Unsupervised Clustering Using Deep Learning

Ananta R. Bhattarai
Carnegie Mellon University
5530 Wilkins Ave, Pittsburgh, PA
`abhattar@andrew.cmu.edu`

Abstract

Clustering is a machine learning technique where we group similar data points. Unsupervised clustering is one of the challenges in the field of machine learning. It is difficult to learn structure within a dataset due to the high dimensionality of the data. Several techniques such as k-means or Gaussian mixture models are being used to group the data into clusters but, their performance is not promising when we have a large dataset with a high dimension. In this paper, we study several methods to improve clustering performance using deep learning architectures. At first, we use auto-encoder to reduce the big data into latent space and run different clustering techniques on obtained latent space.

1 Introduction

Clustering is an unsupervised learning technique that aims to classify data into several classes without label information [1]. With the increasing use of technology, all the data are processed using computers to form high-dimensional, large-scale unlabelled datasets in today's world. Examples of such domains include robotics, computational geometry, computer vision, geographic information system, and many more. As a result of large scale and dimension, it is challenging to analyze, process, and learn internal structures within the data using traditional clustering techniques. The clustering techniques such as k-means or Gaussian mixture model is not quite promising to cluster high dimension data. Different deep learning architectures such as auto-encoders and sparse coding have been proposed to avoid the curse of dimensionality and work with unlabelled datasets. The architectures, like auto-encoder and sparse coding automatically extract abstract features from the data and outperform manually crafted features in terms of intricate learning patterns within the data [2, 3]. As deep learning architectures show excellent performance with high scale datasets, we study various deep learning architectures combined with traditional clustering techniques to achieve high accuracy and performance. We will evaluate our model on MINST [4] dataset, which consists of 70000 handwritten digits of 28-by-28 pixel size and 10 classes. Also, to verify our model's robustness, we will test our model on CIFAR [5] dataset, which consists of 60000 32-by-32 color images in 10 classes, with 6000 images per class.

2 Background

As clustering in a higher dimension can be difficult, we plan to use auto-encoder to reduce the higher dimension data into latent space and use traditional clustering techniques to cluster data points from the latent space. We proposed a deep auto-encoder (AE) [6] combined with a Gaussian mixture model (GMM) [7] as our baseline model. Our baseline results demonstrate that dimension reduction improves clustering accuracy almost by a factor of 2. Our model learns a good representation of features in the latent space and outperforms traditional clustering techniques on both train and test datasets. The results are already close to state-of-the-art clustering methods. However, there are several other deep learning architectures which perform well on images such as convolution neural

1 Final Project Report (Carnegie Mellon University 2019), Pittsburgh.

network. We take inspiration from our baseline results and plan to investigate convolution auto-encoder architecture combined with traditional clustering methods such as k-means and GMM to improve our baseline model. We will research different sets of hyperparameters for the network and see which one performs best. The robustness of our model should also be evaluated. To accomplish this, we have included the CIFAR dataset. We hope to set a benchmark for clustering analysis.

3 Related work

Clustering has been extensively studied in the field of machine learning. In 2016, Xie [8] proposed Deep Embedded Clustering (DEC) method, which learns feature representation and cluster assignments using deep neural networks. In 2017, Jiang and Tian [9] proposed the Variational Deep Embedding technique that clusters data using generative data procedure with a Gaussian Mixture Model and a deep neural network. In 2018, Liu and Zhu [10] proposed Deep Convolutional Embedded Clustering (DCEC) algorithm to cluster images automatically. All of the techniques combine the auto-encoder model with traditional clustering to achieve high accuracy. We gain inspiration from the previous works on unsupervised clustering using deep learning as well as our baseline model and research various other architectures and algorithms to improve performance.

