Performance analysis of neural network topologies and hyperparameters for deep clustering

Muhammed Kucuk
Department of Electrical Engineering
University of South Florida
Tampa, USA
mkucuk@mail.usf.edu

Ismail Uysal
Department of Electrical Engineering
University of South Florida
Tampa, USA
iuysal@usf.edu

***Abstract*—Deep learning found its initial footing in supervised applications such as image and voice recognition successes of which were followed by deep generative models across similar domains. In recent years, researchers have proposed creative learning representations to utilize the unparalleled generalization capabilities of such structures for unsupervised applications commonly called deep clustering. This paper presents a comprehensive analysis of popular deep clustering architectures including deep autoencoders and convolutional autoencoders to study how network topology, hyperparameters and clustering coefficients impact accuracy. Three popular benchmark datasets are used including MNIST, CIFAR10 and SVHN to ensure data independent results. In total, 20 different pairings of topologies and clustering coefficients are used for both the standard and convolutional autoencoder architectures across three different datasets for a joint analysis of 120 unique combinations with sufficient repetitive testing for statistical significance. The results suggest that there is a general optimum when it comes to choosing the coding layer (latent dimension) size which is correlated to an extent with the complexity of the dataset. Moreover, for image datasets, when color makes a meaningful contribution to the identity of the observation, it also helps improve the subsequent deep clustering performance.**

***Index Terms*—Deep clustering, neural networks, k-means, autoencoder, convolutional autoencoder**

I. Introduction

Clustering [1], the unsupervised process that groups similar data examples together based on some distance measures, is one of the primary problems in various research fields, such as machine learning, computer vision, pattern recognition and, data analysis. Many clustering methods have been proposed including k-means [2], [3], [4] and Gaussian Mixture Models (GMM) [5]- [6], however, traditional clustering methods do not perform well with high-dimensional data, due to the inadequacy of distance measures applied in these methods. Besides, these clustering methods are affected by high computational complexity on large datasets. Therefore, dimensionality reduction and feature mapping methods have been studied extensively to represent the original data in a feature (latent) space where original data is separated more effectively by a clustering algorithm. However, the complexity of the latent space still remains a challenging problem. Recent progress in deep learning [7], led to deep neural networks (DNN) being used as non-linear and rich mappings of the data input space into a lower dimensional feature space. In other words, DNNs integrate representation learning with clustering using raw data with a high accuracy rate. This new method of grouping is generally referred to as Deep Clustering (DC).

Researchers have previously considered feature mapping and data grouping (clustering) as two different processes. First, high dimensional input examples are transformed into a generally lower dimensional feature space. Then, the clustering algorithm is applied to the transformed data. DC on the other hand aims to combine these two processes as first introduced with the Deep Embedding Clustering (DEC) [8] which implements feature mapping via a fully connected deep autoencoder [9] with a k-means back-end for clustering. Variations of DEC have been proposed in recent years including, the Discriminatively Boosted Clustering (DBC) which replaces the feature mapping auto-encoder with a convolutional autoencoder (CAE) for image analysis [10], a joint dimensionality reduction technique with k-means based on DNN [11], the Deep Embedded Regularized Clustering (DEPICT) using logistic regression with CAE for joint clustering assignment [12], the Variational Deep Embedding (VaDE) based on a variational auto-encoder (VAE) and Gaussian Mixture Model (GMM) [13],the Joint Unsupervised Learning (JULE) proposed as a recurrent perspective with convolutional neural network (CNN) activated data on agglomerative clustering [14], and a CNN-based joint clustering method which brings an iterative solution with feature drift compensation [15]. While deep clustering remains a popular research field with such recent advances in algorithm design and clustering accuracy [16], the process of choosing many of the hyper - parameters, such as the code size, network topology and clustering coefficient, still remains an inexact science.

