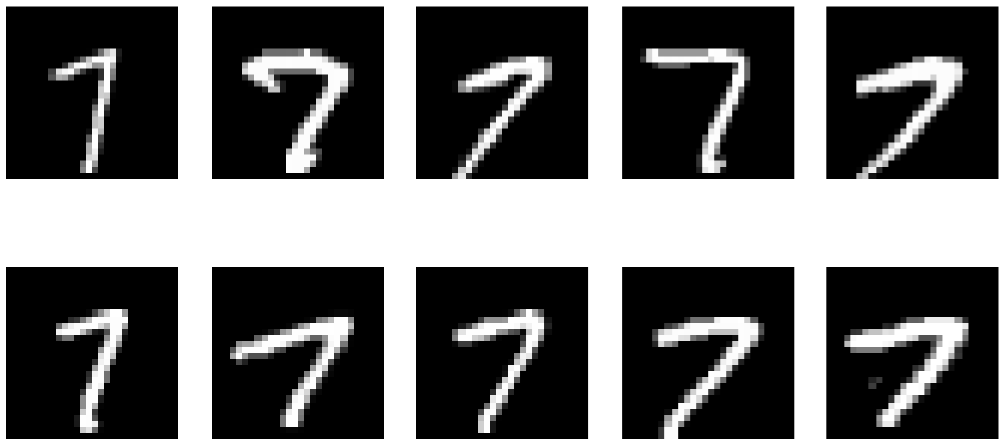

Samples of Cluster 4

4. **When Number of PCA Component is 64 and Number of cluster is 10:**

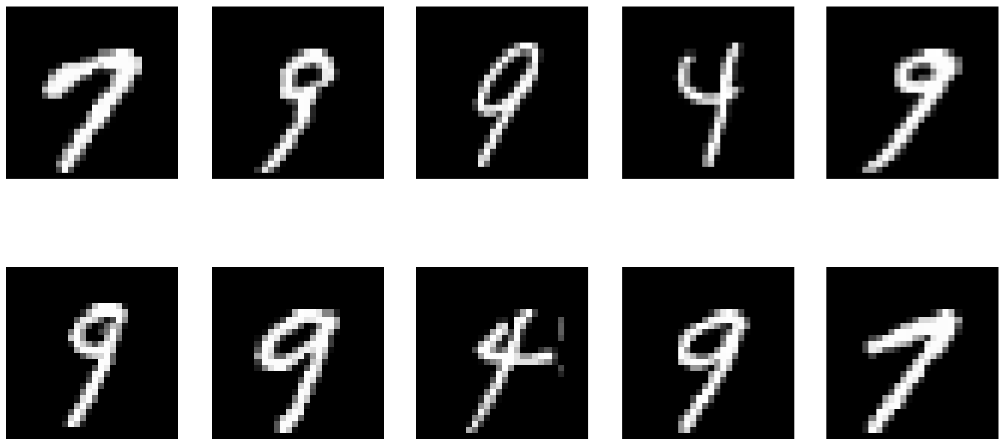
Samples of Cluster 1



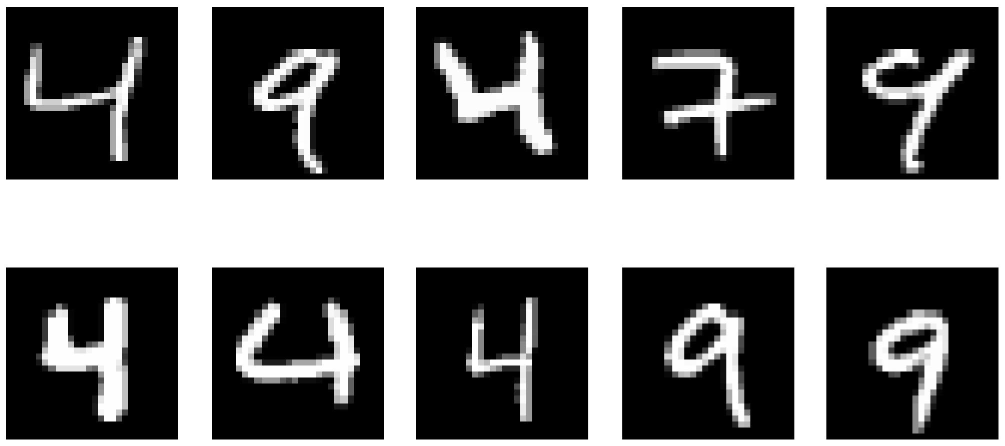
Samples of Cluster 2



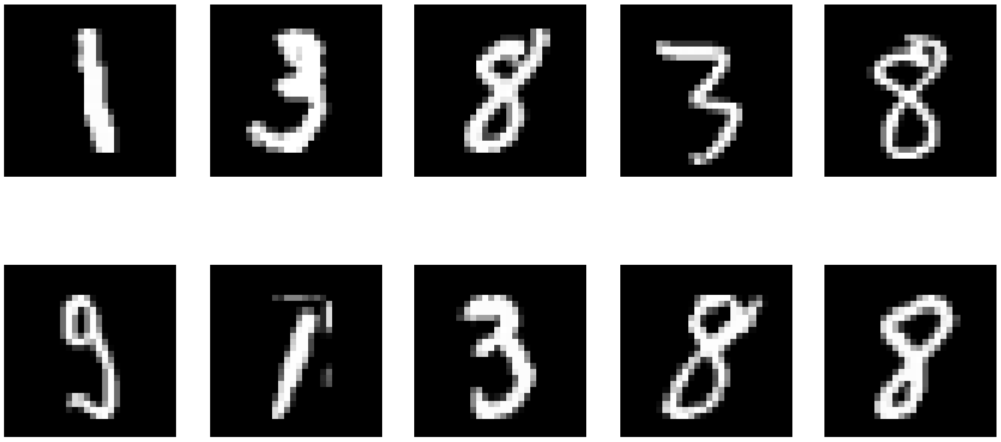
Samples of Cluster 3



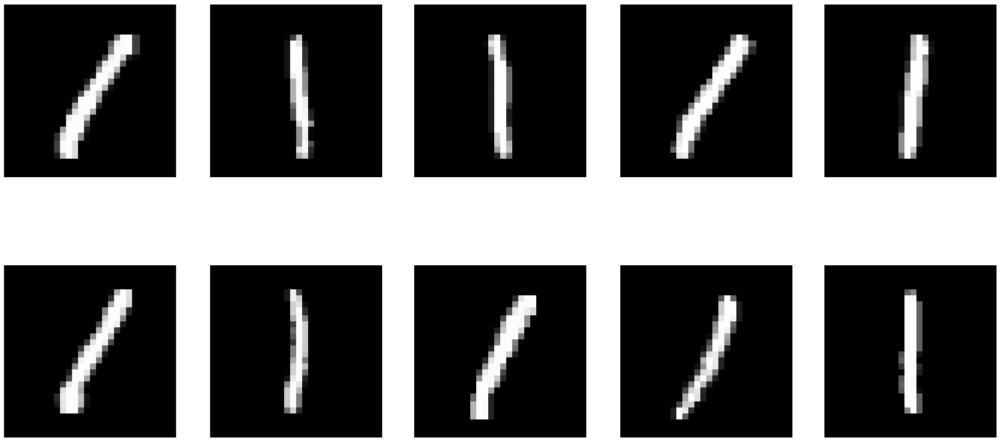
Samples of Cluster 4



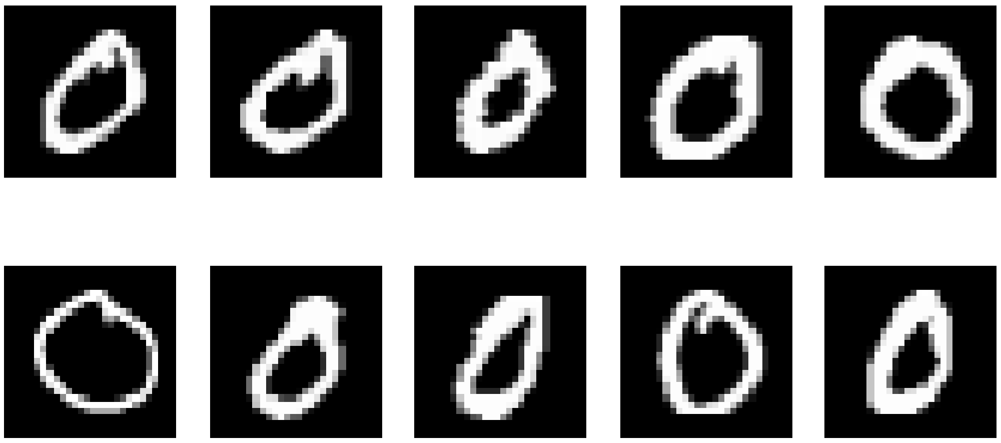
Samples of Cluster 5



