Supplementary Material: Deep Comprehensive Correlation Mining for Image Clustering

Jianlong Wu[1*] Keyu Long[2*] Fei Wang[2] Chen Qian[2] Cheng Li[2] Zhouchen Lin[3,✉] Hongbin Zha[3]
[1]School of Computer Science and Technology, Shandong University
[2]SenseTime Research
[3]Key Laboratory of Machine Perception (MOE), School of EECS, Peking University
jlwu1992@sdu.edu.cn, corylky114@gmail.com, {wangfei, qianchen, chengli}@sensetime.com, zlin@pku.edu.cn, zha@cis.pku.edu.cn

Abstract

This is the supplementary material for the paper entitled "Deep Comprehensive Correlation Mining For Image Clustering". The organization of the supplementary material is listed as follows. We first present the proof of Lemma 1 and Claim 1 in Section 1. Then we give the detailed definition of evaluation metrics used in experiments in Section 2. Section 3 presents the details of these compared methods. Section 4 shows the architectures for different datasets. Finally, in Section 5, we demonstrate the influence of sampling strategy for triplet mutual information computing.[1]

1. Proof of Lemma 1 and Claim 1

Proof of Lemma 1: Since $\omega(e_i) \neq \omega(e_j)$ for $\forall i \neq j$, there exists a strongly increasing sequence of weights $\{\omega_1, \omega_2, \cdots, \omega_{\frac{N(N+1)}{2}}\}$, and we can remove edges from G in the order from smallest weight to largest by increasing threshold t. This action would either increase the current partition number n to $n+1$ or remain it unchanged. At the beginning of the process we have 1 partition and at the end of the process we have N partitions. Since $1 \leq K \leq N$, there exists a K partition in the process.

□

Proof of Claim 1: Select samples $\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^K$ from partition $P^1, P^2, \cdots, P^K$, denote the cosine similarity matrix of their corresponding optimal features $f_{\theta*}(\mathbf{x}_1), f_{\theta*}(\mathbf{x}_2), \cdots, f_{\theta*}(\mathbf{x}_K)$ as $\mathbf{S}$, and $\mathbf{S}$ equals to its K partitions pseudo graph $\mathbf{W}$, which is an identity matrix. Denote $f_{\theta*}(\mathbf{x}_i)$ as $[z_i^1, z_i^2, \cdots, z_i^K]$, where z_i^k denotes the k-th element of the vector $\mathbf{z}_i$.

The set $\{z_1^1, z_1^2, \cdots, z_1^K, \cdots, z_K^1, \cdots, z_K^K\}$ can only have no more than K positive elements, otherwise, according to Pigeonhole principle, there exists a k that $z_i^k = z_j^k$ and $\cos(\mathbf{z}_i, \mathbf{z}_j) > 0$, which is contradicted to $\mathbf{S}_{ij} = 0$.

On the other hand, for the output of a softmax layer, every vector has at least one positive entry. Therefore, every vector has and only has one positive element that equals to 1.

□

2. Defenitions of Metrics

We introduce the following three standarded metrics we used to evaluate our model:

- Normalized Mutual Information (NMI): Let C and C' denote the predicted partition and the ground truth partition respectively, the NMI metric is calculated as:

$$\mathrm{NMI}(C, C') = \frac{\sum_{i=1}^{K}\sum_{j=1}^{S} |C_i \cap C'_j| \log \frac{N|C_i \cap C'_j|}{|C_i||C'_j|}}{\sqrt{(\sum_{i=1}^{K} |C_i| \log \frac{C_i}{N})(\sum_{j=1}^{S} |C'_j| \log \frac{C'_j}{N})}}. \tag{1}$$

- Adjusted Rand Index (ARI): Given a set S of n elements, and two groupings or partitions (e.g. clustering results) of these elements with r and s groups, namely $X = \{X_1, X_2, \ldots, X_r\}$ and $Y = \{Y_1, Y_2, \ldots, Y_s\}$, the overlap between X and Y can be summarized in a contingency table $[c_{ij}]$, where each element c_{ij} denotes the number of objects in common between X_i and Y_j:

$$c_{ij} = |X_i \cap Y_j|. \tag{2}$$

The contingent table is of the following shape:

$X \backslash Y$	Y_1	Y_2	$\ldots$	Y_s	Sums
X_1	c_{11}	c_{12}	$\ldots$	c_{1s}	a_1
X_2	c_{21}	c_{22}	$\ldots$	c_{2s}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
X_r	c_{r1}	c_{r2}	$\ldots$	c_{rs}	a_r
Sums	b_1	b_2	$\ldots$	b_s	

[*]Equal contribution and the work was done during interns at SenseTime Research
[1]Project address: https://github.com/Cory-M/DCCM

Table 1. Network architecture for various datasets we used in experiments.