4 Methods

4.1 Stacked dense auto-encoder

An auto-encoder helps to reduce the dimension of the input and extract abstract features preserving the internal structure and information. An auto-encoder consists of encoder and decoder. Encoder maps the input to latent representation. Mathematically,

$$h(x) = g(\mathbf{a(x)}) \tag{1}$$

$$h(x) = g(\mathbf{b + Wx}) \tag{2}$$

where h(x) is a latent representation, g is activation, **b** is the bias term, **W** is the weight matrix and **x** is the input. A decoder constructs the original input from the latent space. Mathematically,

$$x' = o(\mathbf{a'(x)}) \tag{3}$$

$$x' = o(\mathbf{c + W^*h(x)}) \tag{4}$$

where x' is the reconstructed input, o is activation, **c** is bias term and **W*** is the weight matrix. An auto-encoder is trained in a similar fashion as neural network using back propagation. We minimize mean squared error during training. Specifically, we minimize:

$$l(f(x)) = \frac{1}{2}\sum_{k}(x'_k - x_k) \tag{5}$$

where x' is the reconstructed input, and x is the original input.

Our baseline model uses a stacked dense auto-encoder to transform the dataset into latent space. A stacked auto-encoder consists of several auto-encoders stacked and combined. Our encoder consists of one input layer, three hidden layers, and one latent layer. Similarly, the decoder consists of three hidden layers and one output layer. There is ReLU non-linearity in between the layers except for the underlying layer and output layer. The architecture and the dimensions of the network are shown in Figure 1. We train our stacked auto-encoder in the same way stated above using the backpropagation algorithm. The network is trained until we get a good representation of input.

4.2 Convolutional auto-encoder

A convolutional auto-encoder is composed of encoder and decoder, the same as dense auto-encoder. The main objective of the convolutional auto-encoder is to reconstruct the input minimizing mean