The purpose of this analysis paper is to address this gap in our knowledge of deep clustering methodologies and conduct a comprehensive analysis study on how DC hyperparameters affect the clustering performances of deep embedding networks. The DNNs used in this study are auto-encoders (AE) [17] and convolutional auto-encoders (CAE) [18] applied to different image datasets with varying complexities. For a clear perspective, we choose popular image benchmark datasets MNIST, CIFAR10, and SVHN for a two-phase experiment where each phase implements a change in either topology or hyper-parameter set. In summary this paper has the following contributions:

1) When using deep unsupervised feature extraction, more complex datasets require a higher dimensional latent space to achieve the best subsequent clustering performance.
2) Color information can be useful in statistically significantly improving the deep clustering performance for datasets where color makes a meaningful contribution to the identity of the observation.
3) General trends need to be further evaluated on a wider variety of datasets and clustering domains to make more definitive conclusions.

The rest of this paper is organized as follows: Section II provides general information about the Autoencoder and the Convolutional Autoencoder structures and their applications to deep clustering. Section III describes the experimental setup used in the study and along with the hyperparameters, the datasets, evaluation metrics, and implementation. The results and discussions are presented in section IV whereas section V concludes the paper.

II. METHODS

A. Autoencoder

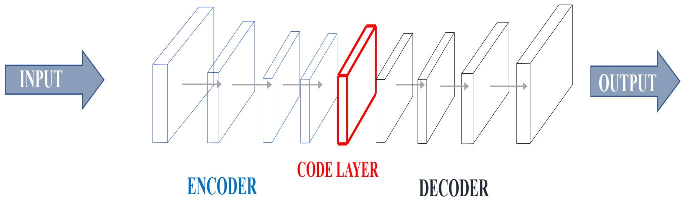

Fig. 1. Autoencoder Network Structure

Autoencoder is a particular artificial neural network topology which has the same input and output layers where the training is performed by presenting the same input data to both layers simultaneously. The general structure of the auto-encoder consists of a visible input layer $\mathbf{x}$, a number of hidden layers $\mathbf{h}$ and the reconstructed output layer $\mathbf{y}$ with a family of nonlinear activation functions $\mathbf{f}$ applied at different layers.

During training, the auto-encoder maps the input $\mathbf{x} \varepsilon \mathbf{R^y}$ to the hidden layers with lesser dimensions than the input data which produces a compressed representation of the original data in which its dimensionality is reduced to the code(latent) layer size $\mathbf{H} \varepsilon \mathbf{R^h}$. This first step is called the "encoder" and is shown on the left side of Fig.1. Later, the compressed information is mapped to the output layer via the "decoder," through a process called "reconstruction." Mathematically, these two steps are formulated as follows:

$$H \equiv f_{WH}(x) = f(W_H x + b_H) \tag{1}$$

$$z \equiv g_{Wz}(x) = g(W_z H + b_Z)$$

Where W_H and W_Z define the encoding weight and decoding weight, respectively, b_H and b_Z define the corresponding encoding bias vector and the decoding bias vector, and $f(.)$ and $g(.)$ are encoding and decoding activation functions such as a sigmoid function or a rectified linear unit, respectively. As previously mentioned, the primary purpose of the auto-encoder is to learn useful latent information on the code layer by minimizing the reconstruction error. For a given N input data samples, the following loss function is used to determine the parameters "$W_H, W_Z, b_H,$ and b_Z" through a back-propagation algorithm commonly used in feed-forward neural networks:

$$L_{AE} = min \frac{1}{N} \sum_{k=1}^{N} ||x_k - z_k||^2 \tag{2}$$

In this paper, we constructed several auto-encoder networks with different topologies and four different code layer sizes to simulate a variety of scenarios and study the impact of topology on reconstruction and clustering performance.