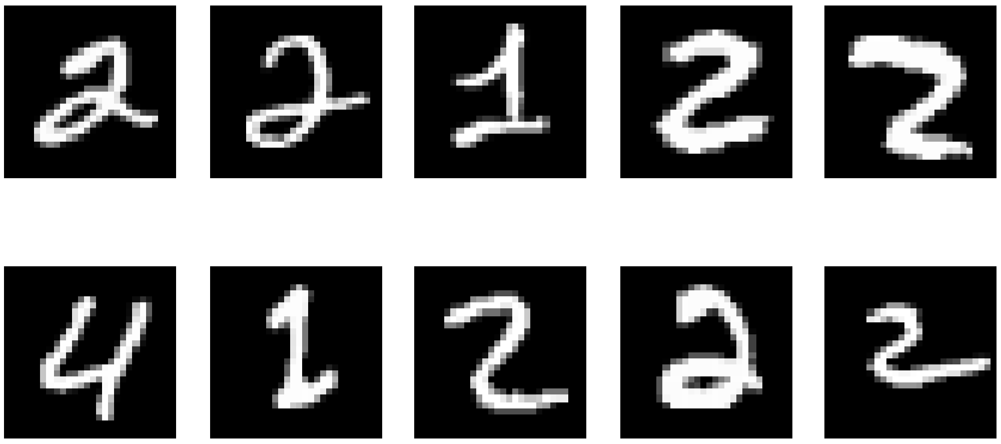
Samples of Cluster 6



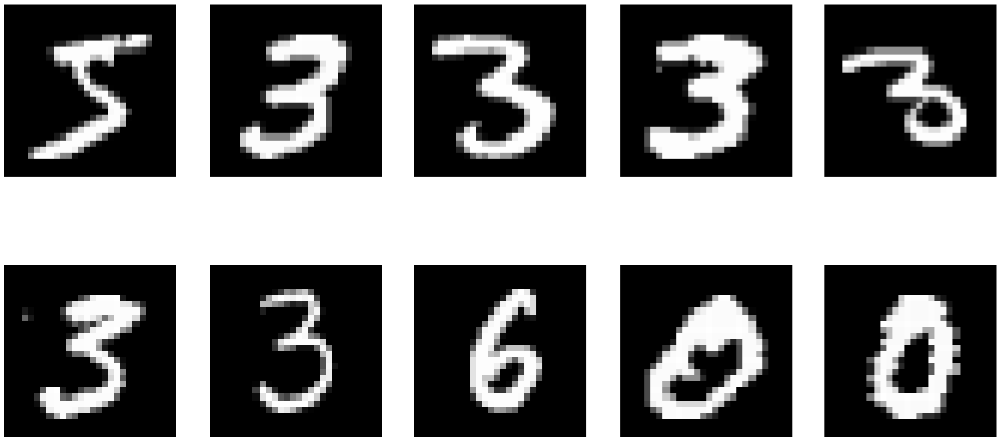
Samples of Cluster 7



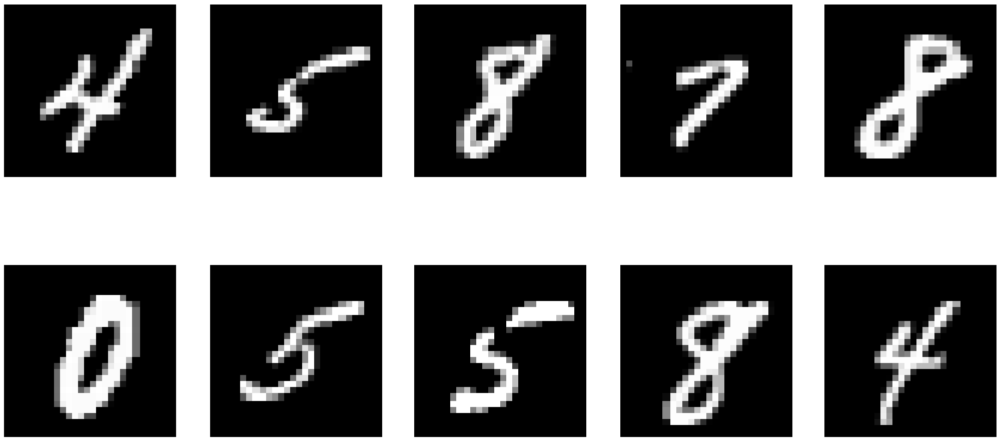
Samples of Cluster 8



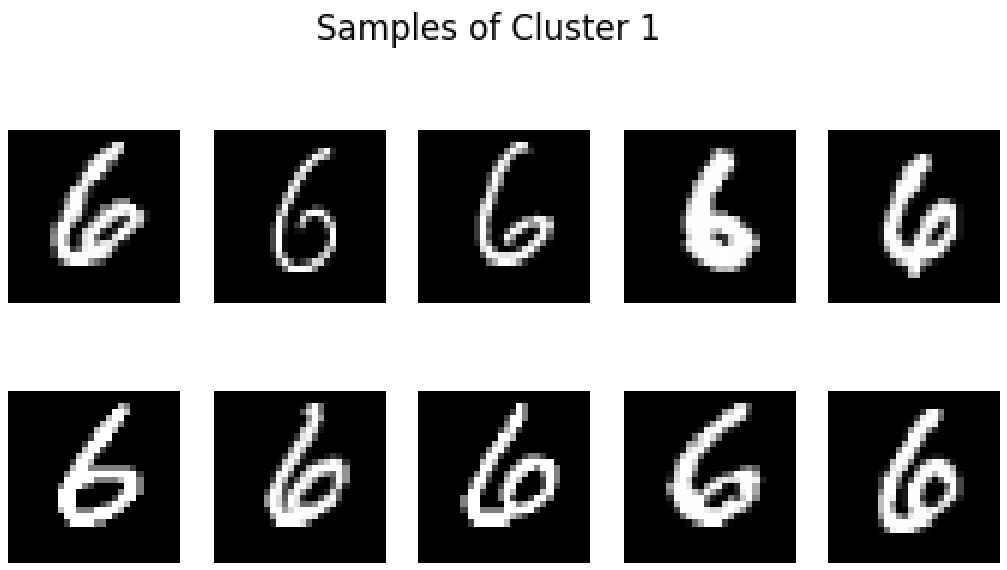
Samples of Cluster 9

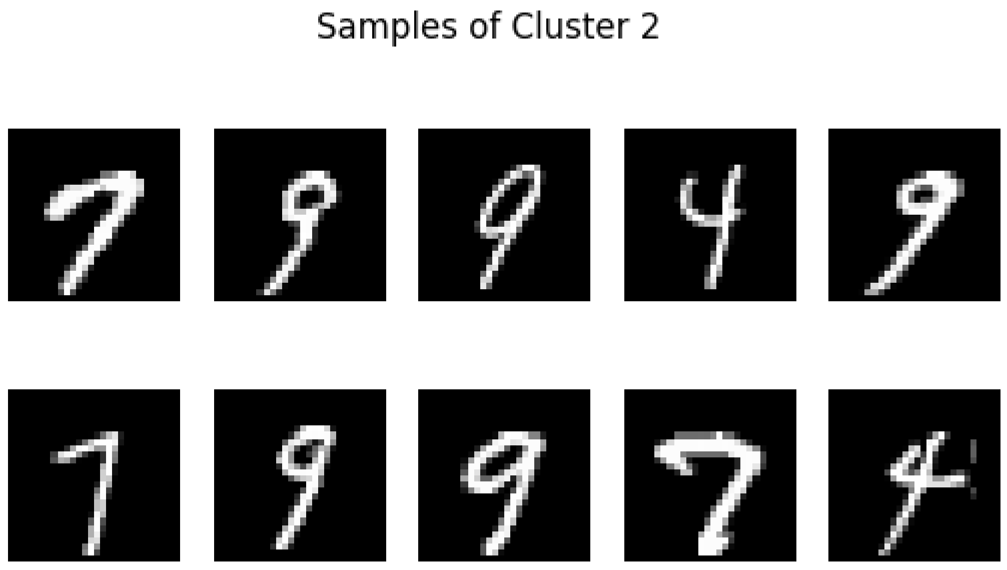


Samples of Cluster 10



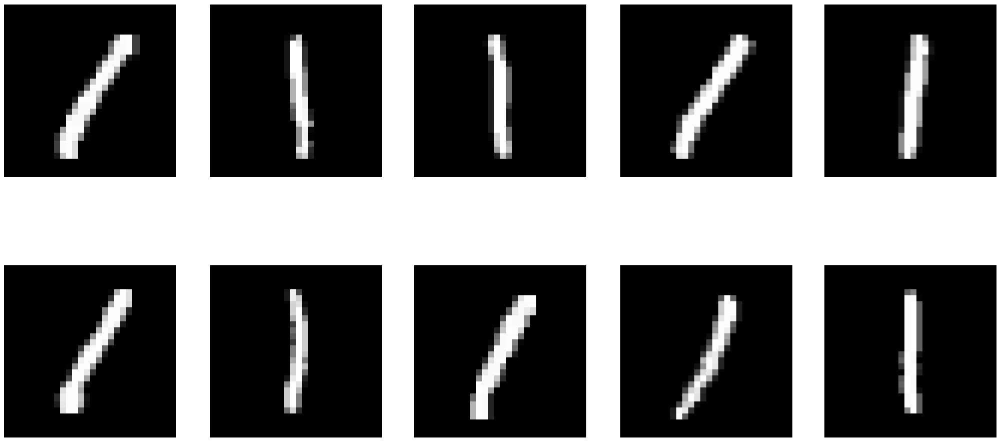
5. **When Number of PCA Component is 64 and Number of cluster is 7:**