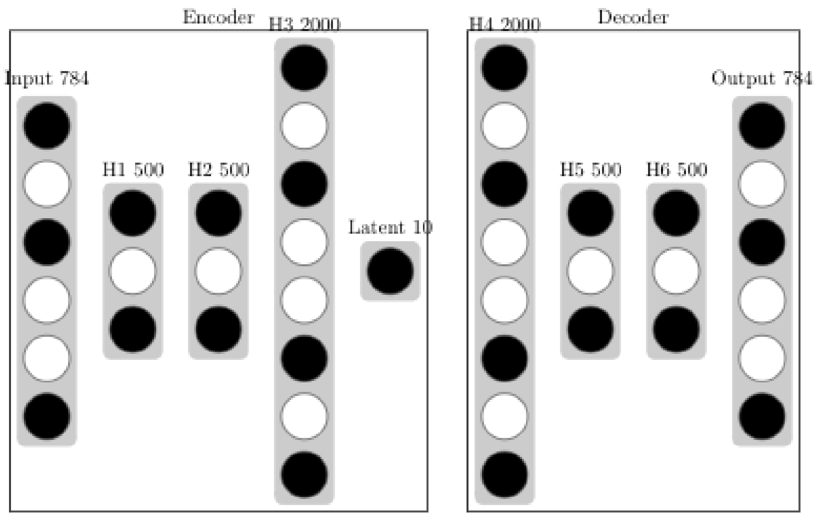

Figure 1: Architecture of a dense auto-encoder for MINST dataset

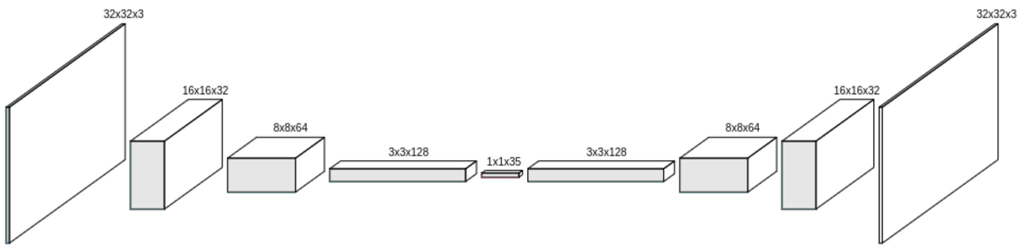

Figure 2: Architecture of a convolutional auto-encoder for CIFAR dataset

squared error. First, our encoder has 3 stacked convolutional layers to extract hierarchical features. Then, we flatten the last convolutional layer followed by one fully connected layer with 10 units, which is our latent space. Similarly, our decoder has a similar architecture to construct original input from the latent space. The architecture is shown in Figure 2. We train the network using a backpropagation algorithm and update the weight parameters minimizing equation 5. There is ReLU activation in between the layers except for the underlying layer and output layer. We use convolutional stride instead of the pooling layer in both encoder and decoder because it allows the network to learn spacial subsampling from the data. The key factor of convolutional layers is that they preserve the local structure of the image and have successfully learned about complex patterns and structures within the image compared to dense layers.

4.3 Gaussian Mixture Model

GMM is an unsupervised clustering technique that uses a probabilistic model and assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. It makes the use of the expectation-maximization (EM) algorithm for fitting the mixture of Gaussian models. For a mixture model, it is assumed that a given sample **x** is the realization of a random vector which distribution is a mixture (convex combination) of conditioned distributions [11]:

$$P(\mathbf{x}|\theta) = \sum_{k=1}^{K} \alpha_k \phi(\mathbf{x}|\theta_k) \tag{6}$$

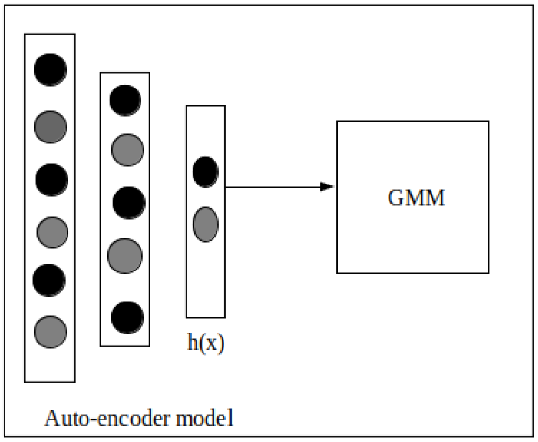

Figure 3: Architecture of an auto-encoder with GMM

where **x** is the features, K is the number of clusters, α_k is the probability that the kth class in the sample set is selected and $\phi(\mathbf{x}|\theta_k)$ represents the Gaussian distribution density of the kth Gaussian model. We can represent it mathematically as

$$\phi(\mathbf{x}|\theta_k) = \frac{1}{\sqrt{2\pi}\sigma_k} exp(-\frac{(\mathbf{x} - u_k)^2}{2\sigma_k^2}) \tag{7}$$

where u_k is the mean, σ_k^2 is the variance. We obtain the complete parameterized GMM by averaging the mean, the variance, and the combined weight of all component models. We can express the parameters as

$$\theta = u_k, \delta_k^2, \alpha_k, (k = 1, 2, 3...K) \tag{8}$$

Once we have the initial GMM configuration and training data, the expectation-maximization algorithm is used to iteratively estimate the parameters of the Gaussian mixture model so that the trained eigenvectors are closely distributed.

4.4 Auto-encoder with GMM

As it is challenging to cluster in the higher dimension, we use an auto-encoder to reduce the input image into the latent dimension. We use a convolutional auto-encoder and dense auto-encoder to reduce the dimension and analyze which model performs best with our dataset. After we get a good feature representation of the input in the latent space, we use GMM to cluster them. The structure of an auto-encoder with GMM is shown in Figure 3. Then, the expectation-maximization algorithm is run for 100 iterations to estimate the model parameters. After the model parameters are set, we can predict the most probable label for the input data points. In other words, we can predict the class label to which input belongs.

5 Results

5.1 Dataset

We select two image datasets to evaluate our model. The information on the datasets is given in Table 1.

