Project 3: Unsupervised Learning

Snehashish Mandal
Department of Computer Science
University at Buffalo
Buffalo, NY 14214
snehashi@buffalo.edu

Abstract

The goal of this project is to perform cluster analysis on fashion MNIST dataset using unsupervised learning. Cluster analysis is one of the unsupervised machine learning technique which doesn't require labeled data. The clustering has been done for Fashion-MNIST clothing images, which are 28X28 pixels in size. There are 60000 such images are available for training the model and 10000 images are available for validating and testing the cluster. Unsupervised learning is a type of machine learning algorithm used to draw inferences from datasets consisting of input data without labeled responses. The most common unsupervised learning method is cluster analysis, which is used for exploratory data analysis to find hidden patterns or grouping in data. The clusters are modeled using a measure of similarity which is defined upon metrics such as Euclidean or probabilistic distance. In this project we are using two clustering algorithms, viz, KMeans algorithm and Gaussian Mixture Model algorithm.

1 Introduction

Unsupervised learning is a type of self-organized Hebbian learning that helps find previously unknown patterns in data set without pre-existing labels. The only requirement to be called an unsupervised learning method is to learn a new feature space that captures the characteristics of the original space while maximizing some objective function. It is also known as self-organization and allows modeling probability densities of given inputs. It is one of the main three categories of machine learning, along with supervised and reinforcement learning. Semi-supervised learning has also been described, and is a hybridization of supervised and unsupervised techniques. [1] [2]

In this project we need to cluster images and identify it as one of many clusters. The cluster has been fitted using the Fashion-MNIST clothing images. There are three tasks performed in this project:

- Use KMeans algorithm to cluster original data space of Fashion-MNIST dataset using Sklearns library.
- Build an Auto-Encoder based K-Means clustering model to cluster the condensed representation of the unlabeled fashion MNIST dataset using Keras and Sklearns library.
- Build an Auto-Encoder based Gaussian Mixture Model clustering model to cluster the condensed representation of the unlabeled fashion MNIST dataset using Keras and Sklearns library.

1.1 KMeans

The KMeans algorithm clusters data by trying to separate samples in n groups of equal variances, minimizing a criterion known as the inertia or within-cluster sum-of-squares. This algorithm requires the number of clusters to be specified. It scales well to large number of samples and has been used across a large range of application areas in many different fields. The k-means algorithm divides a set of samples into disjoint clusters, each described by the mean of the samples in the cluster. The means are commonly called the cluster centroids; note that they are not, in general, points from set of samples, although they live in the same space. [2]

1.2 Auto Encoder

"Autoencoding" is a data compression algorithm where the compression and decompression functions are data-specific, lossy, and learned automatically from examples rather than engineered by a human. Autoencoder uses compression and decompression functions which are implemented with neural networks as shown below. To build an autoencoder, we need three things: an encoding function, a decoding function, and a distance function between the amount of information loss between the compressed representation of our data and the decompressed representation (i.e. a "loss" function). The encoder and decoder will be chosen to be parametric functions (typically neural networks), and to be differentiable with respect to the distance function, so the parameters of the encoding/decoding functions can be optimized to minimize the reconstruction loss, using Stochastic Gradient Descent. [2] In this project the dimension of image is reduced from 784 (28 x 28) to 196 (4 x 7 x 7).

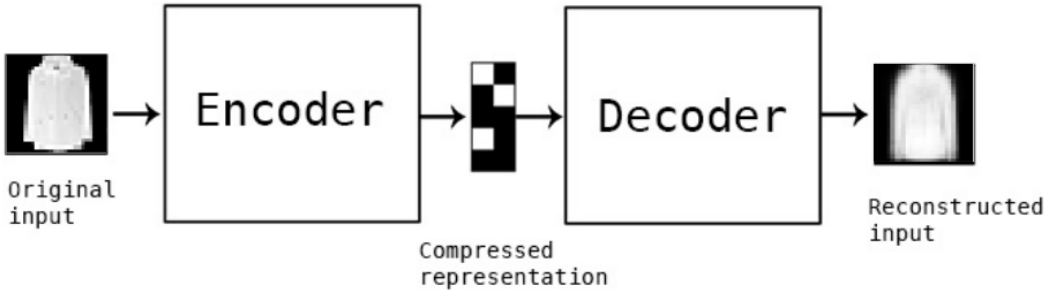

Figure 1. AutoEncoder

1.3 Auto-Encoder with K-Means Clustering

The K-means algorithm aims to choose centroids that minimize the inertia, or within cluster sum of squares criterion:

$$\sum_{i=0}^{n} \min_{\mu_j \epsilon C} \left(\left| \left| x_i - \mu_j \right| \right|^2 \right)$$