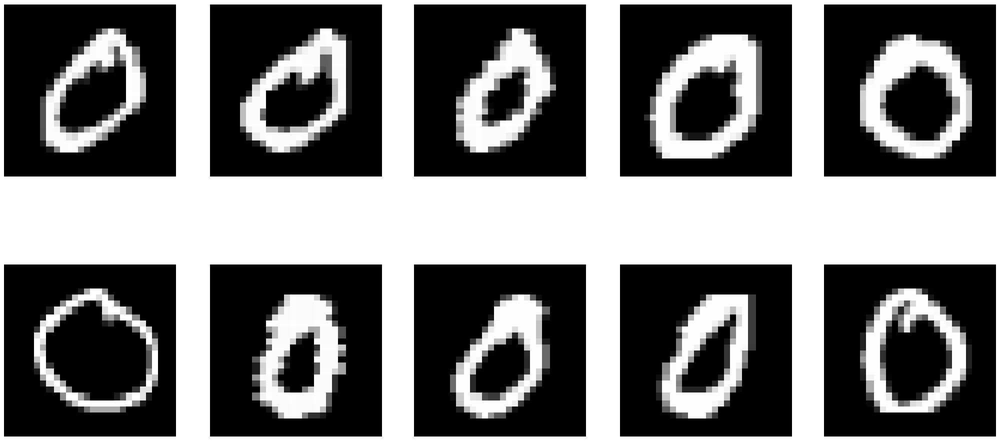


Samples of Cluster 3

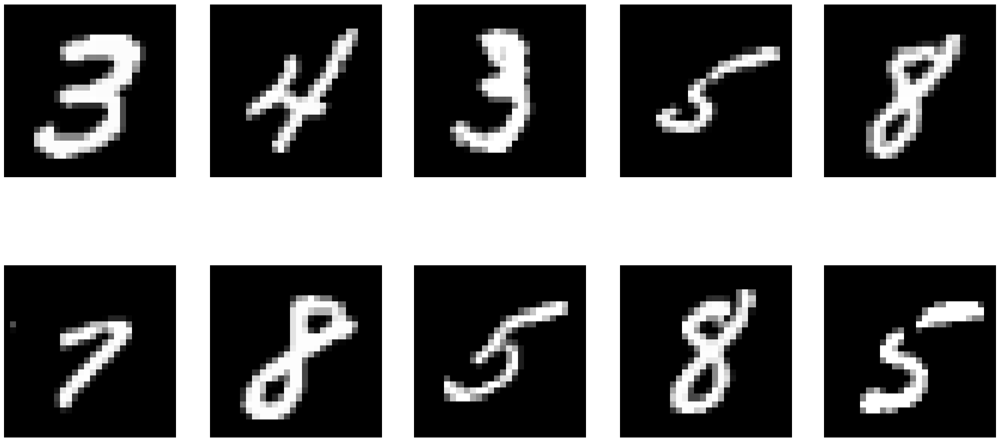
Samples of Cluster 4

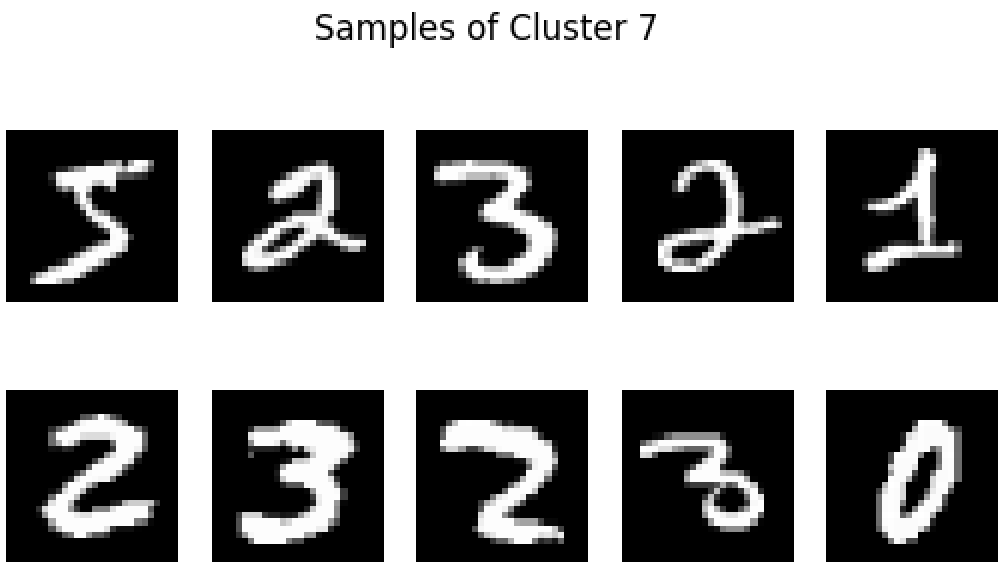


Samples of Cluster 5

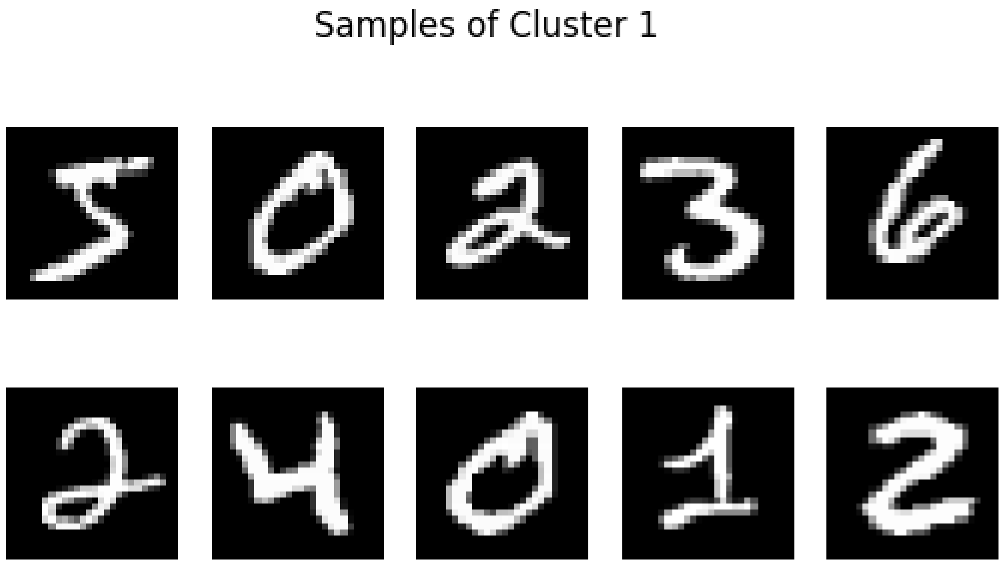


Samples of Cluster 6

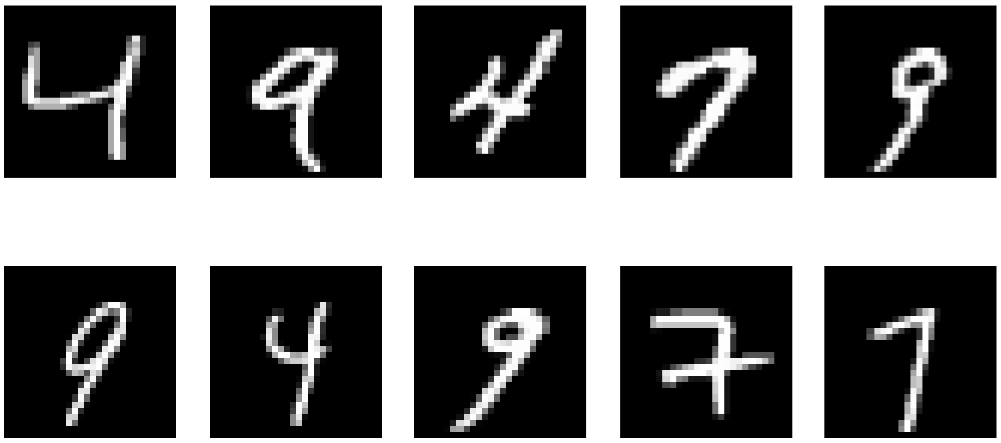




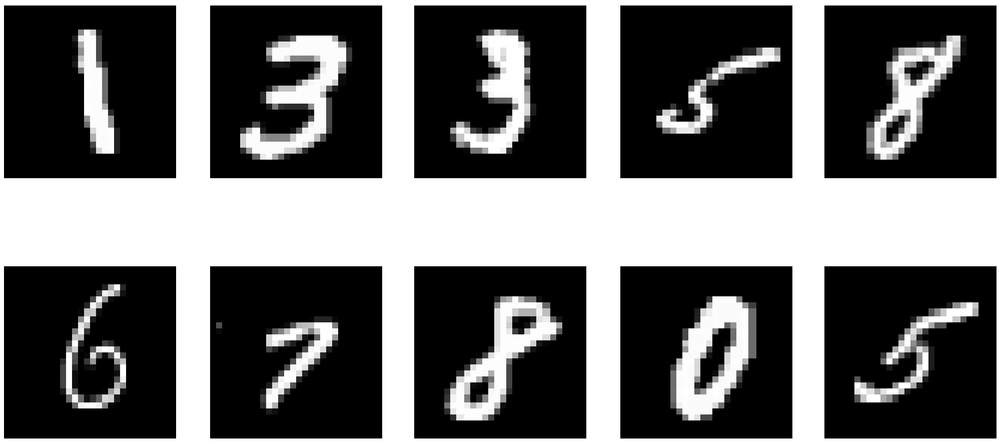
6. **When Number of PCA Component is 64 and Number of cluster is 4:**