Table 1: Dataset description

Dataset	Total samples	Original Dimension	Latent Dimension
MINST	70,000	784	10
CIFAR	60,000	3072	35

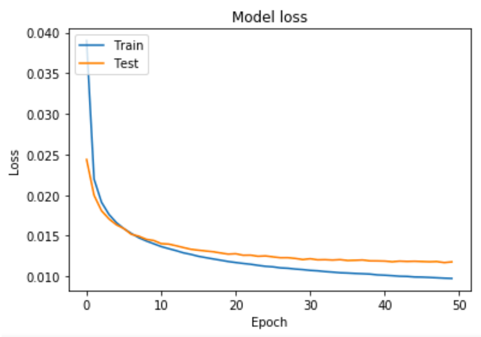

Figure 4: Loss of a dense auto-encoder

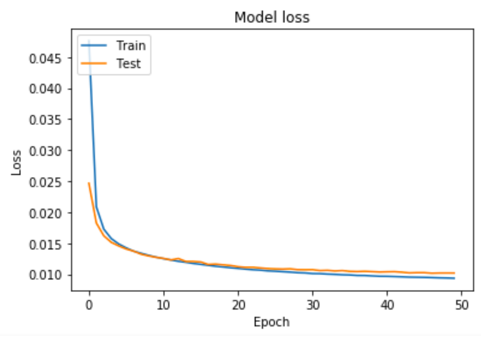

Figure 5: Loss of a convolutional auto-encoder

5.2 Evaluation Metric

To evaluate the accuracy of our model, We use the v-measure [12] score. It is a good evaluation measure because a permutation of the class or cluster label values will not change the score value in any way. So, we compare the v-measure score obtained using our model with state-of-the-art and traditional clustering methods.

5.3 Parameters

Our dense encoder has architecture x-500-500-2000-d, and the decoder has architecture 2000-500-500-x. We set x, d to 784, 10 respectively for MINST dataset and 3072, 35 for the CIFAR dataset. The convolutional encoder architecture for the CIFAR dataset is shown in Figure 2. In the same way as dense auto-encoder, we set the input and output dimension to be 28*28*1 and latent dimension to 10 for the MINST dataset. We train the auto-encoder for 50 epochs to get a good representation of the input. After that, we run the GMM algorithm on the latent space, as described in the Methods section above.

5.4 Evaluation

The loss of an auto-encoder is shown in Figures 4, 5, 6, and 7. Figures 4 and 5 show the loss of a dense and convolutional auto-encoder on the MINST dataset, respectively, whereas Figures 6 and 7 show the loss on the CIFAR dataset. As we can see, the test loss of a convolutional auto-encoder is

