Deep Convolutional K-Means Clustering

Anurag Goel, Angshul Majumdar, Emilie Chouzenoux, Giovanni Chierchia

DEEP CONVOLUTIONAL K-MEANS CLUSTERING

Anurag Goel[1, 2] Angshul Majumdar[1] Emilie Chouzenoux[3] Giovanni Chierchia[4]

[1]Indraprastha Institute of Information Technology, New Delhi, India
[2]Delhi Technological University, New Delhi, India
[3]CVN, Inria, Univ. Paris Saclay, Paris, France
[4]ESIEE, Paris, France

ABSTRACT

Conventional Convolutional Neural Network (CNN) based clustering formulations are based on the encoder-decoder based framework, where the clustering loss is incorporated after the encoder network. The problem with this approach is that it requires training an additional decoder network; this, in turn, means learning additional weights which can lead to over-fitting in data constrained scenarios. This work introduces a Deep Convolutional Transform Learning (DCTL) based clustering framework. The advantage of our proposed formulation is that we do not require learning the additional decoder network. Therefore our formulation is less prone to over-fitting. Comparison with state-of-the-art deep learning based clustering solutions on benchmark image datasets shows that our proposed method improves over the rest in challenging scenarios where there are many clusters with limited samples.

Index Terms— Convolutional Neural Network, K-means Clustering, Convolutional Transform Learning.

1. INTRODUCTION

In a regular feedforward neural network, the data are projected by the network to form the representation. Usually, such a neural network is used for supervised tasks and the generated features are projected through another network to the label / output space. While training, the network weights are learnt by gradient descent / backpropagation. When such a neural network does not have any output, backpropagation leads to a trivial solution –network weights are zeroes and representations are also zeroes. This solution is reached irrespective of the cost function; changing it from Euclidean to Manhattan or KL divergence does not change the trivial solution.

This is the reason conventional feedforward neural networks without any output cannot be used for unsupervised representation learning. The situation does not change when an unsupervised clustering loss is incorporated after the unsupervised representation layer. Irrespective of the clustering loss (K-means, Sparse Subspace, Spectral, etc.) [1], the same trivial solution (network weights are zeroes and representations are zero) is reached. This is the reason clustering loss cannot be embedded in a standard feedforward neural network.

To avoid this issue, prior studies on deep learning based clustering (to be discussed in detail in section II) incorporated clustering losses into the autoencoder framework. Since autoencoder is a self-supervised neural network, backpropagation does not lead to the trivial solution and the framework can be used for unsupervised feature extraction. Embedding a clustering loss after the representation layer of an autoencoder leads to a meaningful non-trivial solution.

However, an autoencoder requires learning twice the number of network weights compared to a standard feedforward neural network. This is because they need to learn an encoder and decoder network; a regular feedforward neural network only requires learning the encoder portion. The decoder portion of the autoencoder is not useful for analysis; its sole purpose is to prevent the trivial solution. The requirement of learning twice the number of network weights may lead to over-fitting in data constrained scenarios.

Conversely, one could have used the unsupervised framework of the restricted Boltzmann machine (RBM) for embedding clustering losses. These are undirected graphs and do not suffer from the trivial solution. In a sense, RBM is more optimal than autoencoders since the former needs to learn only half the number of parameters compared to the latter. However, the RBM cost function is not mathematically amenable; its training via contrastive divergence does not have the same flexibility as that of backpropagation. Strictly speaking, contrastive divergence can only optimize the RBM cost function approximately. This perhaps had been the main deterrent behind the use of RBM based clustering solutions.

The shortcomings of existing solutions are pertinent to the Convolutional Neural Network (CNN) as well. Unless there is an output, CNN ends up in the trivial solution. The only way to prevent the trivial solution is to have deconvolution layers, but then the formulation suffers from the possibility of over-fitting.

In the recent past, we have developed the framework of con-

volutional transform learning (CTL) [2] and deep CTL [3]. This framework is able to learn convolutional filters in an unsupervised fashion without leading to the trivial solution. Furthermore, CTL guarantees the uniqueness of the learnt filters. These advantages of CTL and its deep version makes them ideal candidates for embedding clustering losses. Here we have incorporated K-means clustering since it is the most popular. However, it is possible to incorporate other clustering losses as well.

The contributions of this work can be summarized as -

• We propose a new formulation for deep convolutional transform learning based K-means clustering that is less prone to over-fitting compared to existing solutions since we do not need to learn the deconvolution/decoder layers.

