Learning Embedding Space for Clustering From Deep Representations

Paras Dahal
Marax AI
Bangalore, India
parasdahal99@gmail.com

***Abstract*—Clustering is one of the most fundamental unsupervised tasks in machine learning and is elementary in the exploration of high volume data. Recent works propose using deep neural networks for clustering, owing to their ability to learn powerful representations of the data. In this work, we present a novel clustering approach using deep neural networks that simultaneously learns feature representations and embeddings suitable for clustering by encouraging separation of natural clusters in the embedding space. More specifically, an autoencoder is employed to learn representations of the data. Then a mapping from autoencoder representation space to an embedding space is learned using a deep neural network we call Representation Network. This neural network promotes separation between natural clusters by minimizing cross-entropy between two probability distributions that denote pairwise similarity in autoencoder latent space and representation network's embedding space. The resultant optimization problem can be solved effectively by jointly training the autoencoder and the representation network using minibatch stochastic gradient descent and backpropagation. Ultimately we obtain a K-Means friendly embedding space. Experimental results show that despite being a simple model, the proposed approach outperforms a broad range of recent approaches on Reuters dataset, other autoencoder based models on MNIST dataset and produces consistently good results that are very competitive with other complex and hybrid models.**

***Keywords*-Clustering; deep learning; representation learning;**

I. Introduction

Clustering is a fundamental unsupervised problem whose goal is to group similar objects together. It is an important area of research and its various aspects such as distance metrics, feature selection, grouping methods etc. have been extensively studied. K-Means [1] and GMM [2] are two very popular clustering algorithms. These algorithms perform clustering using distance-based similarity measure typically on low-level features like raw data or gradient orientation histograms (HoG). In high dimensional data, distance metrics are not a good similarity measure as the distance between points become relatively uniform [3]. Thus these algorithms are not suitable for clustering high dimensional data. Spectral Clustering [4] is another widely used algorithm. It initially does dimensionality reduction using Laplacian spectra of the distance matrix and then performs clustering on this new feature space. But Spectral Clustering is difficult to scale to large datasets as full Laplacian matrix needs to be computed which is computationally intensive.

Given widespread success of deep learning, a large number of recent studies have been focused on learning a good representation of data in both supervised and unsupervised setup. Unsupervised scenario like clustering poses a unique challenge as no supervision signal like class labels are available to guide the representation learning process. Because of this, neural networks are unconstrained to preserve the desired properties in the learned representations. Therefore to learn clustering suitable representations, different regularization techniques are required. Many recent deep clustering approaches use autoencoders to learn low dimensional data representations. The small dimension of data representation space constraints the autoencoder to learn representations that preserve the most salient features of data distribution, thus making the latent representations more effective for clustering than the original high dimensional data. However, to achieve better performance, the representation learning process can be guided with an explicit clustering loss to attain a more clustering friendly space. To this end, DEC [5] proposes a clustering loss based on self-training scheme that directly manipulates the representation space to be more clustering friendly. But as identified before [6], such direct manipulation with clustering loss may cause corruption of feature space as the clustering loss cannot guarantee local structure preservation. Preserving local structure is essential to clustering as it contains discriminative information that is important for distinguishing different samples as shown in [7]. Some deep clustering models manipulate the feature space by jointly optimizing both reconstruction loss and clustering loss. For example, DCN [8] proposes K-Means loss as clustering loss and DEN [9] proposes locality preserving and group sparsity loss. Some recent works [10] [11] [12] take a different approach by optimizing the neural network with only clustering loss. In this work, however, we are motivated to develop a simple yet effective model based on autoencoders to learn a clustering friendly embedded space that separates the natural clusters in the data well.

We propose a clustering model in which we employ an autoencoder to learn data representations. We then introduce a deep neural network that aims to learn a clustering friendly embedded space from these data representations. We refer to this deep neural network as Representation Network. This neural network parameterizes the mapping from latent representation space Z of the autoencoder to an embed-