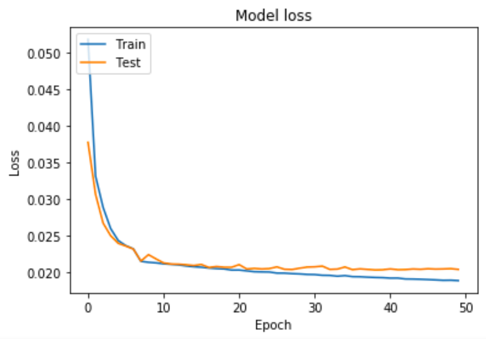

Figure 6: Loss of a dense auto-encoder

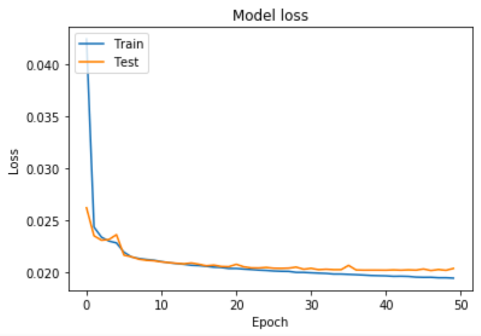

Figure 7: Loss of a convolutional auto-encoder

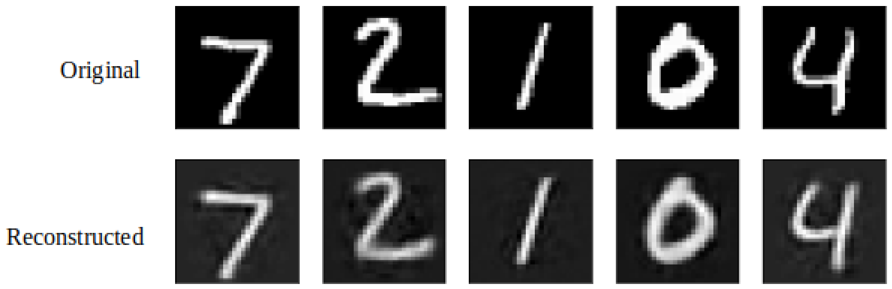

Figure 8: Comparision of orginal and reconstructed input using dense auto-encoder

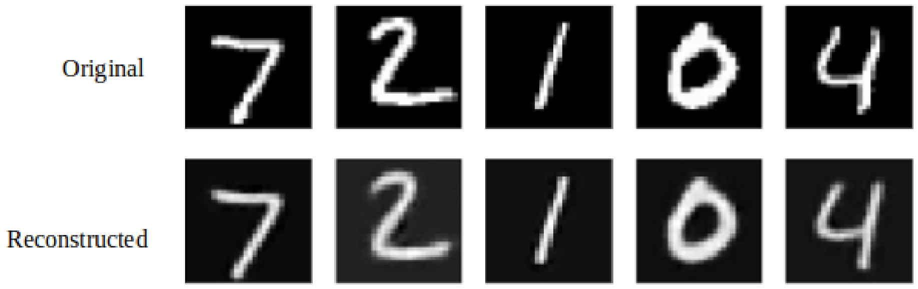

Figure 9: Comparision of orginal and reconstructed input using convolutional auto-encoder

lower at the beginning, showing that convolutional auto-encoder is better at extracting features for a small number of epochs. The test loss curve of a dense auto-encoder starts to diverge from the train loss curve after a few numbers of epochs. This concludes that dense auto-encoder starts to overfit, as we increase the number of epochs. The convolutional auto-encoder outperforms dense auto-encoder in extracting hierarchical features from the image. We get a good representation of an image in the latent space using a convolutional auto-encoder.

We also compare the reconstructed and original input using both the auto-encoders model. Figure 8 shows the original data and reconstructed data using dense auto-encoder, and Figure 9 shows the original input and reconstructed input using a convolutional auto-encoder. There is an absolute difference between two of the input, but, we can see that the information is preserved and well reconstructed. It can be seen on the diagram that convolutional auto-encoder reconstructs the input well than dense auto-encoder. The reconstructed input using dense auto-encoder is noisy compared to convolutional auto-encoder reconstructed input. Our experiments suggest that for a more complex dataset like CIFAR, convolutional auto-encoder has a good representation of features in latent space compared to dense auto-encoder.

We visualize the t-SNE [13] embedding of the latent space for a different number of epochs to select the best hyperparameter. Figure 10 shows the embedding of latent space for the first epoch. We can see that the data points overlap with each other at the beginning. However, as we increase the number of epochs, data points get separated from each other and form clusters. Figures 11, 12, and 13 show the t-SNE embedding of latent space for an increasing number of epochs. The diagrams suggest that after 50 epochs, data points are well separated from each other to form clusters. We also compare the t-SNE embedding for both auto-encoder models. Figure 13 shows the t-SNE embedding of dense auto-encoder after 50 epochs, and Figure 14 shows the embedding of convolutional auto-encoder after 50 epochs. Both the figure looks similar; however, the boundary between the clusters using convolutional auto-encoder is significant compared to dense auto-encoder. This suggests that the convolutional auto-encoder outperforms dense auto-encoder in image clustering tasks.