• Our proposed method shows superior clustering performance in the challenging scenarios where data comprise of high number of clusters and low volume of samples.

• The execution speed of our proposed method is of the same order as that of K-means clustering and is much faster than the deep learning based clustering approaches compared against.

The rest of the paper is organised into several sections. A brief review of various deep learning based clustering formulations is discussed in the next section. Furthermore, as CTL is a relatively new framework, we will discuss it in depth in Section 2. This is necessary for understanding our formulation discussed in Section 3. The experimental evaluation is detailed in Section 4. The conclusion is discussed in Section 5.

2. BACKGROUND

2.1. Deep Learning Based Clustering

In [4], a deep clustering model based on stacked autoencoder is proposed where the clustering layer is embedded after the encoder network. This deep embedded clustering model is trained in piecemeal fashion. Later, [5] proposed the jointly learnt formulation of the stacked autoencoder embedded with the sparse subspace clustering. The performance of jointly learnt formulation [5] is better than the piecemeal technique [4]. In [6-8], K-means clustering is embedded in the stacked autoencoder and trained in joint end-to-end fashion. The distance metric used in K-means clustering is the Student's t-distribution kernel in [6] while the standard Euclidean distance in [7, 8]. In [9], the stacked autoencoder is replaced with the convolutional autoencoder. In [10], the autoencoder is embedded with the spectral clustering loss.

In [11], deep matrix factorization, which is deep dictionary learning with ReLU activation, is proposed and argued that different layers correspond to different concepts in the data.

A recent work [12] claims improvement over [8] by using hierarchical K-means; the authors of [12] claim that such a scheme improves robustness. Another work that proposed a minor variation to the previous autoencoder based schemes is [13]; instead of a simple autoencoder, they proposed using a contractive autoencoder.

We have mentioned before that existing studies that proposed embedding clustering losses in representation learning, were based on autoencoders. The only study that overcomes this issue (over-fitting in autoencoders) is [14]; they only need to learn the encoder network. The said study manages to bypass the trivial solution by imposing constraints on the learnt encoder network.

We have focused on deep learning based clustering papers that are algorithmic in nature.

2.2. Deep Convolutional Transform Learning

In convolutional transform learning (CTL) [2] a set of filters are learnt such that when operated on the data they produce the corresponding representations. Formally this is expressed as

$$t_m * x^{(k)} = z_m^{(k)}, \forall m \in \{1, .., M\} and \forall k \in \{1, .., K\} \quad (1)$$

Here $x^{(k)}$ denotes the k^{th} input sample, t_m the m^{th} convolutional filter and $z_m{}^{(k)}$ the representation of k^{th} sample after applying the m^{th} convolutional filter. There are a total of K samples and M filters. The symbol '*' denotes a convolutional operation with zero padding.

During training, the filters and the representations are learnt by solving the following optimization problem.

$$\min_{(t_m)_m, (z_m^{(k)})_{m,k}} \sum_{k=1}^{K} \sum_{m=1}^{M} (\|t_m * x^{(k)} - z_m^{(k)}\|_2^2 + \psi(z_m^{(k)})) + \lambda\{\|T\|_F^2 - logdet(T)\} \quad (2)$$

Here λ is a positive constant. T is defined as $T = [t_1|...|t_m]$. The log determinant term prevents the trivial solution t_m=0, $z_m{}^{(k)}$=0 and also promotes linear independence among the learnt filters; the other penalty $\|T\|_F^2$ prevents degenerate solutions where $t_m \to \infty, z_m^{(k)} \to \infty$. The penalty on T is borrowed from transform learning [15].

One can see how it is possible to learn convolutional filters from training data in an unsupervised fashion. CNNs do not have the said penalty on the learnt filters and hence can end up at the trivial solution. Furthermore, without the penalty on T, there is no guarantee that CNNs will learn unique filters.

In matrix-vector form, (2) can be expressed as follows

$$\min_{T,Z} \|T \bullet X - Z\|_F^2 + \psi(X) + \lambda\{\|T\|_F^2 - logdet(T)\} \quad (3)$$

where $X = [x^1|...|x^K]$, $Z = [z_1{}^{(k)}|...|z_M{}^{(k)}]_{1 \le k \le K}$,

$T \bullet X = \begin{bmatrix} t_1 * x^{(1)} & ... & t_M * z^{(1)} \\ ... & ... & ... \\ t_1 * x^{(K)} & ... & t_M * z^{(K)} \end{bmatrix}$ and ψ amounts to applying the penalty term column-wise on matrix X and summing.