B. Convolutional Autoencoder (CAE)

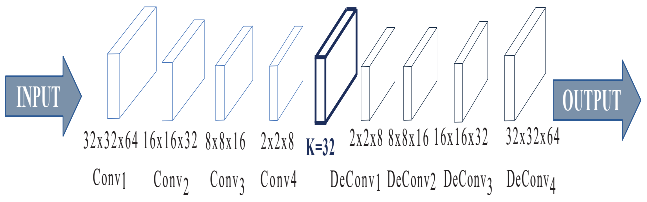

Fig. 2. Convolutional Autoencoder Network Structure

The convolutional auto-encoder (CAE) is similar to the standard auto-encoder except the input layers are replaced with convolutional layers to present a powerful technique specifically for image-processing tasks. CAEs borrow ideas from the Convolutional Neural Networks (CNN) much like how AEs implement standard fully - connected networks. Similar to the equations defined in section II-A, we define the CAE encoding part as follows where multiplications are replaced with 2D convolutions:

$$H \equiv f(W_H * x) \tag{4}$$

$$z \equiv g_{DZ}(H) = g(H * D_Z) \tag{5}$$

where H represents the input image samples as the latent variables in the code layer which then feeds into the fully connected AE hidden layers, W_H and W_Z are encoding and decoding weights, '*' is the 2D convolution operation. The CAE's primary purpose is finding the latent layer representation, sometimes called the coding layer, through minimizing a cost function such as the mean squared error (MSE) between original and reconstructed images where the corresponding loss function is defined as:

$$L_{CAE} = \min_{W_H, D_Z} \frac{1}{N} \sum_{j=1}^{N} ||g_{Dz}(f_{WH}(x_j)) - x_j||^2 \tag{6}$$

where N is the number of input images in the dataset, $x_j \varepsilon R^2$ is the j^{th} image.

As shown in Fig.2, each convolutional layer at the encoder includes filters with a certain size and stride, image normalization followed by max pooling to transform and compress the information included in the original image. The decoder structure is similar but in reverse order which includes up sampling to obtain the reconstructed image at the output layer of the autoencoder.

C. K-means Clustering

Clustering is performed on unlabeled observations in a dataset with the objective to group similar data samples in an unsupervised fashion. One of the most popular clustering algorithms is k-means which stands out from others with the guaranteed convergence property [19]. The hyper-parameter k defines the number of randomly assigned centroids which would be used to identify the center location for each similarly grouped data cluster. The training is done via minimizing the within-cluster sum of squares (WCSS) metric which uses squared Euclidean distances between the assigned centroid locations and observations. The centroid locations are then updated by calculating the new centroid location based on the observations assigned to the initial centroid assumption.

While k-means is easy to implement and its training is straightforward, it suffers from scalability issues where higher dimensional observations have poor clustering accuracy compared to the other methods. However, the advance of deep feature learning such as the autoencoder and convolutional autoencoder allowed for rich statistical representation of the input space at the code layers of deep neural networks which can be used with a k-means backend negating its drawbacks with high dimensional data. In this paper the latent representation of the input images at the code layer of both the autoencoder and convolutional autoencoder structures are used as inputs to the k-means algorithm.

III. Experimental Setups

A. Datasets

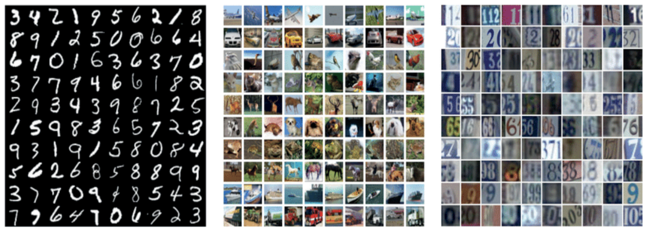

Fig. 3. MNIST(a), Cifar10(b), and SVHN(c) Images Dataset Representation

We implemented the AE and CAE based neural network structures on three different datasets; MNIST, CIFAR10, and SVHN to analyze the effect of hyper-parameters on clustering performances and associated reconstruction losses.