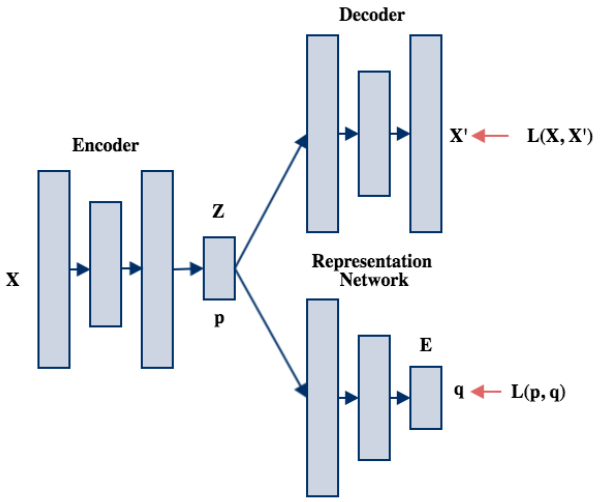

Figure 1. Network structure of the proposed method. The reconstruction loss helps to learn useful feature representations in the latent space Z. The representation network parameterizes a mapping from latent space Z to an embedding space E. p and q are probabilities of pairwise similarity of points in Z and E respectively. The representation network loss encourages separation of natural clusters in the embedding space E.

ding space E by minimizing cross-entropy between two probability distributions that denote pairwise similarity of points in autoencoder latent space and the embedding space respectively. The training procedure of our model learns encoder, decoder and representation network parameters simultaneously. We then perform K-Means clustering on the embedding space to get the clusters.

The main contributions of this paper are:

1) We propose a simple yet effective clustering model that simultaneously learns both valid feature space Z and embedding space suitable for clustering E.
2) We introduce a novel architecture by incorporating representation network to autoencoder framework.
3) We show how to jointly optimize the autoencoder with the representation network.
4) Experimental results show our model achieves superior performance on Reuters dataset among a broad range of recently proposed models. On MNIST dataset, it performs very competitively if not superior results.

II. Related Work

Clustering using deep neural networks have received much attention in recent time, given their powerful capability of learning complex nonlinear dependencies. Most of the proposed deep clustering models focus on learning a good high-level representation of data to facilitate clustering task. Existing deep clustering algorithms broadly fall into 3 categories: (i) Autoencoder based (ii) Generative model based and (iii) Direct optimization under clustering objective. For a general survey on clustering we refer the readers to [13] and for the survey on very recent works on deep clustering, we refer to [14].

In autoencoder based approaches, one of the pioneering work is Deep Embedded Clustering (DEC) [5]. DEC starts with pretraining an autoencoder to initialize encoder parameters and then removes the decoder. The encoder is then fine-tuned by optimizing a KL divergence between soft cluster assignment and an auxiliary target distribution derived from this assignment. This training can be thought of as a self-training process to refine the representations while doing cluster assignment iteratively. Discriminately Boosted Clustering (DB C) [15] builds on DEC by using convolutional autoencoder. Deep Clustering Network (DCN) [8] jointly optimizes the reconstruction loss and K-means loss with alternating cluster assignments. Deep Embedded Regularized Clustering (DEPICT) [16] uses softmax layer stacked on top of convolutional autoencoder. It jointly optimizes reconstruction loss and balanced cluster assignment loss. A very recent approach called Deep Continuous Clustering (DCC) [17] is based on Robust Continuous Clustering [18]. It also jointly optimizes reconstruction loss with RCC objective.

Generative model based clustering methods extend the clustering task to generating samples from clusters, which can be useful in several applications. VaDE [19] incorporates probabilistic clustering problem within the framework of VAE [20] by imposing a GMM priori over VAE. The optimization essentially minimizes reconstruction loss and KL divergence between Mixture of Gaussians prior to the variational posterior. VaDE stands on the strong theoretical ground of VAE. InfoGAN [21] is another generative approach under GAN [22] framework that learns disentangled representations. Learning disentangled representation can be beneficial for clustering but GANs being a complex model are notorious for their optimization difficulty.

The third category of deep clustering models differs such that they only use clustering loss to optimize the deep neural network. JULE [10] proposes a convolutional neural network with agglomerative clustering loss. As agglomerative clustering requires the construction of undirected affinity matrix, JULE suffers from computational and memory complexity issues causing scalability problems. Deep Adaptive Image Clustering (DAC) [11] uses convolutional neural network with a binary pairwise classification as clustering loss and adds a regularization constraint that helps learn label features as one hot encoded features. IMSAT [12] learns discreet representations of data using information maximization between input and cluster assignment and employs data augmentation technique called Self Augmentation Training.