In [3], a deeper extension of the unsupervised formulation was proposed leading to deep convolutional transform learning (DCTL). As the name suggests, in the said formulation, multiple convolutional filters were being applied on the samples one after the other to generate the representation. This is expressed as follows,

$$\min_{T_1,T_2,T_3,Z} \|T_3 \bullet (T_2 \bullet (T_1 \bullet X)) - Z\|_F^2 + \psi(Z) + \lambda \sum_{i=1}^{3} \{\|T_i\|_F^2 - logdet(T_i)\} \quad (4)$$

Here, T_i refers to the i^{th} layer of convolutional filters. In each layer, the T_i is formed by stacking the filters as columns of a matrix. The formulation (4) is shown for three layers but can be extended further.

3. PROPOSED FORMULATION

The schematic diagram of the proposed architecture is shown in Fig. 1. We embed a K-means clustering loss into the DCTL formulation. The image is convolved by a set of convolutional filters (in T1). The resulting feature map is max-pooled followed by a scaled exponential linear unit (SELU) activation function. This constitutes the first stage of the convolutional layer. The thus obtained feature map is convolved through the second set of convolutional filters (T2). The resulting feature map undergoes max-pooling before being input to K-means clustering.

A piecemeal solution where the features were learnt separately via CTL / DCTL and passed onto K-means clustering separately was shown in [2, 3]. In this work, we propose to train the clustering embedded DCTL as a single optimization problem. Mathematically the expression is shown as:

$$\min_{T_1,T_2,Z,H} \|T_2 \bullet (T_1 \bullet X) - Z\|_F^2 + \psi(Z) + \lambda \sum_{i=1}^{3} \{\|T_i\|_F^2 - logdet(T_i)\} + \mu \|Z - ZH^T(HH^T)^{-1}H\|_F^2 \quad (5)$$

Here, H is is the matrix of binary indicator variables h_{ij}; h_{ij}=1 if $x_j \in$ cluster i and 0 otherwise. Problem (5) is solved iteratively in two parts. In the first part, the H is assumed to be constant and T1, T2, Z are updated -

$$P1 : \min_{T_1,T_2,Z} \|T_2 \bullet (T_1 \bullet X) - Z\|_F^2 + \psi(Z) + \lambda \sum_{i=1}^{3} \{\|T_i\|_F^2 - logdet(T_i)\} + \mu \|Z - ZH^T(HH^T)^{-1}H\|_F^2 \quad (6)$$

In the second part, T1, T2, Z are assumed to be fixed and H is updated –

$$P2 : \min_{H} \|Z - ZH^T(HH^T)^{-1}H\|_F^2 \quad (7)$$

The first part (P1) is solved via ADAM optimizer and the second part (P2) is solved via K-means clustering. Solutions to P1 and P2 are carried out alternately till convergence. By convergence, we mean a condition when the cluster centers do not vary significantly in subsequent iterations.

4. EXPERIMENTAL RESULTS

We evaluate our algorithm on three well known image databases. They are YaleB [20], Extended YaleB [21] and AR Faces [22]. The YaleB consists of 5760 images of 10 different subjects; each under 576 different lighting conditions. The Extended YaleB contains 16128 images of 28 subjects under different poses and illumination conditions. The AR Faces contains over 4000 images of 126 different subjects. For all the datasets the dense shift invariant feature transform (DSIFT) features were first extracted; then principal component analysis (PCA) was used to further reduce the dimensions to 300. This protocol was followed by several prior clustering studies [16-19].

We compared our proposed formulation with several benchmarks including Deep Learning friendly Clustering (DLC) [7], Deep K-Means (DKM) [8], Deep Clustering with Convolutional Autoencoder (DCEC) [9] and AutoEncoded K-Means (AEKM) [4]. We have also used the K-means algorithm with standard Euclidean distance as distance metric for comparison.

Normalized Mutual Information (NMI), Adjusted Rand Index (ARI) and Accuracy are used as metrics [5, 14]. We have used μ=1 and λ=.001 in all the experiments. For all the experiments, 3 filters of sizes 9x9 have been used in both the first and second layer of convolutions. The max-pooling kernel size is 2x2. The results are shown in Table 1. For YaleB and AR Faces our method yields the best results; for Extended YaleB our results are a close second. The YaleB and the AR Face datasets are more challenging compared to Extended YaleB. This is because the first two have a larger number of clusters (an order of magnitude higher than Extended YaleB) and fewer images (an order of magnitude lower than Extended YaleB). On these two challenging datasets, we do better than the existing benchmarks. In the relatively simpler case (Extended YaleB) we are doing slightly worse than DCEC in terms of ARI and NMI but is better in terms of accuracy. It is interesting to note that, existing deep learning algorithms are doing worse than the simple K-means for the more challenging datasets.