Inertia can be recognized as a measure of how internally coherent clusters are. Inertia is not a normalized metric: we just know that lower values are better and zero is optimal. But in very high-dimensional spaces, Euclidean distances tend to become inflated (this is an instance of the so-called curse of dimensionality). Running a dimensionality reduction algorithm such as Principal component analysis (PCA) or Auto-encoder prior to k-means clustering can alleviate this problem and speed up the computations. [2]

1.2 Auto-Encoder with GMM Clustering

A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. One can think of mixture models as generalizing k-means clustering to incorporate information about the covariance structure of the data as well as the centers of the latent Gaussians.

The GaussianMixture object implements the expectation-maximization (EM) algorithm for fitting mixture of Gaussian models. It can also draw confidence ellipsoids for multivariate models, and compute the Bayesian Information Criterion to assess the number of clusters in the data. A GaussianMixture.fit() method is provided that learns a Gaussian Mixture Model from train data. Given test data, it can assign to each sample the Gaussian it mostly probably belongs to using the GaussianMixture.predict() method. [2]

2 Dataset

For training and testing of our classifiers, the Fashion-MNIST dataset has been used. The Fashion-MNIST is a dataset of Zalando's article images, consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes.

Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255. The training and test data sets have 785 columns. The first column consists of the class labels, and represents the article of clothing. The rest of the columns contain the pixel-values of the associated image. [2]

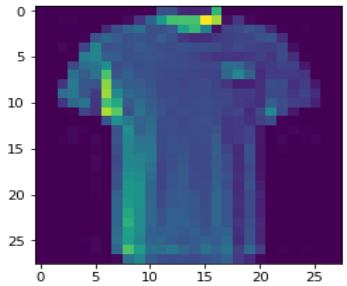

Figure 2. Image from Fashion-MNIST dataset

The labels are:

1 T-shirt/top

2 Trouser

3 Pullover

4 Dress

5 Coat

6 Sandal

7 Shirt

8 Sneaker

9 Bag

10 Ankle Boot

3 Preprocessing

The preprocessing step includes data partitioning, training of the auto encoder using stochastic gradient descent with back propagation algorithm, tuning the hyper-parameters and finally testing the model using the test dataset. And for clustering we need to set the number of clusters, the number of initializations and maximum iterations per initializations.

3.1 Tuning the hyper parameters

Hit and trial method has been used to tune the hyper parameters number of epochs, batch size and number of layers for autoencoder. Validation of the model has been done using the validation dataset and by looking at the loss and accuracy of the model on the training and validation set, the optimal values of the hyper-parameters has been set and then the model has been tested on the testing dataset. For Kmeans and GMM clustering, different values of n-init, max-iter and algorithms are tried.

3.2 Testing the model

For testing the accuracy of the clustering, first we need to map the cluster labels to the original labels as during the formation of clusters the labels are not as per the original labels. The mapping is done by checking the most frequent value for each label at the indices of the original labels, then updating the labels to that value.

The accuracy for the KMeans clustering is 55.99%, for Auto-Encoder with K-Means Clustering is 56.14% and for Auto-Encoder with GMM Clustering is 61.66%.

The accuracy for autoencoder part, i.e 2nd and 3rd could have been much more, but due to restriction of hardware in my laptop the CNN architecture of the autoencoder was taking a lot of time to train and hence it was very difficult to train the model again and again with different architectures and hyper-parameters.