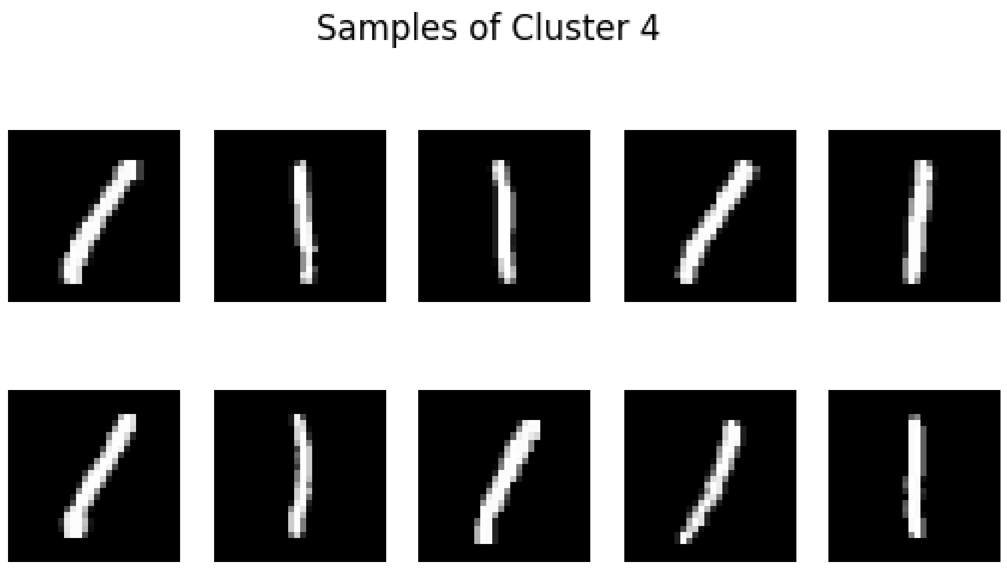


Samples of Cluster 2

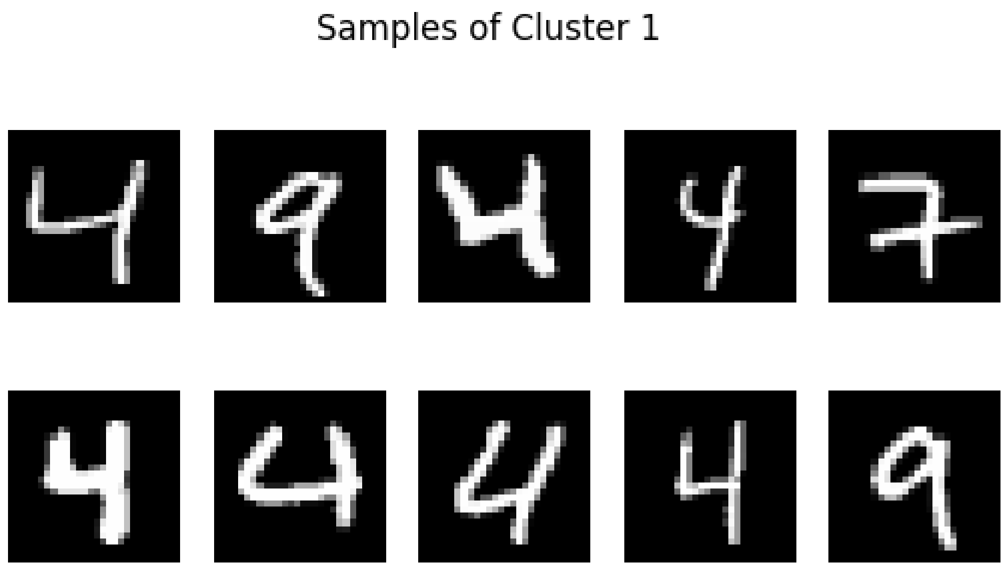


Samples of Cluster 3

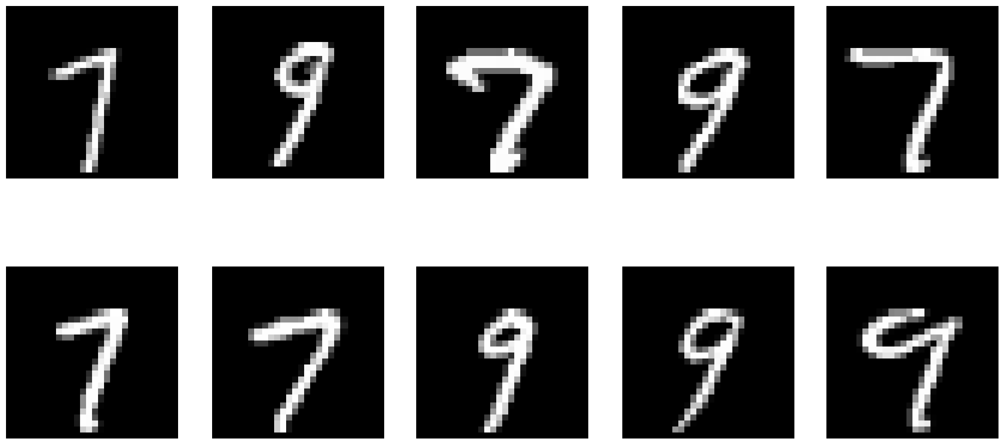




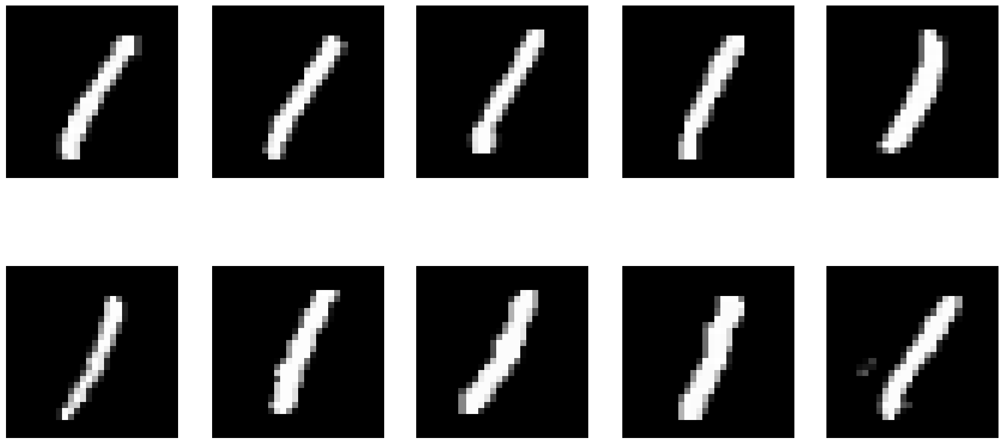
7. **When Number of PCA Component is 32 and Number of cluster is 10:**