The experiments were run on a 64 bit Intel i5 clocked at 1.6 GHz with 16GB of RAM. The operating system is Ubuntu. All the algorithms were run on Python. The runtime (in seconds) for various techniques are shown in Table 1 under Time column. Unsurprisingly K-means is the fastest, our proposed method is in the same order as that of K-means and is much faster than the rest of the deep learning based clustering techniques.

Fig. 1. Schematic diagram of the proposed approach

Table 1. Clustering Results

	YaleB				Extended YaleB				ARFaces			
Models	Accuracy	NMI	ARI	Time	Accuracy	NMI	ARI	Time	Accuracy	NMI	ARI	Time
K-Means	0.618	0.669	0.448	**24**	0.098	0.131	0.014	**1769**	0.146	0.457	0.047	**377**
DKM	0.338	0.430	0.158	397	0.087	0.125	0.010	7789	0.133	0.449	0.042	1414
DLC	0.386	0.477	0.186	428	0.098	0.151	0.016	5436	0.138	0.455	0.045	1418
AEKM	0.376	0.462	0.169	124	0.103	0.103	0.018	2536	0.137	0.456	0.045	522
DCEC	0.539	0.624	0.377	2112	0.295	**0.453**	**0.173**	24367	0.074	0.26	0.02	7008
Proposed	**0.649**	**0.708**	**0.510**	51	**0.349**	0.448	0.132	2417	**0.159**	**0.463**	**0.051**	757

Table 2. Results From Ablation Studies

	YaleB				Extended YaleB				ARFaces			
Metric	Prop1L	Piece1L	Prop2L	Piece2L	Prop1L	Piece1L	Prop2L	Piece2L	Prop1L	Piece1L	Prop2L	Piece2L
Accuracy	0.620	0.588	**0.649**	0.636	0.146	0.135	**0.349**	0.320	0.142	0.122	**0.159**	0.151
NMI	0.686	0.648	**0.708**	0.691	0.204	0.193	**0.448**	0.432	0.453	0.434	**0.463**	0.456
ARI	0.447	0.412	**0.510**	0.462	0.033	0.031	**0.132**	0.123	0.044	0.040	**0.051**	0.048

4.1. Ablation Studies

In this section, we show how the results vary with the number of layers. We also show how the results vary when the problem is solved jointly (as proposed) versus the piecemeal solution; by piecemeal we mean features are generated from (deep) convolutional transform learning and the features are separately fed into K-means clustering. The results are shown in Table 2. The joint solution, be it one layer or two-layer, yields better results than the piecemeal solution. This is expected; even in the past, jointly formulated solutions yielded better results than piecemeal ones. For both the piecemeal and joint solutions, going deeper helps, that is, the results obtained from two layers are always better than the ones from one layer.

Finally we show the empirical convergence plot in Fig. 2. We find that the proposed algorithm converges within 50 iterations. The convergence plot for other depths show a similar trend.

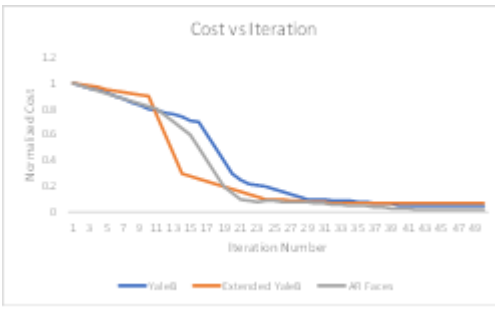

Fig. 2. Empirical Convergence Plot

5. CONCLUSION

We have proposed a convolutional representation learning based clustering formulation that does not require learning additional decoder/deconvolutional filters as is required by all existing approaches. This results in the requirement of learning fewer parameters, which in turn is less prone to over-fitting and hence can cluster from fewer data. The improvement in results can be seen in the experiments. Our method achieves higher clustering scores with fewer samples and a larger number of clusters compared to the state-of-the-art deep learning techniques compared against.

In the future, we would like to embed popular clustering techniques like subspace clustering [23], fuzzy C-means clustering [24] and hierarchical clustering [25] into the framework.

6. ACKNOWLEDGEMENT

E.C. and G.C. acknowledge support from the Agence Nationale de la Recherche of France under MAJIC (ANR-17-CE40-0004-01) project.