The following are the notable differences in our proposed model and the above-discussed approaches:

1) Autoencoder based models directly manipulate the encoded representations to make them clustering

Table I
SUMMARY OF NOTATIONS

Notation	Description
X	Training sample set
Z	Latent representation space of autoencoder
E	Embedded space of representation network
L_R	Reconstruction loss
L_E	Representation network loss
W, W'	Autoencoder parameters
ϕ	Representation network parameters
p	Probability matrix of pairwise similarity in Z
q	Probability matrix of pairwise similarity in E
α	Degree of freedom for t-distribution of p
γ	Weighing coefficient for L_E
d	Dimension of Z and E

friendly. However, we presume there is a tradeoff involved in preserving reconstruction capability and separation of clusters when manipulating the encoded representations. We circumvent this tradeoff by using clustering loss directly on a separate neural network instead of the encoder, creating a separate embedding space for clustering task.

2) Compared to the generative models, the proposed model is simpler to implement, easier to train and achieves better performance in our experiments.

III. OUR APPROACH

Let X, Z and E denote the domains of the inputs, latent representations, and clustering embeddings. Given training samples $x_1, x_2, ..., x_N$, we aim to find a non linear mapping $f : X \rightarrow Z$, and $g : Z \rightarrow X'$, where each sample $x_i \in X$ has latent representation $z_i \in Z$ and reconstructed sample $x'_i \in X'$. We then find another mapping $h : Z \rightarrow E$ from the latent representation space to embedding space parametrized by representation network. The goal of our approach is to learn embeddings suitable for clustering task by simultaneously learning the above-mentioned mappings.

We organize Section III as follows. In section A, we review autoencoders. In section B, we introduce Representation network and its objective function. In section C, we discuss the optimization procedures of encoder, decoder and representation network.

A. Review of Autoencoders

Autoencoder is a neural network that is trained to copy its input to its output. It found its initial use in non-linear dimensionality reduction and eventually feature learning and denoising. An autoencoder consists of two parts: an encoder function $z = f(x)$ and a decoder function $x' = g(z)$ that attempts to reconstruct the original input. Generally, autoencoders are designed to be unable to perfectly copy the input to its output, hence constraining them to learn useful properties of data. The model is trained in an unsupervised manner by minimizing the reconstruction error of the decoder output with original input, typically using minibatch gradient descent following the gradients computed by backpropagation.

1) Basic Autoencoder: The encoder is a function that maps the data $x \in R^{d_x}$ to d_z hidden units to get latent representation as:

$$z = f(x) = s_f(Wx + b_z)$$

where s_f is a nonlinear activation function. Encoder has parameters as weight matrix W and bias vector b_z.

The decoder function g maps the outputs of hidden units to the original input space as:

$$x' = g(z) = s_g(W'z + b_o)$$

where s_g is a nonlinear activation function which is typically the same as the encoder. Decoder has parameters weight matrix W' and bias vector b_o.

The model parameter $\theta = \{W, b_h, W', b_o\}$ are learned by minimizing the reconstruction error on training data. The objective function is given by:

$$L_R = \sum_{i=1}^{n} L(x_i; g(f(x_i)))$$

where L is a loss function, typically a mean squared error $L(x_i, x'_i) = \sum_{i=1}^{n} ||x_i - x'_i||^2$.

If the input lies in range [0,1], cross-entropy loss $L(x_i, x'_i) = \sum_{i=1}^{d_x} x_i log(x'_i) + (1 - x_i) log(1 - x'_i)$ is usually chosen.

2) Undercomplete Autoencoder: If encoder and decoder functions are allowed too much capacity, the autoencoder can learn to simply copy the input to its output without learning much useful information about data distribution. One way to force autoencoder to learn useful features is by setting the dimension of latent space representation z to be much smaller than x. This forces the encoder to learn useful representation that preserves important features of data generating distribution which is why reconstruction is possible. This configuration is called Undercomplete Autoencoder.

Autoencoders with nonlinear encoder and decoder functions can learn a powerful nonlinear generalization of PCA, learning the most salient features of the data distribution [23].

3) Denoising Autoencoder: Denoising Autoencoder (DAE) [24] receives corrupted data points as input and is trained to predict the original, uncorrupted data point as its output, allowing representations to be robust to partial corruption of the input patterns. An artificial corruption process $C(\widetilde{x}|x)$ is introduced which is typically an additive isotropic Gaussian noise, $x = x + \epsilon, \epsilon \sim N(0, \sigma^2 I)$. Instead of reconstructing the original input sample, DAE learns to reconstruct clean input x from the corrupted version $\widetilde{x}$. Training is done by optimizing the following objective function

$$L_R = \sum_{i=1}^{n} E_{\widetilde{x} \sim C(\widetilde{x}|x_i)}[L(x_i, g(f(\widetilde{x})))] \tag{1}$$

Denoising training forces f and g to implicitly learn the structure of data [25].