- MNIST [20]: MNIST is a set of black and white handwritten image examples between 0 and 9 used as a popular benchmark dataset for deep learning applications in image classification. It has 60,000 images for training and 10,000 for testing where each image contains 28x28 pixels. Fig.3a shows a collection of representative examples from this dataset.
- CIFAR10 [21]: Another popular benchmark dataset, CIFAR10, includes 10 classes with 6000 image samples per class to constitute 60,000 colored images of size 32x32 pixels where 50,000 images are used for training and 10,000 images are used for testing. The 10 different classes that are represented in the dataset include airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. Fig.3b shows a collection of representative examples from this dataset.
- SVHN [22]: Google's Street View House Numbers dataset consists of real-world images and includes 73,257 digits for training, 26,032 digits for testing with an image size of 32x32 pixels. The SVHN is similar to MNIST as it has 10 classes with numbers ranging from 0 to 9 except the images have color. Fig.3c shows a collection of representative examples from this dataset.

B. Hyperparameters

The main hyper-parameters used in this study are the network topologies (both in terms of network size and input layer structure), size of the code layer (latent space) and clustering coefficients. In AE, the general network topology is 784-128-64-32-16. Four different code sizes are defined as 128, 64, 32, and 16 with five different clustering coefficients $K = 10, ..., 50$ for every 10 incremental of K. Unlike AE, we used a different network topology for different code sizes for the CAE as shown in Fig.4 with the same five clustering coefficients.

C. Evaluation Metrics

There are different clustering performance (evaluation) metrics defined in the literature separated as internal and external metrics such the Davies–Bouldin index [23] and Dunn index [24] for internal metrics, and Purity [25], Rand Index [26], F-measure [27], Jaccard Index [28], Dice Index [29], Fowlkes-Mallows Index [30], and Confusion Matrix [31] for external metrics for a variety of applications. We use the purity evaluation metric in this study to find the clustering accuracy for both algorithms for a fair comparison of the effect of hyper-parameters. To calculate the purity metric, each cluster is labeled as the group with the most frequent samples in that cluster, and the accuracy of this assignment is measured by finding the ratio of correctly assigned observations to the general population in that cluster for each group. Its formal definition is,

$$Purity(\Omega, \mathbb{C}) = \frac{1}{N} \sum_k max_j |\omega \cap c_j| \qquad (8)$$

Where $\Omega = \omega_1, \omega_2, ..., \omega_k$ is the set of clusters and $\mathbb{C} = c_1, c_2, ..., c_j$ is the set of groups, N is the total number of data points.

D. Implementation

All implementations are done using the Keras [32] library on Google's Colab platform. There are differences in how the input data is represented based on the specific network topology being used. In the case of the autoencoder, the fully connected network topology for the encoder is D-128-64-32-16 where D is the input space dimension (feature space) size of 784 for the MNIST and 3072 for the CIFAR10/SVHN datasets and 16 is the code size (latent space). For a fair comparison of the datasets, CIFAR10 and SVHN are converted to grayscale like MNIST (1D) prior to training on the autoencoder. After training the autoencoder, the latent representations of each observation in the dataset are collected in a transformed dataset (such as 60,000 x 16 for the MNIST dataset when using a code size of 16) on which the k-means clustering algorithm is applied to find the associated purity metric for each clustering coefficient (i.e., K = 10 through 50). The centroids are initialized randomly 20 times and the purity metrics are averaged to find statistically meaningful results. A similar process is repeated for the convolutional autoencoder except the colored images in CIFAR10 and SVHN are represented via the three RGB channels available at the convolutional front layer of this topology. In order to represent the samples from the MNIST dataset, the same image is presented to each channel creating a pseudo 3D representation for a fair comparison of the network structures. Fig.4 shows the detailed overview of each topology to obtain the desired bottleneck size for each experiment. We repeat the same procedure of applying k-means using different clustering coefficients on the transformed datasets for each code size.

IV. Experimental Results and Discussions

The clustering performances of different network topologies on different image datasets for both autoencoder and convolutional autoencoder are presented in tables 1 and 2 below respectively.