Samples of Cluster 2



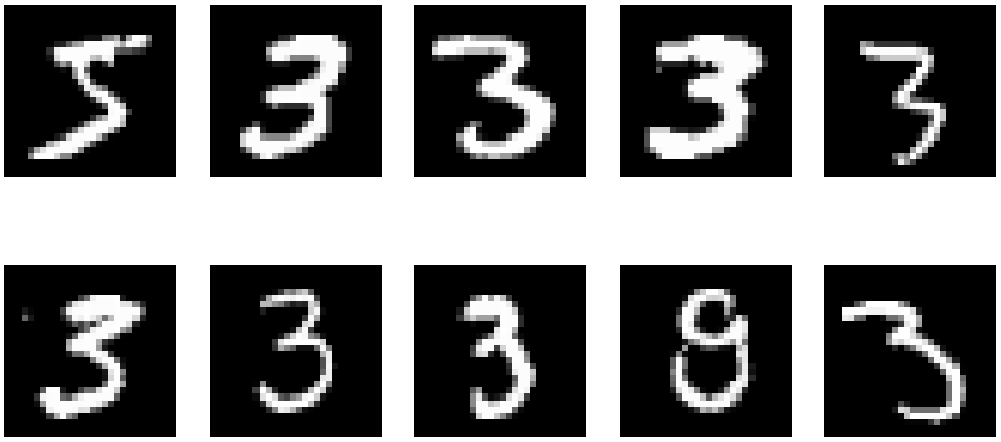
Samples of Cluster 3



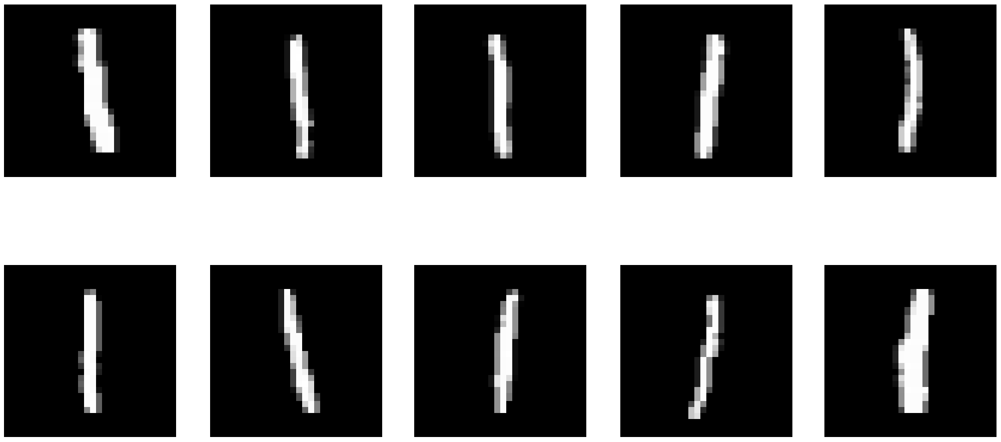
Samples of Cluster 4

Samples of Cluster 5

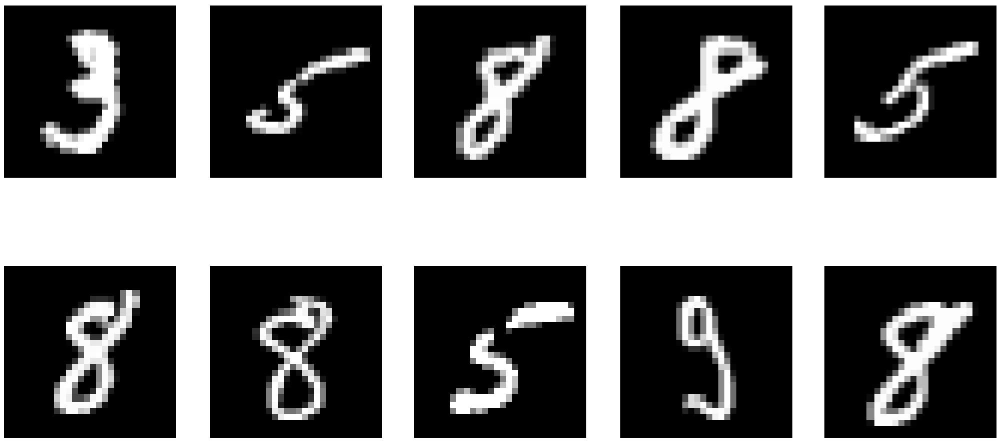
Samples of Cluster 6



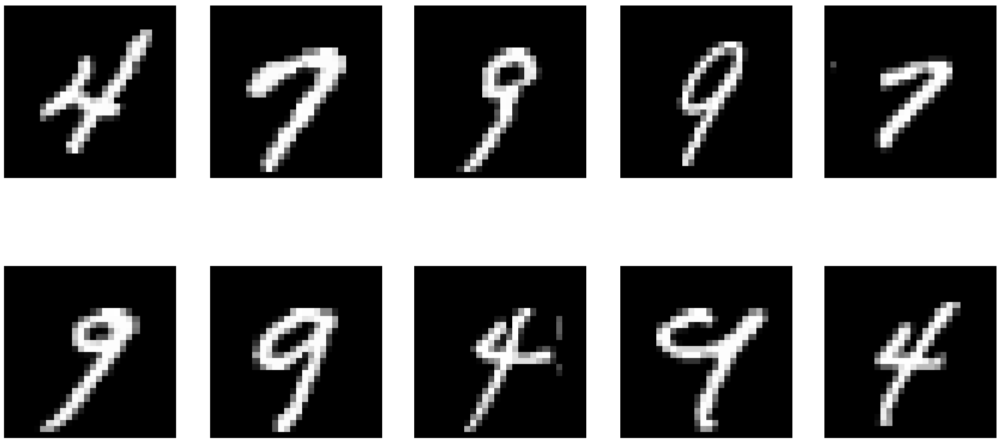
Samples of Cluster 7



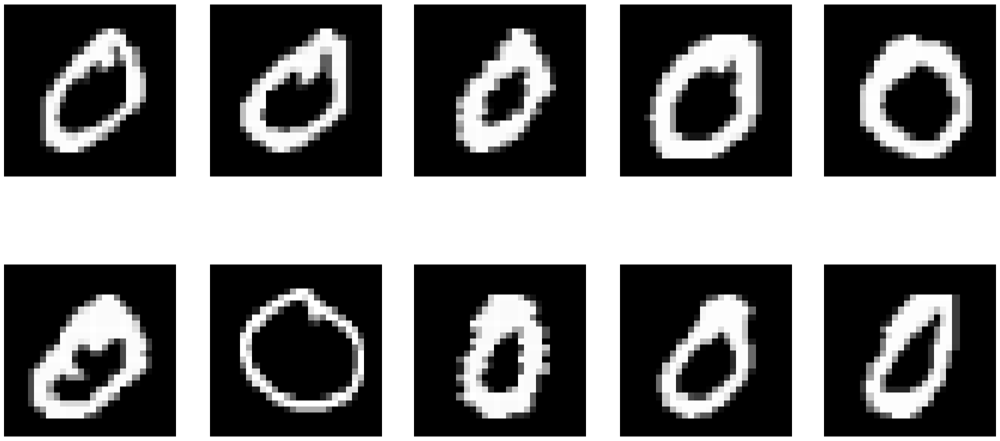
Samples of Cluster 8



Samples of Cluster 9

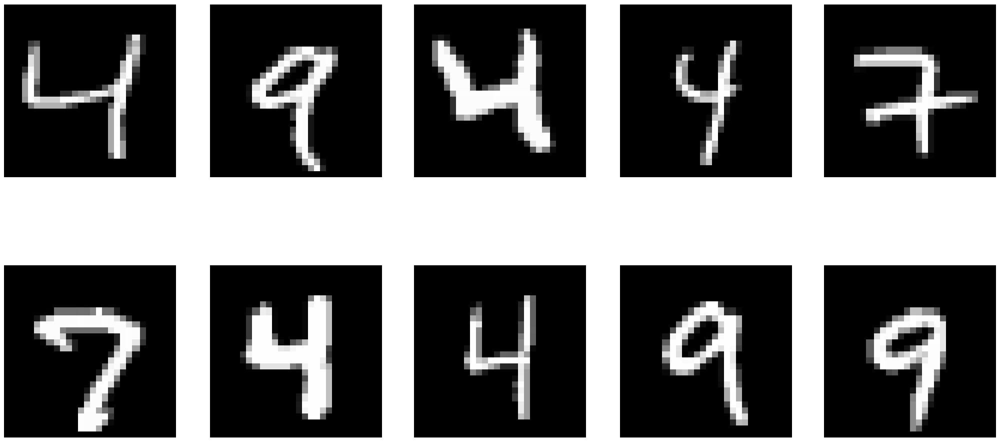


Samples of Cluster 10

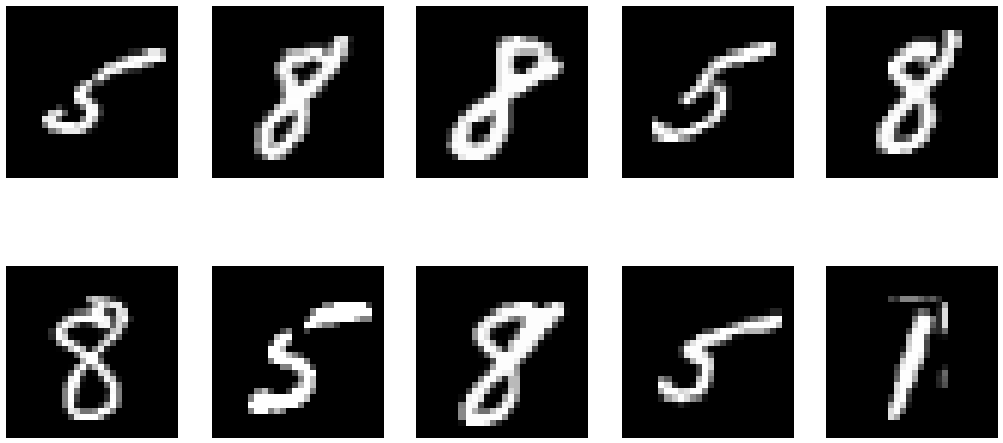


8. **When Number of PCA Component is 32 and Number of cluster is 7:**

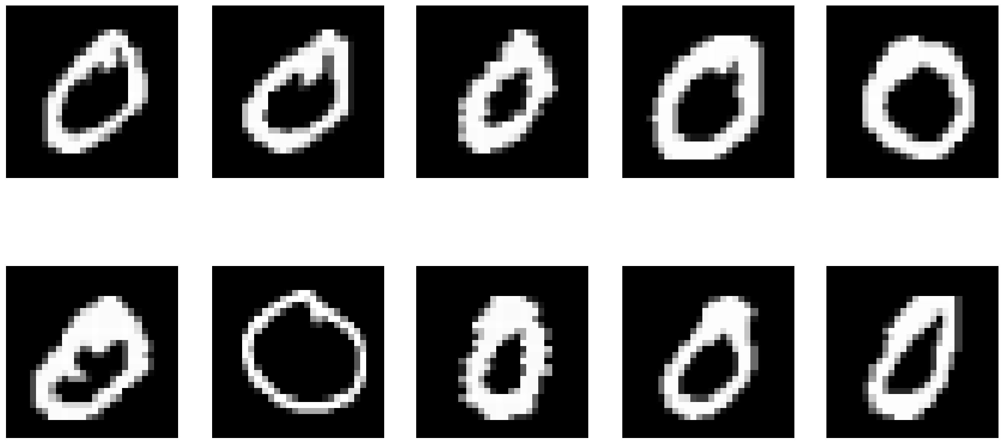
Samples of Cluster 1



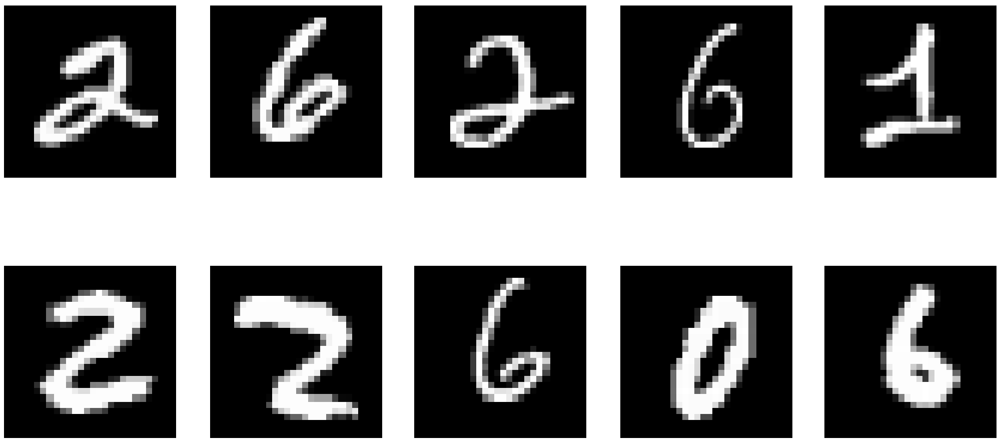
Samples of Cluster 2



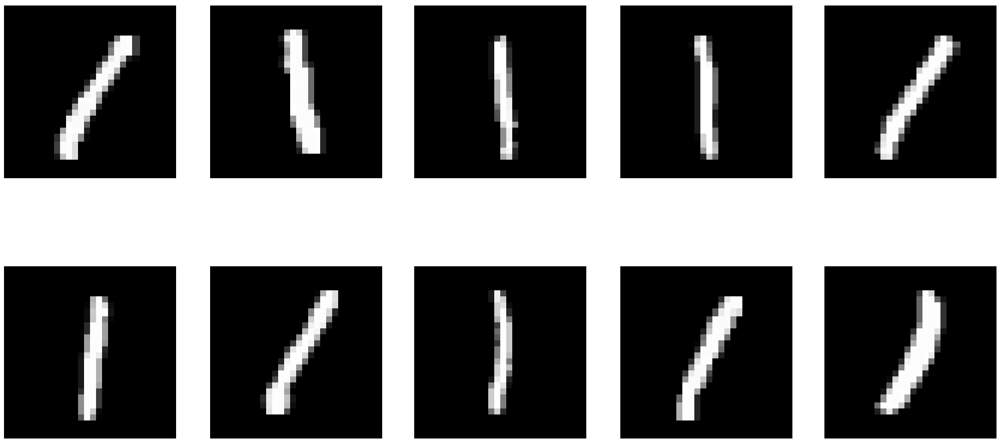
Samples of Cluster 3



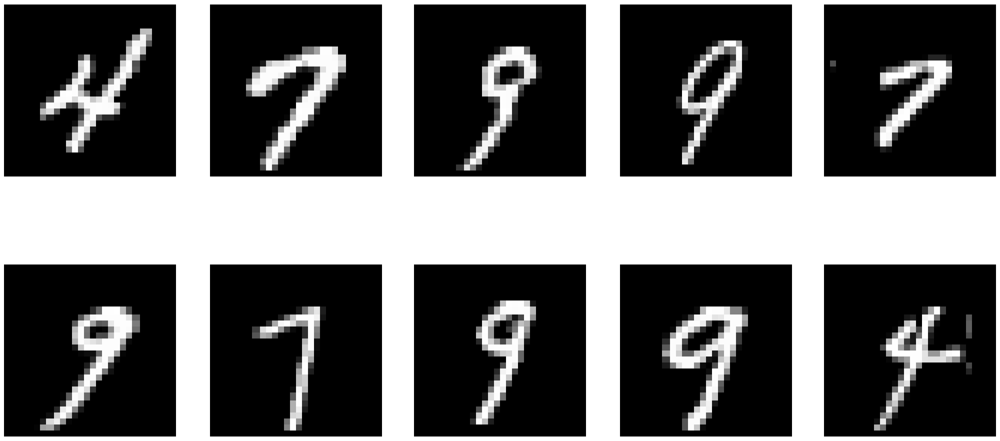
Samples of Cluster 4



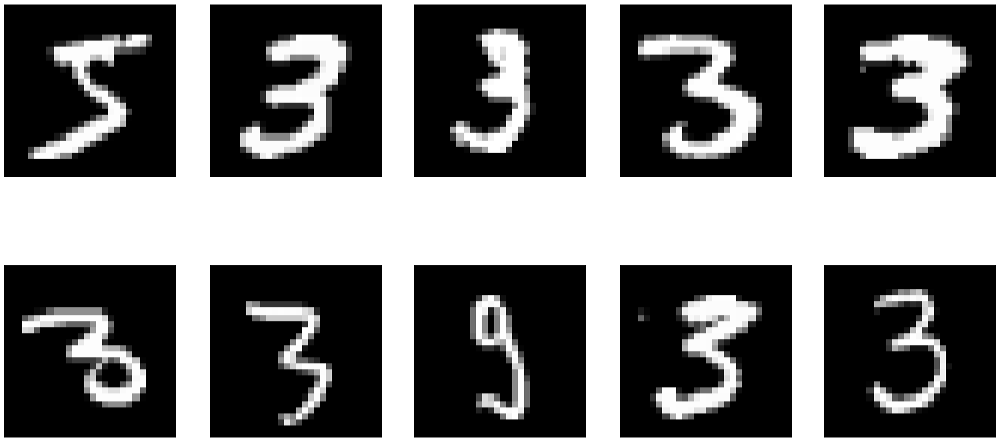
Samples of Cluster 5



Samples of Cluster 6

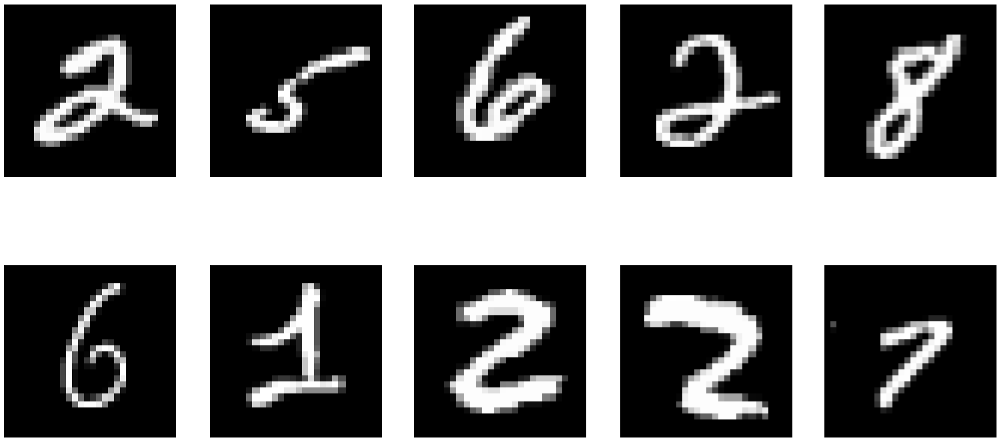


Samples of Cluster 7

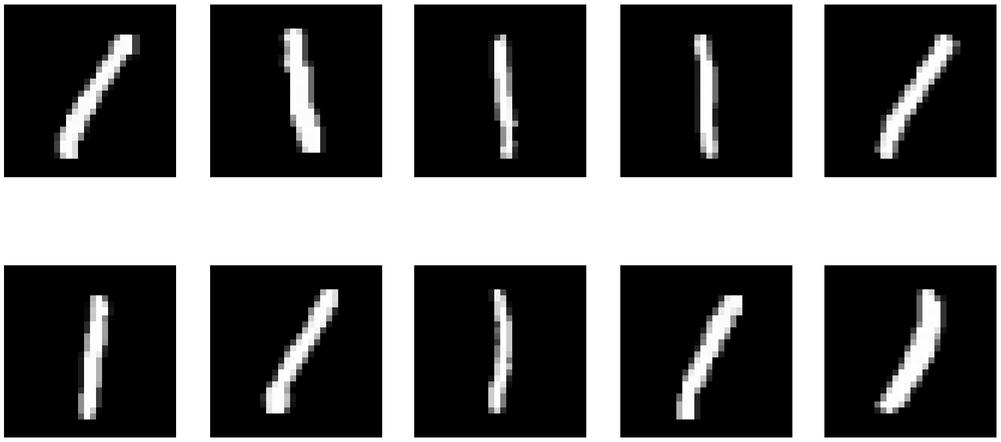


9. **When Number of PCA Component is 32 and Number of cluster is 4:**

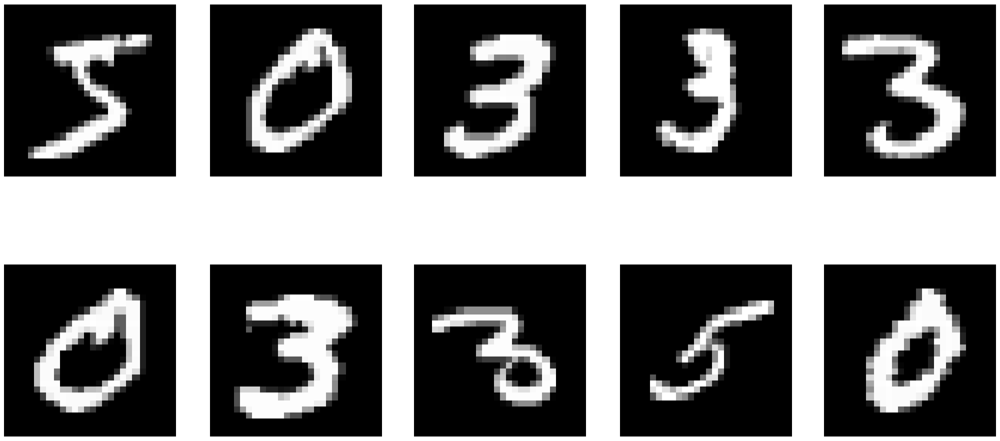
Samples of Cluster 1



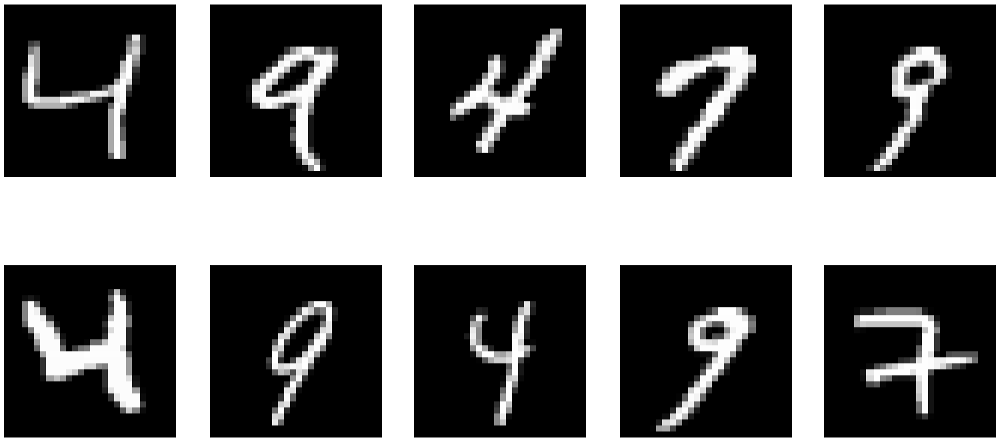
Samples of Cluster 2



Samples of Cluster 3



Samples of Cluster 4



c.) Please comment on cluster characteristics and comment with respect to previous task about what difference you see.