4 Results

Layer (type)	Output Shape	Param #
input_3 (InputLayer)	(None, 28, 28, 1)	0
conv2d_32 (Conv2D)	(None, 28, 28, 32)	320
batch_normalization_31 (Batc	(None, 28, 28, 32)	128
conv2d_33 (Conv2D)	(None, 28, 28, 32)	9248
batch_normalization_32 (Batc	(None, 28, 28, 32)	128
max_pooling2d_7 (MaxPooling2	(None, 14, 14, 32)	0
conv2d_34 (Conv2D)	(None, 14, 14, 64)	18496
batch_normalization_33 (Batc	(None, 14, 14, 64)	256
conv2d_35 (Conv2D)	(None, 14, 14, 64)	36928
batch_normalization_34 (Batc	(None, 14, 14, 64)	256
max_pooling2d_8 (MaxPooling2	(None, 7, 7, 64)	0
conv2d_36 (Conv2D)	(None, 7, 7, 128)	73856
batch_normalization_35 (Batc	(None, 7, 7, 128)	512
conv2d_37 (Conv2D)	(None, 7, 7, 128)	147584
batch_normalization_36 (Batc	(None, 7, 7, 128)	512

```
_________________________________________________________________
conv2d_38 (Conv2D)           (None, 7, 7, 4)           4612
_________________________________________________________________
batch_normalization_37 (Batc (None, 7, 7, 4)           16
_________________________________________________________________
conv2d_39 (Conv2D)           (None, 7, 7, 4)           148
_________________________________________________________________
batch_normalization_38 (Batc (None, 7, 7, 4)           16
_________________________________________________________________
conv2d_40 (Conv2D)           (None, 7, 7, 128)         4736
_________________________________________________________________
batch_normalization_39 (Batc (None, 7, 7, 128)         512
_________________________________________________________________
conv2d_41 (Conv2D)           (None, 7, 7, 128)         147584
_________________________________________________________________
batch_normalization_40 (Batc (None, 7, 7, 128)         512
_________________________________________________________________
conv2d_42 (Conv2D)           (None, 7, 7, 64)          73792
_________________________________________________________________
batch_normalization_41 (Batc (None, 7, 7, 64)          256
_________________________________________________________________
conv2d_43 (Conv2D)           (None, 7, 7, 64)          36928
_________________________________________________________________
batch_normalization_42 (Batc (None, 7, 7, 64)          256
_________________________________________________________________
up_sampling2d_3 (UpSampling2 (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_44 (Conv2D)           (None, 14, 14, 32)        18464
_________________________________________________________________
batch_normalization_43 (Batc (None, 14, 14, 32)        128
_________________________________________________________________
conv2d_45 (Conv2D)           (None, 14, 14, 32)        9248
_________________________________________________________________
batch_normalization_44 (Batc (None, 14, 14, 32)        128
_________________________________________________________________
up_sampling2d_4 (UpSampling2 (None, 28, 28, 32)        0
_________________________________________________________________
conv2d_46 (Conv2D)           (None, 28, 28, 1)         289
=================================================================
Total params: 585,849
Trainable params: 584,041
Non-trainable params: 1,808
_________________________________________________________________
```

Figure 3. AutoEncoder Model Summary

	Accuracy	**Normalized mutual info score**	**Homogeneity**	**Completeness**	**V measure**
Kmeans	55.99%	0.512670636	0.500965541	0.524649221	0.512533927
AutoEncoder + Kmeans	56.14%	0.525708039	0.506411928	0.545739401	0.52534067
AutoEncoder + GMM	61.66%	0.593247667	0.588644764	0.597886563	0.593229672