Cluster Size (K)	Cifar10(CAE)				Mnist(Grey-Scaled-CAE)				SVHN(CAE)			
	Latent Space(k)				Latent Space(k)				Latent Space(k)			
	k=16	k=32	k=64	k=128	k=16	k=32	k=64	k=128	k=16	k=32	k=64	k=128
	Accuracy				Accuracy				Accuracy			
	Standard Deviation				Standard Deviation				Standard Deviation			
K=10	22.97	24.90	26.28	25.27	52.10	51.40	45.69	42.54	20.31	20.44	20.43	20.06
+/-	1.57	1.15	0.92	0.67	3.47	4.26	3.81	2.69	0.65	0.59	0.53	0.23
K=20	26.43	27.70	29.13	28.42	63.74	64.51	62.31	57.50	21.31	21.47	21.66	20.34
+/-	1.80	0.74	0.73	0.51	3.94	4.05	2.17	2.63	0.72	0.68	0.66	0.37
K=30	27.54	28.66	30.01	29.46	71.24	72.81	71.63	67.22	21.60	21.87	22.36	21.22
+/-	1.29	1.17	0.73	0.51	3.41	1.95	2.45	2.02	0.58	0.67	0.83	0.42
K=40	28.77	29.98	31.39	30.47	73.44	75.56	73.94	71.94	22.23	22.64	23.03	21.31
+/-	0.88	0.78	0.79	0.73	2.98	1.91	1.98	2.73	0.54	0.66	0.76	0.44
K=50	29.16	30.46	31.92	31.49	76.07	77.18	75.97	74.36	22.39	22.95	23.83	21.69
+/-	1.19	0.87	0.70	0.74	2.87	2.21	2.92	1.40	0.49	1.02	0.81	0.37

Table 1: Convolutional Autoencoder for MNIST(Replicated), Cifar10, and SVHN

In **Table 1**, the following observations can be made. The highest clustering accuracy values are obtained pretty consistently at two different code sizes for the three different datasets. Where the maximum accuracy is observed at the code size of K = 32 for the MNIST dataset, a higher code size of K = 64 is needed for both CIFAR10 and SVHN datasets to achieve the highest clustering accuracy. This is expected due to the inherent complexity of the images of these two datasets when compared to MNIST. A similar trend is observed for supervised classification applications where the performances reported in the literature for MNIST are significantly higher than the ones reported for CIFAR10 and SVHN. Assuming that the back-end K-means algorithm perform similarly between different datasets; a larger code size is better able to capture the latent statistics of the more sophisticated images in CIFAR10 and SVHN. Another observation is that the clustering accuracy also increases as the K factor increases for the back-end clustering algorithm. This is also expected due to the performance metric used in this study (purity) which dictates that as the number of clusters increases, the probability of samples falling into a wrong cluster decrease. For instance, at the limit, when K is equal to the number of observations, the clustering accuracy would be 100% which would have no practical meaning. A standard practice in comparing clustering accuracies is to choose the K value to be either the same or twice the number of classes in the dataset.

Cluster Size (K)	Cifar10(AE-Grey-Scaled)				Mnist(AE)				SVHN(AE-Grey-Scaled)			
	Latent Space(k)				Latent Space(k)				Latent Space(k)			
	k=16	k=32	k=64	k=128	k=16	k=32	k=64	k=128	k=16	k=32	k=64	k=128
	Accuracy				Accuracy				Accuracy			
	Standard Deviation				Standard Deviation				Standard Deviation			
K=10	20.42	20.71	20.90	20.95	64.04	66.57	66.52	59.23	20.05	19.86	19.92	19.83
+/-	0.93	0.74	0.59	0.45	3.39	1.76	1.06	2.21	0.47	0.23	0.32	0.23
K=20	23.24	24.31	23.49	23.81	73.23	74.99	73.56	70.06	20.90	20.94	20.72	20.46
+/-	1.12	0.53	0.86	0.45	2.20	2.04	1.53	2.63	0.68	0.70	0.65	0.30
K=30	24.71	24.31	24.84	24.74	78.09	79.80	77.86	76.47	21.74	21.68	21.87	21.48
+/-	1.05	0.74	0.57	0.45	2.16	1.40	1.17	1.83	1.16	0.66	0.60	0.31
K=40	25.33	25.13	25.58	25.70	81.46	82.47	81.27	79.26	22.68	22.49	22.47	22.11
+/-	0.99	0.65	0.79	0.48	1.53	1.14	1.03	1.65	1.53	0.52	0.64	0.28
K=50	26.39	26.69	26.57	26.49	82.21	83.52	83.27	81.05	23.21	23.44	23.47	22.53
+/-	0.86	1.65	0.59	0.62	1.67	0.73	1.19	1.61	1.12	0.86	0.95	0.37