7. REFERENCES

[1] C. Bauckhage, "K-means clustering is matrix factorization," arXiv preprint arXiv:1512.07548, 2015.

[2] J. Maggu, E. Chouzenoux, G. Chierchia and A. Majumdar, "Convolutional Transform Learning", ICONIP, pp. 162-174, 2018.

[3] J. Maggu, E. Chouzenoux, G. Chierchia and A. Majumdar, "Deep Convolutional Transform Learning", ICONIP, pp. 300-307, 2020.

[4] F. Tian, B. Gao, Q. Cui, E. Chen and T. Y. Liu, "Learning deep representations for graph clustering," 28th AAAI conference on Artificial Intelligence, pp. 1293-1299, 2014.

[5] X. Peng, S. Xiao, J. Feng, W. Y. Yau and Z. Yi, "Deep Sub-space Clustering with Sparsity Prior," International Joint Conference on Artificial Intelligence, pp. 1925-1931, 2016.

[6] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," International Conference on Machine Learning, pp. 478-487, 2016.

[7] B. Yang, X. Fu, N. D. Sidiropoulos and M. Hong, "Towards k-means-friendly spaces: Simultaneous deep learning and clustering," International Conference on Machine Learning, pp. 3861-3870, 2017.

[8] M. M. Fard, T. Thonet and E. Gaussier, "Deep k-means: Jointly clustering with k-means and learning representations," Pattern Recognition Letters, vol. 138, pp.185-192, 2020.

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

[10] X. Yang, C. Deng, F. Zheng, J. Yan and W. Liu, "Deep Spectral Clustering Using Dual Autoencoder Network," IEEE Conference on Computer Vision and Pattern Recognition, pp. 4061-4070, 2019.

[11] G. Trigeorgis, K. Bousmalis, S. Zafeiriou and B. W. Schuller, "A Deep Matrix Factorization Method for Learning Attribute Representations," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 3, pp. 417-429, 2017.

[12] S. Huang, Z. Kang, Z. Xu and Q. Liu, "Robust Deep K-Means: An Effective and Simple Method for Data Clustering," Pattern Recognition, p.107996, 2021.

[13] B. Diallo, J. Hu, T. Li, G. A. Khan, X. Liang and Y. Zhao, "Deep embedding clustering based on contractive autoencod-er," Neurocomputing, 433, pp. 96-107, 2021.

[14] X. Peng, J. Feng, J. T. Zhou, Y. Lei and S. Yan, "Deep Subspace Clustering," IEEE Transactions on Neural Networks and Learning Systems, vol. 31, no. 12, pp. 5509-5521, Dec. 2020.

[15] S. Ravishankar, B. Wen and Y. Bresler, "Online Sparsifying Transform Learning—Part I: Algorithms," in IEEE Journal of Selected Topics in Signal Processing, vol. 9, no. 4, pp. 625-636, June 2015.

[16] Y. Chen, G. Li and Y. Gu, "Active Orthogonal Matching Pur-suit for Sparse Subspace Clustering," IEEE Signal Processing Letters, vol. 25, no. 2, pp. 164-168, 2018.

[17] X. Peng, S. Xiao, J. Feng, W. Y. Yau and Z. Yi, "Deep Subspace Clustering with Sparsity Prior," IJCAI, pp. 1925-1931, 2016.

[18] J. Maggu, A. Majumdar and E. Chouzenoux, "Transformed Subspace Clustering", IEEE Transactions on Knowledge and Data Engineering, vol. 33, no. 4, pp. 1796-1801, 1 2021.

[19] J. Maggu, A. Majumdar, E. Chouzenoux and G. Chierchia, "Deeply Transformed Subspace Clustering", Signal Processing, vol. 174, 107628, 2020.

[20] `http://vision.ucsd.edu/~leekc/ExtYaleDatabase/Yale%20Face%20Database.htm`

[21] `http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html`

[22] `https://www2.ece.ohio\protect\discretionary{\char\hyphenchar\font}{}{}state.edu/~aleix/ARdatabase.html`

[23] R. Vidal, "Subspace Clustering," in IEEE Signal Processing Magazine, vol. 28, no. 2, pp. 52-68, March 2011.

[24] J. C. Bezdek, R. Ehrlich and W. Full, "FCM: The fuzzy c-means clustering algorithm," Computers & geosciences, vol. 10, no. 2-3, pp.191-203, 1984.

[25] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: an overview," Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 2, no. 1, pp.86-97, 2012.