Table 1. Accuracy and different scores for the 3 parts

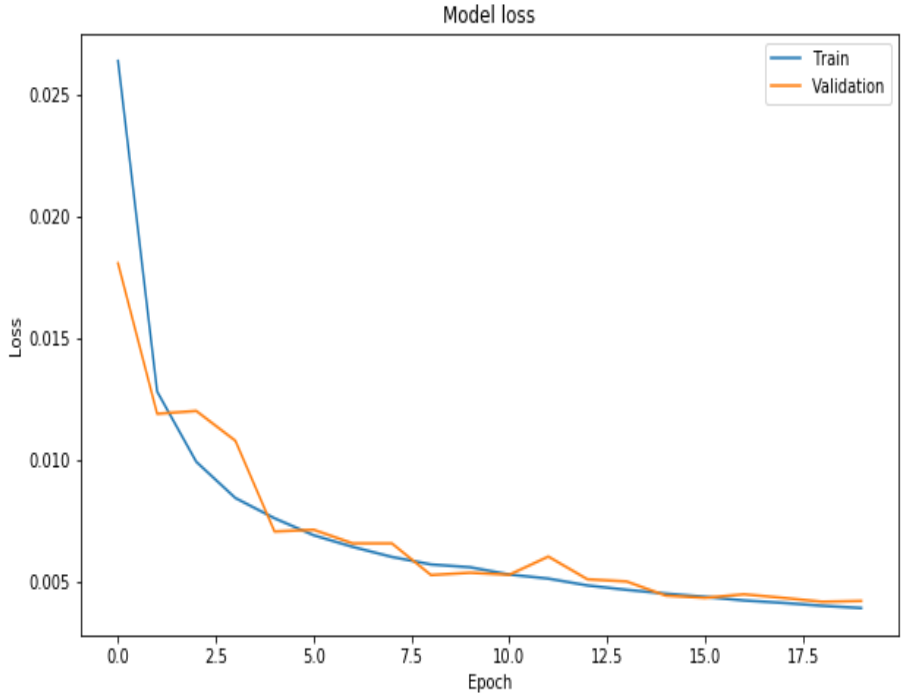

Figure 4. Loss vs epochs for AutoEncoder

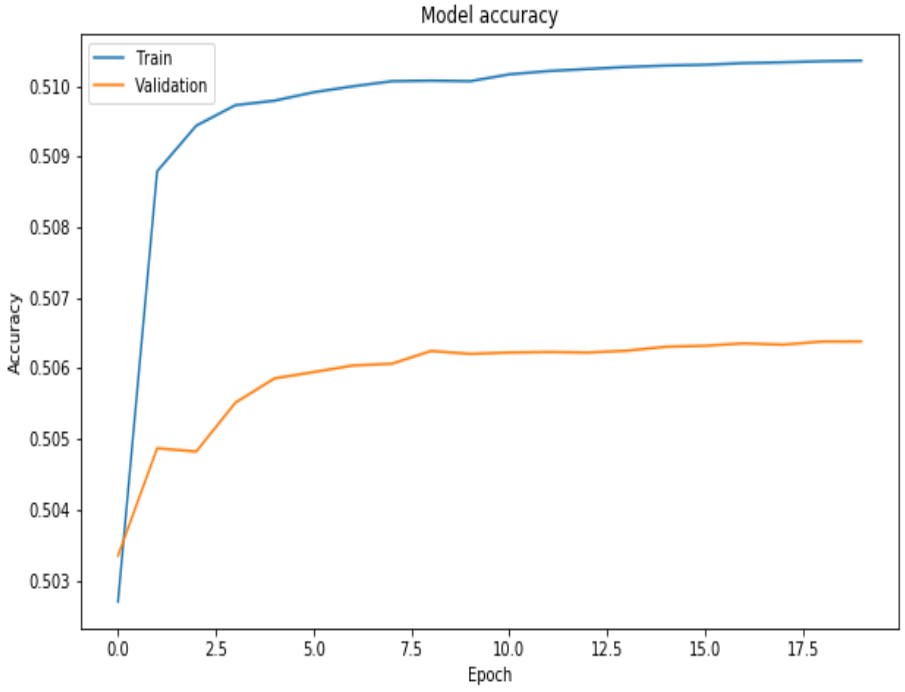

Figure 5. Accuracy vs epochs for AutoEncoder

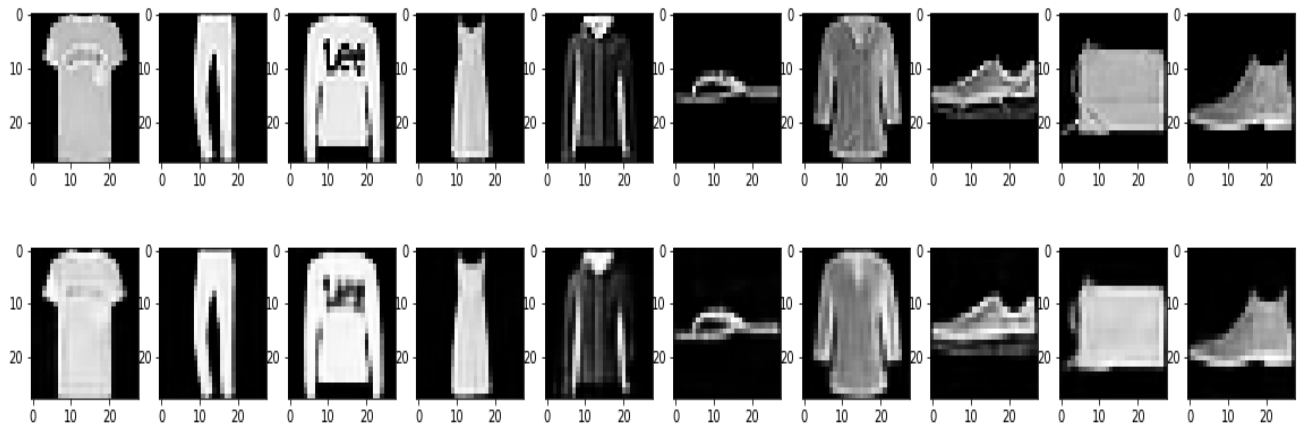