Table 2: Autoencoder for MNIST, Cifar10(Greyscale), and SVHN(Greyscale)

In **Table 2**, a similar result is obtained for the MNIST dataset where the code size of k = 32 provides the highest clustering accuracy. In fact, the accuracy for the autoencoder in this case is greater than the accuracy reported for the convolutional autoencoder on the same dataset (as in table 1). This can be explained by the fact that MNIST images have been replicated at the convolutional input layers designed for an RBG colored image which increases the number of parameters to be trained in the case of CAE which may have in turn reduced the maximum possible accuracy from the network topology due to a lesser ratio of observations to weight parameter comparatively. On the contrary, the clustering accuracies are lower for CIFAR10 and SVHN when using the regular autoencoder which suggests that the convolutional layers can properly utilize the additional information coming from the colored images. This effect is more significant for the CIFAR-10 dataset where color also signifies a meaningful feature of the observation (for instance a dog or a cat, as the two classes in the dataset, can only have a specific range of color values) compared to the SVHN dataset where color

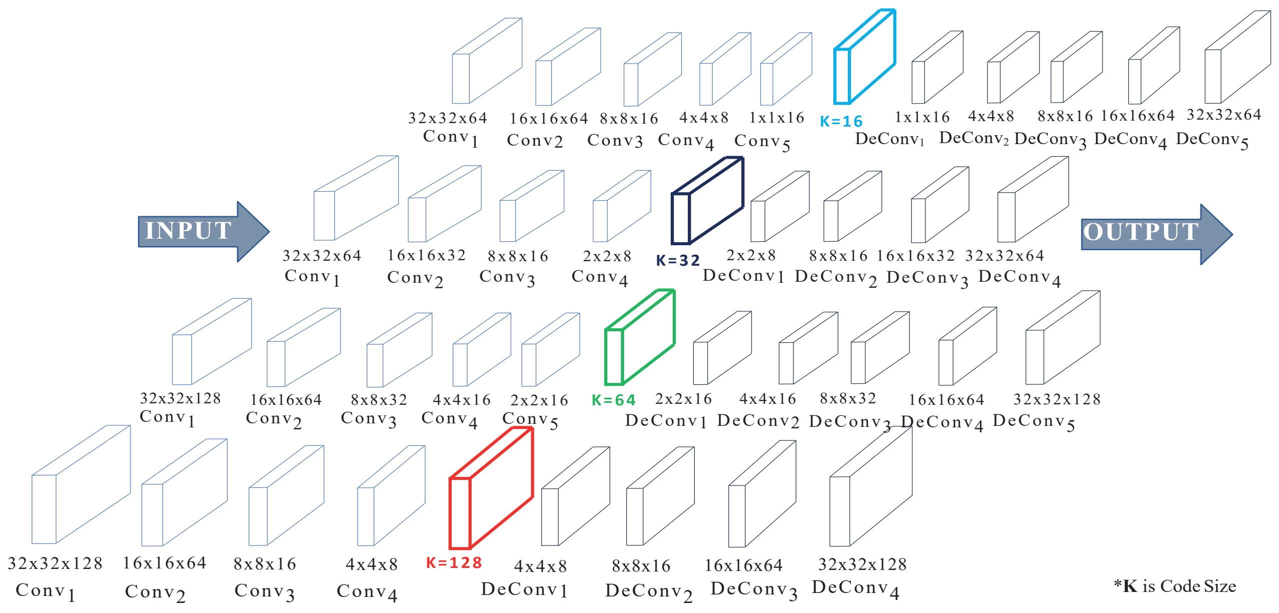

Fig. 4. Implementation Representation of Convolutional Autoencoder Structures

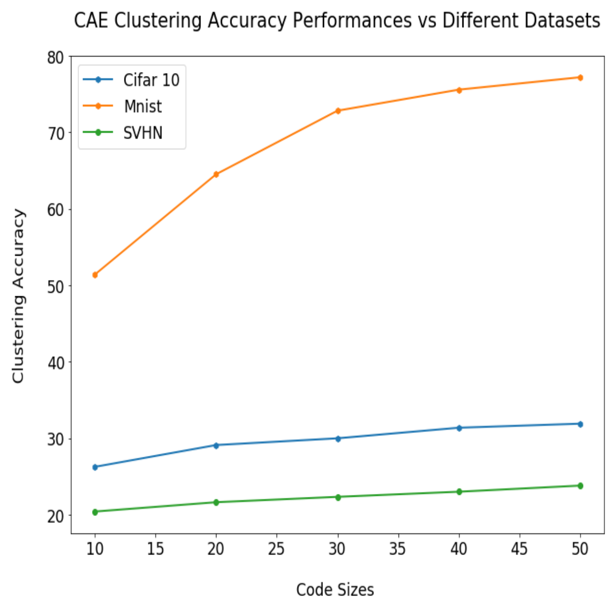

Fig. 5. Convolutional Autoencoder Accuracies Across Different Settings and Datasets

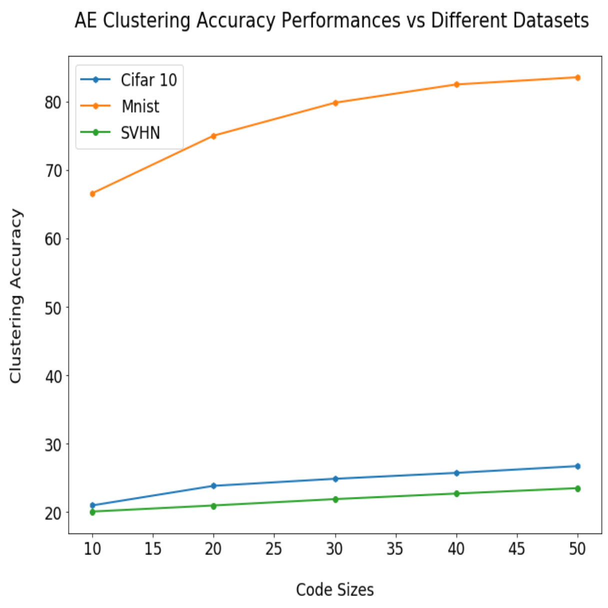

Fig. 6. Autoencoder Accuracies Across Different Settings and Datasets