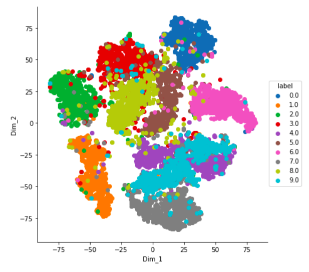
Figure 10: TSNE visualization of MINST data for 1 epoch using dense auto-encoder

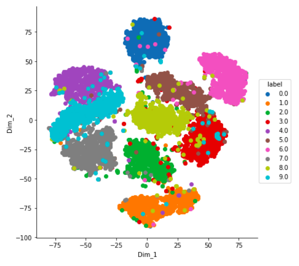
Figure 11: TSNE visualization of MINST data for 5 epochs using dense auto-encoder

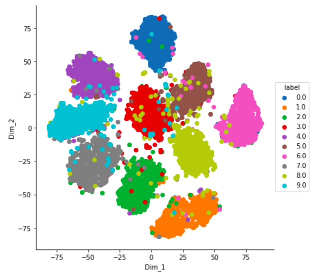
Figure 12: TSNE visualization of MINST data for 20 epochs using dense auto-encoder

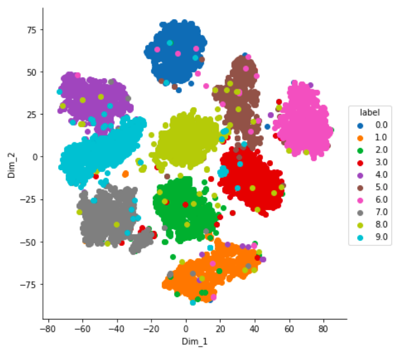
Figure 13: TSNE visualization of MINST data for 50 epochs using dense auto-encoder

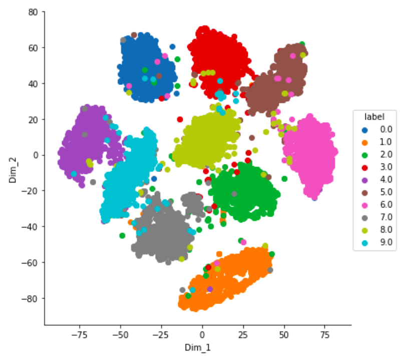
Figure 14: TSNE visualization of MINST data for 50 epochs using convolutional auto-encoder

We also try to determine an optimal number of clusters. To carry out this task, we obtain the v-score for different numbers of clusters initialization. The evaluation is shown in Figure 15. We can see that the v-score increases significantly until we have 10 clusters. After that, there is a sharp drop in generalizability, which suggests that there are 10 clusters in the dataset. This is true as we have 10 classes in the images.

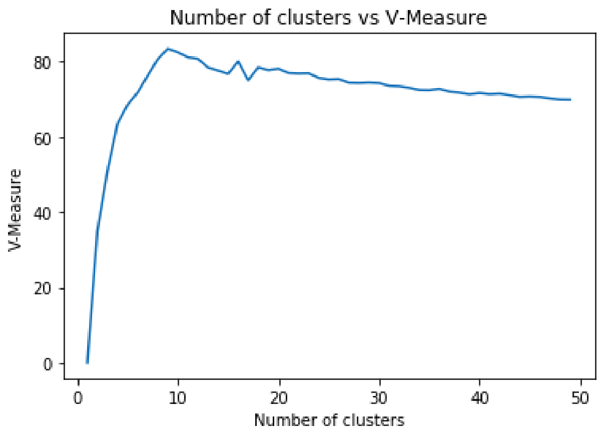

Figure 15: Number of clusters vs v-measure for MINST

Table 2: Comparison of clustering accuracy on MINST dataset

Method	Accuracy (v-measure)	
	Train	Test
GMM	32.5%	50.7%
k-means	49.0%	52.0%
AE + k-means	74.8%	76.7%
AE + GMM	81.7%	82.6%
conv AE + k-means	75.7%	78.4%
conv AE + GMM	**84.1%**	**88.7%**