In our algorithm, we use denoising autoencoder with cross-entropy reconstruction loss.

B. Representation Network

Autoencoders, constrained by the smaller dimension of latent space Z, are known to exploit the idea that data generating distribution concentrates near a low-dimensional manifold. They try to model representations that implicitly capture local coordinate system for this manifold in the latent space [23]. However, this leads to the formation of compactly positioned representations with severe overlapping between their natural clusters. Because of this, performing clustering on the latent space Z is not very effective. To achieve better clustering, the encoder needs to be manipulated in a way that natural clusters in representation space are separated well. But such direct manipulation brings a risk of corrupting the representation space leading to inferior results [6].

In order to circumvent this problem, we propose to incorporate a Representation Network which is a fully connected deep neural network into the autoencoder framework. This network aims to learn the mapping from latent space Z to embedding space E i.e. $h : Z \rightarrow E$ by learning the weights ϕ that preserves the local structure of latent representations and encourages larger separation of natural clusters in the embedded space. This results in embeddings that have dense but well-separated clusters, which makes clustering on these embeddings more effective than the original representation space. Note that the representation network performs no dimensionality reduction as the dimension of latent space Z and E are kept the same.

To capture the local structure of data, we convert the pairwise distance in both latent space and embedded space into probabilities. We use Student's t-distribution to measure the pairwise similarities in the latent space Z and define a p_{ij} as below:

$$p_{ij} = \frac{(1 + ||f(x_i) - f_(x_j)||^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{k \neq l}(1 + ||f(x_k) - f(x_l)||^2/\alpha)^{-\frac{\alpha+1}{2}}} \tag{2}$$

where $f(x_i) = z_i \in Z$ corresponds to representations $x_i \in X$, f is encoder function and α is the degree of freedom of Student's t-distribution. This probability p_{ij} can be interpreted as the the probability of z_i andz_j lying close on the latent representation space Z.

While using a Gaussian similarity kernel for latent space Z may seem like a logical choice, we opt to use Student's t-distribution because of the following reasons:

1) Student's t-distribution approximates Gaussian distribution for high values of degrees of freedom.
2) We observe that it assigns stricter probabilities (for higher degrees of freedom) which makes it suitable target distribution to learn from high confidence pairs.
3) It eliminates the need to tune or estimate kernel width avoiding additional parameter like perplexity or number of effective neighbors.

The goal of p is to convert similarities of representations in latent space into probabilities. We set a high degree of freedom for p as the high value of the degree of freedom α allows more space to model the local structure and the smaller separation of clusters in latent space [26]. In our experiments, we set α for p to $2d$ where d is the dimension of latent representation space. Our experiments prove that degree of freedom $2d$ gives the best performance on all the datasets.

Similar to the clustering loss in DEC, we use Student's t-distribution with the degree of freedom 1 in the embedded space E and define a distribution q_{ij} as below:

$$q_{ij} = \frac{(1 + ||h(z_i) - h(z_j)||^2/)^{-1}}{\sum_{k \neq l}(1 + ||h(z_k) - h(z_l)||^2/)^{-1}} \tag{3}$$

We choose degrees of freedom 1 for q as a low degree of freedom causes larger separation between natural clusters [26]. We can see in figure 2 that q starts off by assigning a low probability to most pairs. As the training progresses we can observe how q starts to assign a higher probability to pairs having high probability in p pushing these pairs together in the embedded space while creating a strong repulsive force between low probability pairs. As q is limited by its low degree of freedom to model local structure, it approaches target distribution p by increasing inter-cluster distance and reducing intra-cluster distance in the embedded space E. This results in good separation of the clusters in the embedded space.