is not as relevant. However, there is no standard code-size which provides the highest accuracy for all K values – which indicates that the lack of convolutional layers affects the code size required for maximum performance. In fact, for the CIFAR10 dataset, a code size of 128 (twice that of table 1) is generally required which may suggest that the information encoded in the convolutional layers is now represented (and compensated) in the increased bottleneck layer size. However, the results for SVHN do not support this conclusion where some of the high accuracies have been obtained at even lower code sizes such as 16. This is a very interesting observation which indicates that further research is required to understand such behavior and how removing convolutional layers could impact the training of the rest of the network for different datasets and how data is subsequently represented in the code layer. Figures 5 and 6 summarize the best performance curves for each topology as the latent space dimensions change.

V. Conclusions

This paper presents a comprehensive study in deep unsupervised learning to observe the effect of hyper-parameters including network topology and clustering coefficients on deep clustering accuracy. Popular autoencoder and convolutional autoencoder architectures are used to obtain clustering friendly features in the latent space where a standard k-means back-end algorithm implements the clustering. The experimental results show that the hyper-parameters influence clustering accuracy performance in both expected and unexpected ways. For instance, more complex datasets require a higher dimensional latent space to achieve the best subsequent clustering performance. For the image datasets, color can help improve the clustering performance but only when it signifies relevant information on the identity of the observation. Additional studies need to be conducted to truly understand the complex relationship between the hyperparameters, network topologies and the statistics and complexities of the datasets when it comes to deep clustering applications.