Table 3: Comparison of clustering accuracy on CIFAR dataset

Method	Accuracy (v-measure)	
	Train	Test
GMM	8.1%	8.5%
k-means	7.8%	8.4%
AE + k-means	10.1%	10.9%
AE + GMM	13.8%	15.0%
conv AE + k-means	9.9%	10.5%
conv AE + GMM	**15.7%**	**16.2%**

Table 2 and 3 describes the comparison of clustering accuracy for different algorithms and our proposed convolutional auto-encoder with GMM algorithm on two datasets. It can be seen that our algorithm improves clustering accuracy almost by a factor of 2. Our algorithm outperforms all other algorithms on both datasets. Our results also suggest that on a more complex dataset like CIFAR, convolutional auto-encoder with GMM increases the accuracy by a considerable margin.

6 Discussion and Analysis

Our proposed algorithm maps the input into the latent dimension using a convolutional auto-encoder and clusters the latent space using GMM. Our results demonstrate that dimension reduction using an auto-encoder helps in improving clustering accuracy. The results are already close to state-of-the-art clustering methods. The convolutional auto-encoder outperforms dense auto-encoder in image clustering tasks of complex datasets. However, for datasets like CIFAR, which has sophisticated features and patterns, it is challenging to perform clustering. This is the active field of research in deep learning to try different other algorithms and architectures, such as variational auto-encoder and generative adversarial networks. In the future, we will not be limited to proposed architecture; instead, we will explore some other algorithms which will improve the clustering performance and accuracy. We hope to set a milestone in the image clustering task.

References

[1] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos & Prabhakar Raghavan. *Automatic subspace clustering of high dimensional data for data mining applications*, volume 27. ACM, 1998.

[2] Yi-Ou L, Hang L & Xiao-Yu L I et al 2017 Deep learning in NLP: methods and applications *Journal of University of Electronic Science & Technology of China* **46(6)** pp 913-19.

[3] Gheisari M, Wang G & Bhuiyan M Z A 2017 A survey on deep learning in big data *IEEEInternational Conference on Computational Science & Engineering*.

[4] LeCun, Yann, Bottou, Leon, Bengio, Yoshua & Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278– 2324, 1998.

[5] Krizhevsky, Alex (2009). "Learning Multiple Layers of Features from Tiny Images.

[6] Hinton, Geoffrey E & Salakhutdinov, Ruslan R. Reducing the dimensionality of data with neural networks. Science, 313(5786):504–507, 2006.

[7] Bishop, Christopher M. *Pattern recognition and machine learning*. Springer New York, 2006.

[8] Xie, J., Girshick, R.B. & Farhadi, A. (2015). Unsupervised Deep Embedding for Clustering Analysis. *ICML*.

[9] Jiang, Zhuxi & Zheng, Yin & Tan, Huachun & Tang, Bangsheng & Zhou, Hanning. (2017). Variational Deep Embedding: An Unsupervised and Generative Approach to Clustering. 1965-1972. 10.24963/ijcai.2017/273.

[10] Guo, Xifeng & Liu, Xinwang & Zhu, En & Yin, Jianping. (2017). Deep Clustering with Convolutional Autoencoders. 373-382. 10.1007/978-3-319-70096-0_39.

[11] Dilokthanakul N, Mediano P A M and Garnelo M et al 2016 Deep unsupervised clustering with gaussian mixture variational autoencoders J

[12] Andrew Rosenberg & Julia Hirschberg, 2007. V-Measure: A conditional entropy-based external cluster evaluation measure

[13] van der Maaten, L.J.P.; Hinton, G.E. Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research 9:2579-2605, 2008.