The weights of representation network ϕ are learned by minimizing the cross-entropy between the distributions p and q defined as

$$L_E = -\sum_i \sum_j p_{ij} log(q_{ij}) \tag{4}$$

$$= H(p) + KL(p||q) \tag{5}$$

We choose to minimize cross-entropy between distributions p and q as this is equivalent to minimizing the entropy of distribution p as well as the KL divergence between p and q. This cross-entropy loss and the representation network achieve two following effects:

1) Due to the inter-cluster strong repulsive force caused by Student's t-distribution, the clusters are pushed farther away and overlapping is minimized.
2) As p is not fixed during joint training, optimizing cross-entropy loss regularizes the encoder f to scatter

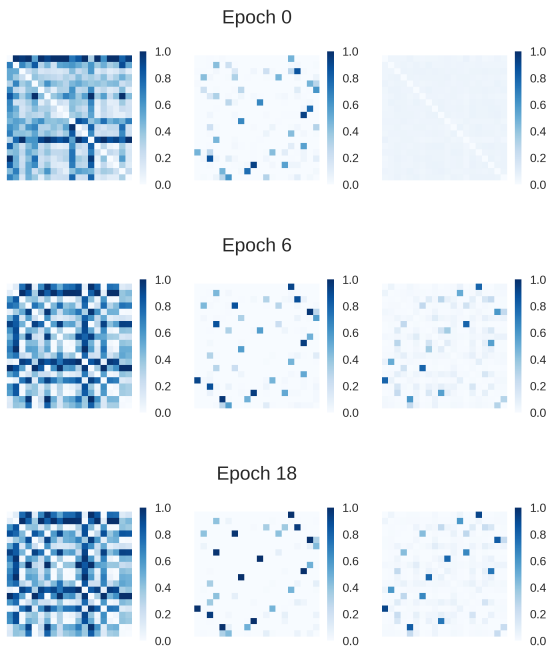

Figure 2. Visualization of distance between embeddings in E (scaled to [0,1]), p and q respectively of a subset of MNIST as the training progresses.

points in latent space Z in such a way that entropy $H(p)$ is minimized.

Algorithm 1 Model training and cluster assignment

Input: Input data $X = \{x_i, ..., x_n\} \in X$, batch size b, number of clusters k
Output: Cluster assignments y
Initialize W, W' according to pretraining described in section III-C1
repeat
Compute points in latent space $\{z_i = f(x_i)\}_{i=1}^{n}$
Compute points in embedded space $\{e_i = h(z_i)\}_{i=1}^{n}$
Choose a batch $S \in X$ of size b
Update W, W' and ϕ on S
until stopping criteria is met
$\mathbf{Z} \leftarrow f(\mathbf{X})$
$\mathbf{E} \leftarrow f(\mathbf{Z})$
$\mathbf{y} \leftarrow kmeans(\mathbf{E}, k)$

C. Optimization

Optimization of this algorithm is done in two stages:

1) Pre-training: In the pre-training stage, we train the autoencoder with cross-entropy reconstruction loss. The objective function is minimized using the standard backpropagation algorithm. This gives a good parameter initialization for encoder network as it starts to learn salient features of data distribution. After pre-training, points in latent space Z are valid feature representations for input samples.

2) Joint Training: In the joint training phase, we introduce the representation network. Specifically, there are three types of parameters to optimize during joint training: encoder, decoder, and representation network.

Encoder We update encoder parameters with a weighted sum of autoencoder reconstruction loss and representation network's cross-entropy loss. We introduce a coefficient γ that controls the degree of distortion of the encoder's latent space Z. The objective function is given as:

$$L = L_R + \gamma L_E \tag{6}$$

We presume modifying the latent space Z slightly with embedded loss will not cause corruption, but it, in fact, regularizes the encoder weights to scatter points in the latent space so that entropy term $H(p)$ of equation (6) is minimized as discussed above.

Representation Network The responsibility of representation network is to promote separation of clusters in the embedding space Z. We update representation network parameters using only the cross-entropy loss from equation (4).

Decoder Decoder's role in joint training is to reconstruct the input samples from latent space Z even when encoder's weights are distorted a little by the joint loss. This ensures the latent space representations a are valid feature representations throughout the training. We update decoder's parameters with reconstruction loss.

IV. Experiments

In the following sections, we describe the experiments and compare the performance of our proposed model with that of several recently proposed deep clustering models. The model has been implemented in Python using Tensorflow[27].

A. Datasets

We evaluate our model on popular benchmark image dataset and text dataset.

- **MNIST**: The MNIST dataset[28] consists of 70,000 grayscale handwritten images of 28x28 pixels with 10 classes.
- **REUTERS**: Reuters [29] contains about 810,000 English news stories labelled with a category tree. Following experiments from DEC, we use 4 root categories: corporate/industrial, government/social, markets and economics as labels and excluded all documents with multiple labels, which results in 685,071 articles. We then compute tf-idf features of 2000 most frequently occurring words to represent all articles.