References

[1] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[2] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[3] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[4] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl *et al.*, "Constrained k-means clustering with background knowledge," in *Icml*, vol. 1, 2001, pp. 577–584.

[5] D. Reynolds, "Gaussian mixture models," *Encyclopedia of biometrics*, pp. 827–832, 2015.

[6] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[7] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[8] E. Aljalbout, V. Golkov, Y. Siddiqui, M. Strobel, and D. Cremers, "Clustering with deep learning: Taxonomy and new methods," *arXiv preprint arXiv:1801.07648*, 2018.

[9] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of machine learning research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[10] F. Li, H. Qiao, and B. Zhang, "Discriminatively boosted image clustering with fully convolutional auto-encoders," *Pattern Recognition*, vol. 83, pp. 161–173, 2018.

[11] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards k-means-friendly spaces: Simultaneous deep learning and clustering," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3861–3870.

[12] K. Ghasedi Dizaji, A. Herandi, C. Deng, W. Cai, and H. Huang, "Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5736–5745.

[13] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou, "Variational deep embedding: An unsupervised and generative approach to clustering," *arXiv preprint arXiv:1611.05148*, 2016.

[14] J. Yang, D. Parikh, and D. Batra, "Joint unsupervised learning of deep representations and image clusters," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5147–5156.

[15] C.-C. Hsu and C.-W. Lin, "Cnn-based joint clustering and representation learning with feature drift compensation for large-scale image data," *IEEE Transactions on Multimedia*, vol. 20, no. 2, pp. 421–429, 2017.

[16] X. Peng, I. W. Tsang, J. T. Zhou, and H. Zhu, "k-meansnet: When k-means meets differentiable programming," *arXiv preprint arXiv:1808.07292*, 2018.

[17] C. Song, F. Liu, Y. Huang, L. Wang, and T. Tan, "Auto-encoder based data clustering," in *Iberoamerican Congress on Pattern Recognition*. Springer, 2013, pp. 117–124.

[18] X. Guo, X. Liu, E. Zhu, and J. Yin, "Deep clustering with convolutional autoencoders," in *International Conference on Neural Information Processing*. Springer, 2017, pp. 373–382.

[19] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[20] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[21] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[22] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[23] D. Davies and D. Bouldin, "A cluster separation measure, ieee transactions on patter analysis and machine intelligence. vol," 1979.

[24] J. C. Dunn, "Well-separated clusters and optimal fuzzy partitions," *Journal of cybernetics*, vol. 4, no. 1, pp. 95–104, 1974.

[25] M. Sanderson, "Christopher d. manning, prabhakar raghavan, hinrich schütze, introduction to information retrieval, cambridge university press. 2008. isbn-13 978-0-521-86571-5, xxi 482 pages." *Natural Language Engineering*, vol. 16, no. 1, p. 100–103, 2010.

[26] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical association*, vol. 66, no. 336, pp. 846–850, 1971.

[27] Y. Sasaki, "Power control unit for high power hybrid system," in *SAE2007World Congress*, 2007, pp. 1–5.

[28] R. Real and J. M. Vargas, "The probabilistic basis of jaccard's index of similarity," *Systematic biology*, vol. 45, no. 3, pp. 380–385, 1996.

[29] L. R. Dice, "Measures of the amount of ecologic association between species," *Ecology*, vol. 26, no. 3, pp. 297–302, 1945.

[30] E. B. Fowlkes and C. L. Mallows, "A method for comparing two hierarchical clusterings," *Journal of the American statistical association*, vol. 78, no. 383, pp. 553–569, 1983.

[31] J. T. Townsend, "Theoretical analysis of an alphabetic confusion matrix," *Perception & Psychophysics*, vol. 9, no. 1, pp. 40–50, 1971.

[32] F. Chollet *et al.*, "Keras," 2015.