Figure 6. AutoEncoder input(up) vs output(down)

Confusion Matrix for KMeans clustering:

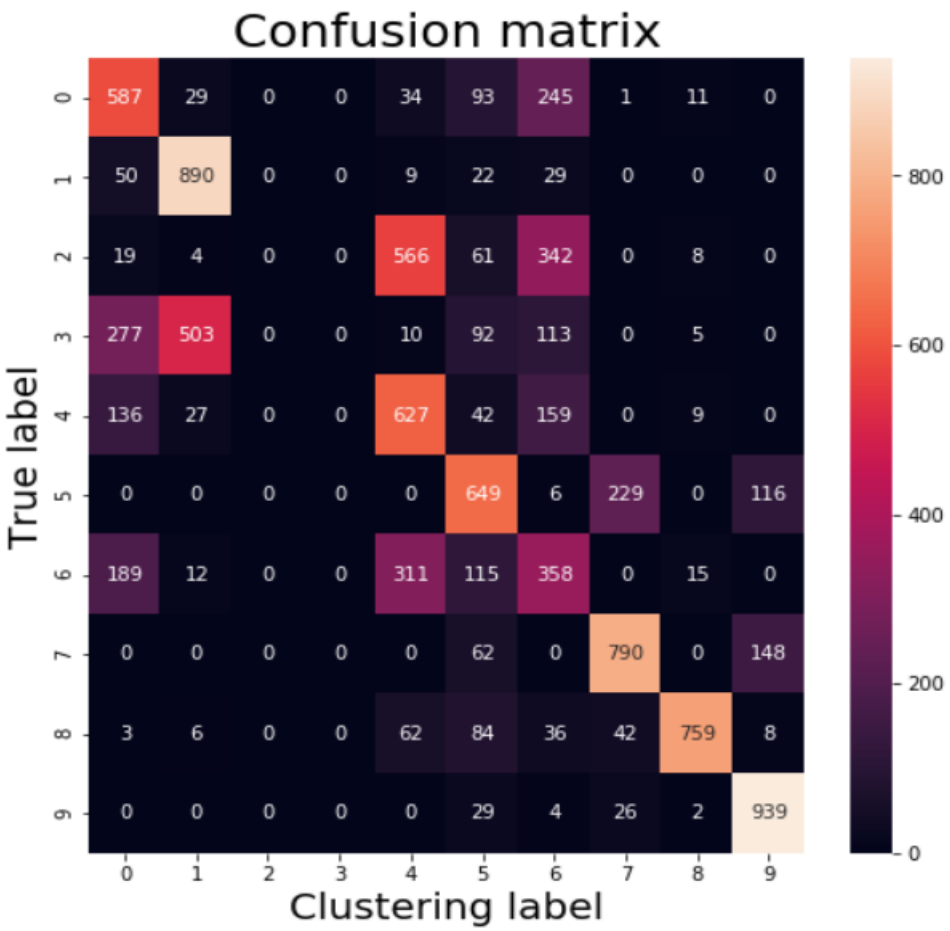

Figure 7. Confusion matrix for KMeans clustering

Confusion Matrix for Auto-Encoder with KMeans clustering:

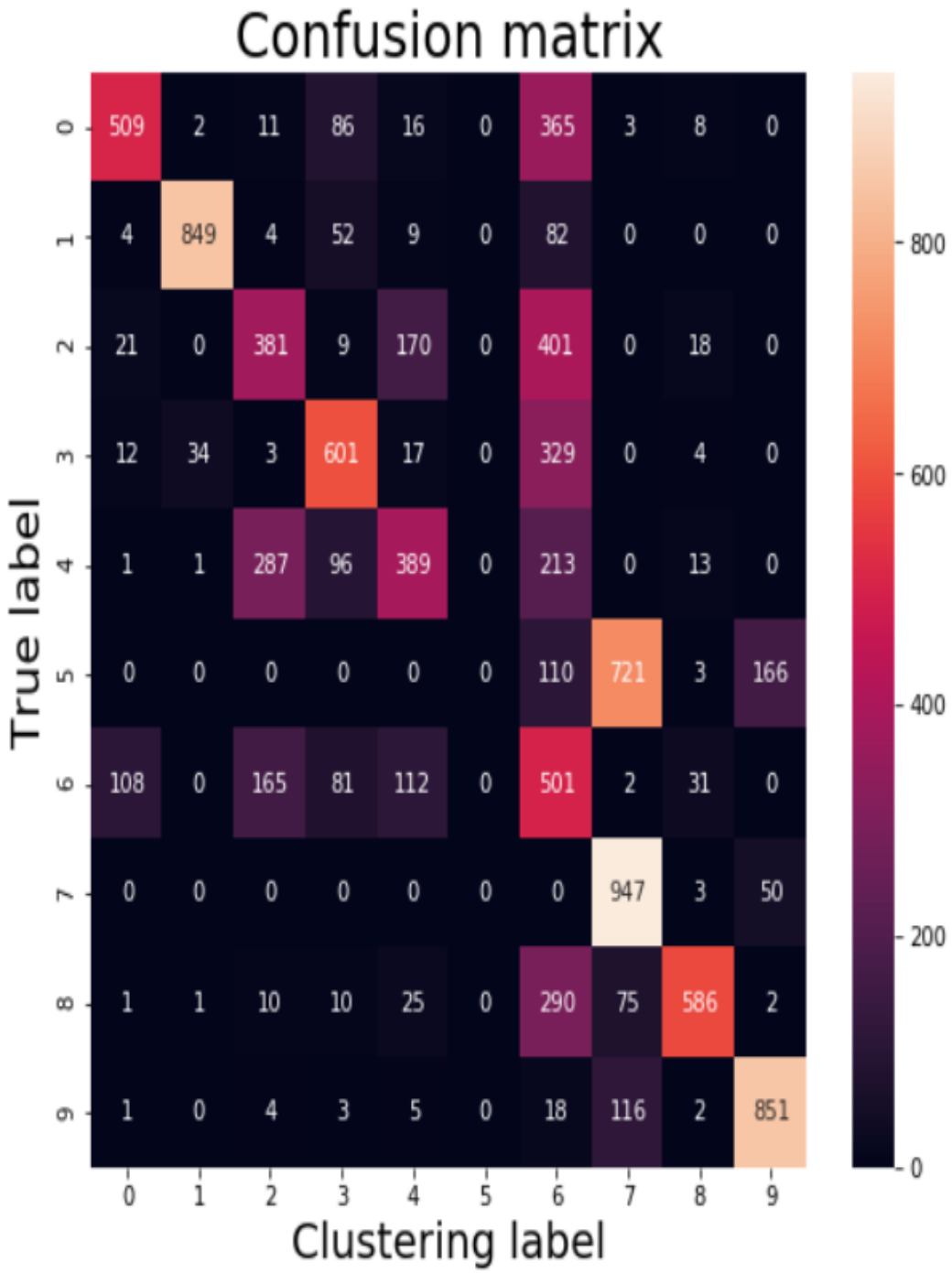

Figure 8. Confusion matrix for AutoEncoder with KMeans clustering

Confusion Matrix for Auto-Encoder with GMM clustering:

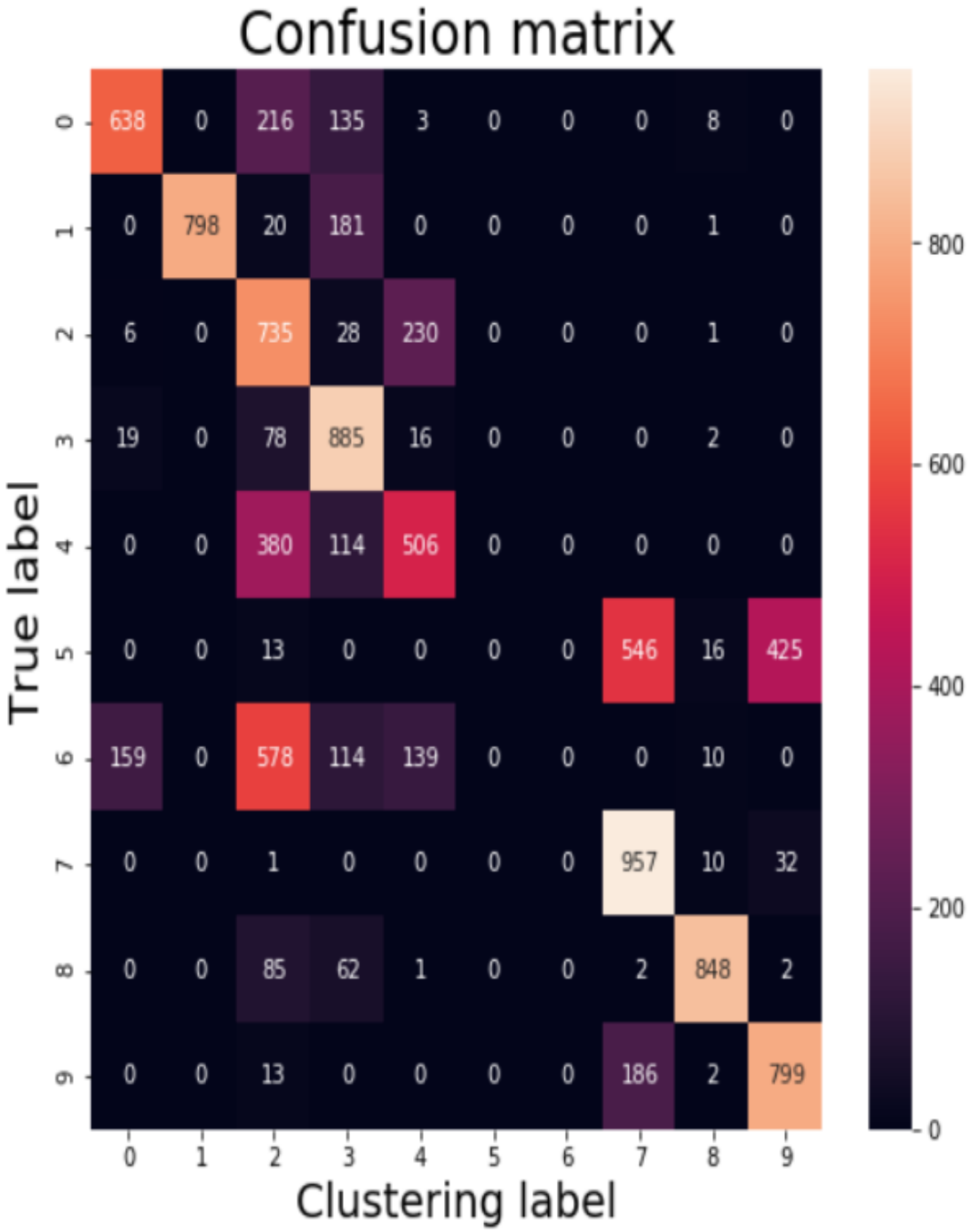

Figure 9. Confusion matrix for AutoEncoder with GMM clustering

5 Conclusion

In this project, at first, KMeans clustering was done using the sklearns library. Then an Auto Encoder was built using convolutional layers to decrease the dimensionality of the input image and then feed the reduced dimensions to KMeans clustering and GMM clustering.

By looking at the accuracy, it can be said that Auto Encoder with GMM clustering has the best accuracy, then Auto Encoder with KMeans clustering and KMeans clustering with original image dimensions has the lowest accuracy.

References

[1] Project description PDF.

[2] Wikipedia: https://en.wikipedia.org/wiki/Unsupervised_learning