We preprocess both the datasets by normalizing each example x so that $x \in [0, 1]$.

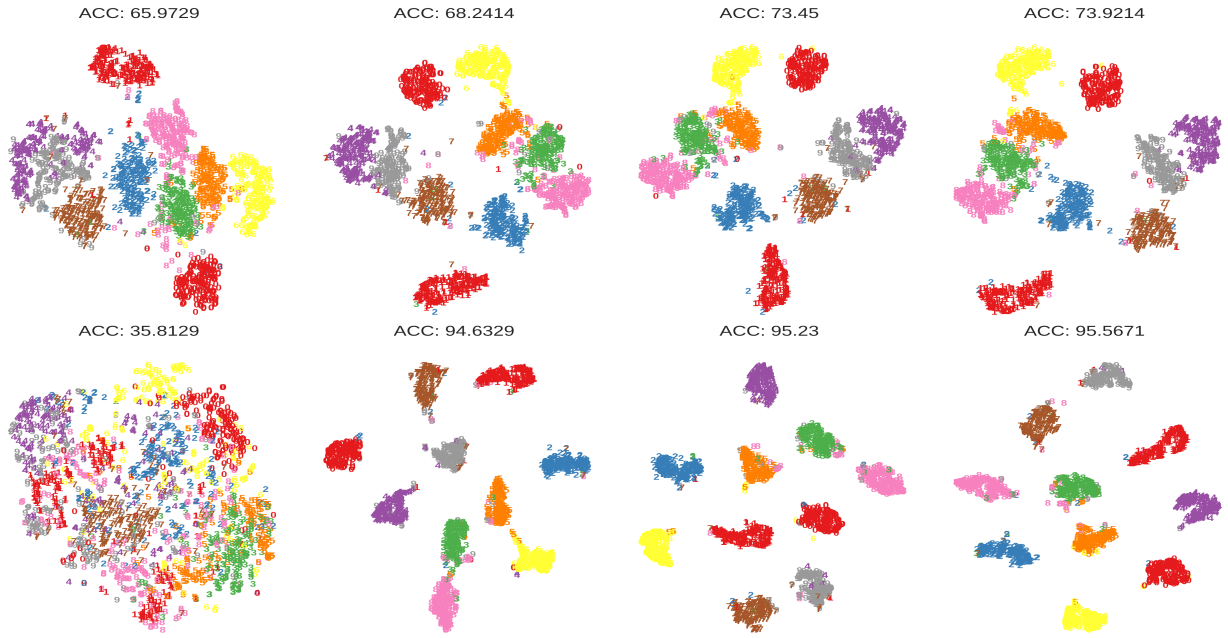

Figure 3. t-SNE visualization of 5000 random MNIST representations and embeddings as the training proceeds. First row is the representations from the latent space Z and second row is the corresponding embeddings from the embedded space E. Color code denotes the ground truth for the digits.

B. Experiment Setup

1) Network Architecture: We use similar network architecture as DEC. The architecture of encoder and decoder network are $D-500-500-2000-d$ and $d-2000-500-500-D$ respectively where D is input dimensionality and d is the dimension of the latent representation space. The architecture of representation network is $d-2000-500-500-d$ which is the same as the decoder except for the output layer d which is also the dimension of the embedding space. All layers are fully connected layers. ReLU [30] activation is used for all the hidden layers.

2) Parameter Setting: We use Adam [31] optimizer with minibatch size 100. The learning rate for MNIST is set to 0.001 and for Reuters, it is set to 0.0005. After pretraining, the coefficient γ is set to 0.01. k is set to the number of classes of each dataset and we do 10 random restarts of K-Means and pick the result with best objective value.

3) Evaluation Metric: We use unsupervised clustering accuracy which is a popular evaluation metric for deep clustering algorithms. It is defined as follows:

$$ACC = max_{m \in M} \frac{\sum_{i=1}^{N} 1\{l_i = m(c_i)\}}{N}$$

where N is the total number of samples, l_i is the ground truth label, c_i is the cluster assignment obtained by the model, M is the set of all possible one to one mapping between cluster assignments and labels. The best mapping can be obtained using Hungarian Algorithm [32].