PCA reduces the dimensionality of the MNIST dataset, which is originally high-dimensional. Reducing dimensionality can help in capturing the most significant features while reducing noise.
Secondly, GMM is a probabilistic clustering method, which means that each data point can belong to multiple clusters with associated probabilities. This leads to more flexible and nuanced cluster assignments compared to hard clustering methods like k-means.
GMM can produce overlapping clusters, and this is especially useful when data points exhibit complex, overlapping patterns. It's possible that some digits may have high probabilities of belonging to multiple clusters, indicating ***variations in writing styles or digit appearances***.

Also, different PCA components (32, 64, and 128) are used to explore how dimensionality reduction impacts the clustering results. Higher PCA components capture more variance but may also introduce more complexity.

GMM can be more robust to noisy data compared to k-means because it models data as a combination of Gaussian distributions, which are less affected by outliers.

The characteristics of clusters can vary based on the combination of PCA components and the number of clusters. Clusters can represent variations in writing styles, distinct digit appearances, or a mixture of these aspects.

K-means is likely to create non-overlapping clusters where each digit is assigned to a single cluster.
The clusters may have similar sizes and shapes, and the characteristics of each cluster correspond to the mean appearance of the digits within the cluster.
K-means can be sensitive to the scale and initialization, leading to different results in different runs.

d.) Can you find the optimal number of components the PCA should choose which covers almost all the necessary patterns in the data? Can you comment on where PCA can fail?
To calculate the optimal number of components of pca, I wrote a function. Although it is like a brute force method to calculate it.
The function takes two main arguments: data, which represents the dataset you want to perform PCA on, and threshold, which is a user-defined threshold (defaulted to 0.95). The threshold signifies the proportion of variance that you want to retain in the data after dimensionality reduction; typically, this is set to a high value like 0.95 to retain 95% of the variance.

Inside the function, it iterates through different numbers of principal components, starting from 1 and incrementing by 5 at each step (it can be iterated 1 at each step for more accurate result but computationally extensive). For each number of components, it calculates the cumulative explained variance, which quantifies the amount of information retained by that many principal components. The explained variance is computed by taking the sum of variances of the projected data along each principal component axis and dividing it by the total variance of the original data.

The code continues this process until it finds a number of components where the cumulative explained variance meets or exceeds the specified threshold. Once this condition is met, it returns the number of components as the optimal choice.

After running this, I got that to capture 95 percent of the variance, the minimum value of the component is **156**.

PCA can fail under the following situations.

1. PCA fails in presence of outliers. PCA is sensitive to outliers. Even a small number of outliers in the data can disproportionately influence the principal components. This can lead to principal components being skewed towards the outliers and not capturing the main data structure effectively.
2. PCA is sensitive to the scale of the input features. If the features have significantly different scales, PCA may be influenced more by features with larger variances, potentially leading to suboptimal results. Standardizing or normalizing the data is often necessary to mitigate this issue.