C. Results

We report the quantitative comparison of our model on two datasets in Table II. We also present reported results of a broad range of recently proposed models for comparison. They are grouped according to the three different avenues of approaches as described in II. The network type either fully connected (FCN) or convolutional (CCN) is also reported. The results of the experiments can be summarized as follows:

1) The proposed model achieves 83.62% on Reuters which is the highest reported clustering accuracy among all the compared models, defining a new state of the art for this dataset.
2) The model also significantly outperforms all the autoencoder based approaches (both CNN and FCN) on MNIST dataset with 97.08%.
3) IMSAT reports the highest clustering accuracy on MNIST with 98.4%, but it is notable that it doesn't perform well on Reuters with just 71.9%. However, the proposed model achieves a consistently good result on both datasets suggesting suitability with both image and text datasets.
4) Despite being relatively simple, the proposed model achieves very competitive if not superior results with complex and hybrid models.

This suggests the addition of representation network to autoencoder framework helps in learning embeddings much suitable for clustering.

Table II
CLUSTERING ACCURACY (%) PERFORMANCE COMPARISON ON MNIST AND REUTERS DATASETS.

Method	Network	MNIST	Reuters
AE+K-Means	FCN	79.8	71.97
DEC [5]	FCN	84.3	75.63
DBC [15]	CNN	96.4	-
DCN [8]	FCN	83.0	80.0
DCC [17]	Both FCN/CNN	96.3	59.6
DEPICT [16]	CNN	96.5	-
VaDE [19]	FCN	94.46	79.83
InfoGAN [21]	CNN	95.0	-
JULE [10]	CNN	96.1	-
DAC [11]	CNN	97.75	-
IMSAT [12]	FCN	**98.4**	71.9
Our approach	FCN	97.08	**83.62**

V. DISCUSSION

A. Contribution of Representation Network

In Figure 3, we use t-SNE [33] visualization to observe the progression of latent space Z and embedded space E for a randomly sampled subset of MNSIT. We observe that in latent space Z, clusters are separated, but still lie close to each other and are not very dense. In the embedding space E however, we see that the natural clusters are separated very well and are very dense. This shows that the embeddings produced by representation network are much better suited to perform clustering.

B. Impact of Number of Clusters

For the clustering results above, we set the number of clusters to prior knowledge i.e. $k = 10$ for MNIST. To demonstrate the representation power of the proposed approach as an unsupervised model, we choose different numbers of clusters k and observe the samples from those clusters on MNIST. In figure 4, each row corresponds to samples grouped by our approach as a cluster. On the top image, we set $k = 7$ while on the bottom image we set $k = 14$. We can observe that for k smaller than the actual number of class i.e. 10, digits with a similar appearance like 4,9 and 3,5,8 are grouped together. When k is larger than the actual number of clusters, some digits are divided into subgroups based on slight visual variation. It can be seen that straight and tilted digits like 7, 1 and 6 are separated into new clusters.

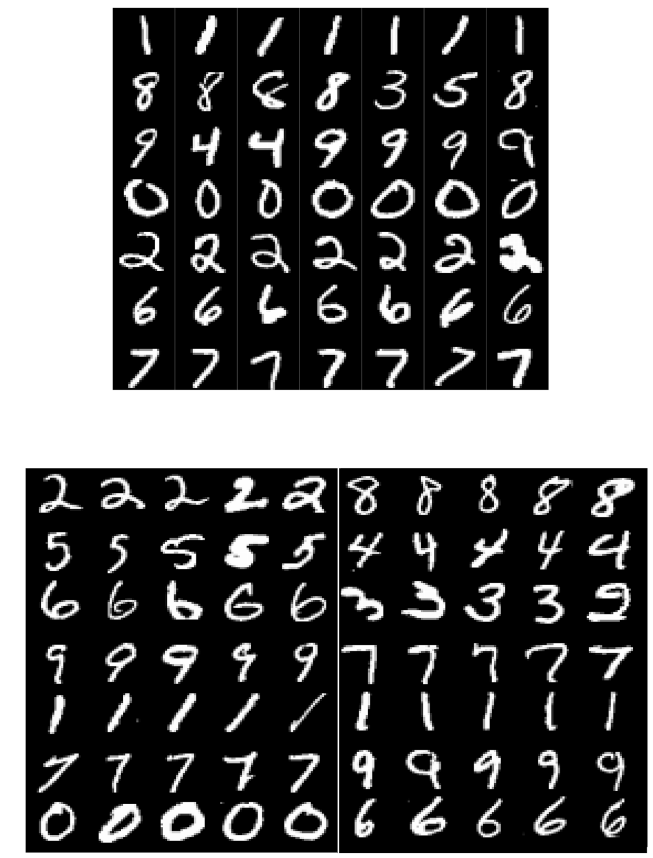

Figure 4. Clustering MNIST with different number of clusters. Top image has k=7 while bottom image has k=14. Note that when K is smaller than actual clusters, similar digits are grouped together whereas when k is larger, variations of the same digit are grouped into separate clusters.

VI. CONCLUSION

This paper presents a novel clustering approach using deep neural networks which jointly learns the feature representation and its clustering friendly embedding. We introduce representation network as an addition to the autoencoder framework that learns clustering friendly embeddings from the latent representation space of autoencoder. Clustering on these embedding over latent space representations is clearly more effective. The representation network is trained along with autoencoder to encourage separation of natural clusters in data by optimizing cross-entropy based loss of probabilities in latent and embedded space. Experimental results have shown that our proposed approach outperforms all recently proposed approaches on Reuters dataset and also

outperforms other autoencoder based models on MNIST dataset and produces consistently good results that are competitive with other complex models. Note that although we use fully connected layers for both autoencoder and representation network, use of convolutional layers for image datasets can be adapted flexibly, which will be our future work.

ACKNOWLEDGMENT

This paper is based on the author's work as an AI Associate with Marax AI, Inc. The author would like to thank the team at Marax AI, Inc. for helpful guidance.

REFERENCES

[1] 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.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[3] M. Steinbach, L. Ertöz, and V. Kumar, "The challenges of clustering high dimensional data," in *New directions in statistical physics*. Springer, 2004, pp. 273–309.

[4] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[5] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International conference on machine learning*, 2016, pp. 478–487.

[6] X. Guo, L. Gao, X. Liu, and J. Yin, "Improved deep embedded clustering with local structure preservation," in *International Joint Conference on Artificial Intelligence (IJCAI-17)*, 2017, pp. 1753–1759.

[7] L. Bottou and V. Vapnik, "Local learning algorithms," *Neural computation*, vol. 4, no. 6, pp. 888–900, 1992.

[8] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards k-means-friendly spaces: Simultaneous deep learning and clustering," *arXiv preprint arXiv:1610.04794*, 2016.

[9] P. Huang, Y. Huang, W. Wang, and L. Wang, "Deep embedding network for clustering," in *Pattern Recognition (ICPR), 2014 22nd International Conference on*. IEEE, 2014, pp. 1532–1537.

[10] 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.

[11] J. Chang, L. Wang, G. Meng, S. Xiang, and C. Pan, "Deep adaptive image clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5879–5887.

[12] W. Hu, T. Miyato, S. Tokui, E. Matsumoto, and M. Sugiyama, "Learning discrete representations via information maximizing self-augmented training," *arXiv preprint arXiv:1702.08720*, 2017.

[13] C. C. Aggarwal and C. K. Reddy, *Data clustering: algorithms and applications*. CRC press, 2013.

[14] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long, "A survey of clustering with deep learning: From the perspective of network architecture," *IEEE Access*, vol. 6, pp. 39 501–39 514, 2018.

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

[16] K. G. Dizaji, A. Herandi, C. Deng, W. Cai, and H. Huang, "Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization," in *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 2017, pp. 5747–5756.

[17] S. A. Shah and V. Koltun, "Deep continuous clustering," *arXiv preprint arXiv:1803.01449*, 2018.

[18] S. Shah and V. Koltun, "Robust continuous clustering," *Proceedings of the National Academy of Sciences*, vol. 114, no. 37, pp. 9814–9819, 2017. [Online]. Available: http://www.pnas.org/content/114/37/9814

[19] 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.

[20] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[21] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in neural information processing systems*, 2016, pp. 2172–2180.

[22] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[23] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[24] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.

[25] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[26] L. Maaten, "Learning a parametric embedding by preserving local structure," in *Artificial Intelligence and Statistics*, 2009, pp. 384–391.

[27] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[28] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[29] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "Rcv1: A new benchmark collection for text categorization research," *Journal of machine learning research*, vol. 5, no. Apr, pp. 361–397, 2004.

[30] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 315–323.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[32] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[33] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.