CIFAR-10 / CIFAR-100 $32\times32\times3$	Tiny-ImageNet $64\times64\times3$	ImageNet-10/ImageNet-dog-15/STL-10 $96\times96\times3$
3×3 conv. 64 BN ReLU (S) 3×3 conv. 64 BN ReLU 2×2 MaxPooling with stride 2 3×3 conv. 128 BN ReLU 2×2 MaxPooling with stride 2 3×3 conv. 256 BN ReLU 4×4 AvgPooling with stride 2	5×5 conv. 64 BN ReLU 5×5 conv. 64 BN ReLU 4×4 MaxPooling with stride 4 (S) 3×3 conv. 128 BN ReLU 3×3 conv. 128 BN ReLU 4×4 MaxPooling with stride 4 1×1 conv. 256 BN ReLU 2×2 AvgPooling with stride 2	5×5 conv. 64 BN ReLU 5×5 conv. 64 BN ReLU 4×4 MaxPooling with stride 4 (S) 3×3 conv. 128 with BN ReLU 3×3 conv. 128 BN ReLU 4×4 MaxPooling with stride 4 1×1 conv. 256 with BN ReLU 4×4 AvgPooling with stride 4
(D) Linear(256, 64) BN ReLU Linear(64, c) SoftMax	(D) Linear(256, 256) BN ReLU Linear(256, c) SoftMax	(D) Linear(256, 64) BN ReLU Linear(64, c) SoftMax

and ARI is defined by:

$$\mathrm{ARI} = \frac{\sum_{ij}\binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}]/\binom{n}{2}}. \tag{3}$$

- Accuracy (ACC): Suppose the clustering algorithm is tested on N samples. For a sample $\mathbf{x}_i$, we denote its cluster label as r_i and its ground truth as t_i. The clustering accuracy is defined by:

$$\mathrm{ACC}(R,T) = \frac{\sum_{i=1}^{N}\delta(t_i, \mathrm{map}(r_i))}{N}, \tag{4}$$

where

$$\delta(a,b) = \begin{cases} 1, & \text{if } a=b, \\ 0, & \text{otherwise}, \end{cases} \tag{5}$$

and function $\mathrm{map}(x)$ denotes the best permutation mapping function gained by Hungarian algorithm [3].

3. Compared Methods

For clustering, we adopt both traditional methods and deep learning based methods, including K-means, spectral clustering (SC) [22], agglomerative clustering (AC) [11], the nonnegative matrix factorization (NMF) based clustering [4], auto-encoder (AE) [1], denoising auto-encoder (DAE) [18], GAN [17], deconvolutional networks (DECNN) [21], variational auto-encoding (VAE) [14], deep embedding clustering (DEC) [19], jointly unsupervised learning (JULE) [20], and deep adaptive image clustering (DAC) [6].

For classification task, we compare DCCM against several unsupervised feature learning methods, including variational auto-encoder (VAE) [14], adversarial auto-encoder (AAE) [16], BiGAN [9], noise as targets (NAT) [2], and deep infomax (DIM) [12].

4. Architechtures Details

In Table 1, we present the architectures for different datasets.

For CIFAR-10/CIFAR-100 [15], we set 4 conv layers and 3 pooling layers, followed with 2 fully-connected layers. Batch Normalization [13] and ReLU are used on all hidden layers. The output features after the second conv layer (S for shallow) and the first fc layer (D for deep) are used to compute the mutual information (MI) loss, concatenated as the input of discriminator. For other datasets, such as Tiny-ImageNet [8] and STL-10 [7], we set 5 conv layers instead of 4. Due to their larger input size, we use the feature maps after the third conv layer as S. For all experiments, the output was a $class\_num$ dimensional vector.

5. Comparison Under the Same Architecture

In Table 2, we present the additional comparisons using the same network. On CIFAR-10/100, DeepCluster does not work well based on its released official PyTorch code. DAC has similar performance with that in their paper. Our DCCM achieves the best results.

Please note that we only use a simple shallow version of AlexNet in the paper, and our results are much better than the best reported results of other methods.

Besides, our algorithm is relatively efficient. On CIFAR-100, it costs 19 hours for training on a single GTX 1080Ti GPU. Multiple GPU cards and better GPU can improve this.

6. Sampling Strategy

The experiment result corresponding to the analysis in line 836-843 is listed in Table 3. We tried four strategies to fetch positive and negative pairs from pseudo-graph $\mathbf{W}$, and the terms used in the table refer to:

- nearest means that for each sample, we select its nearest sample from the minibatch to construct a positive

Table 2. Result comparison under the same architecture (except the last layer of RotNet) on CIFAR-10/100. '<' denotes 'less than'.

	CIFAR-10				CIFAR-100			
	NMI	Clustering ACC	ARI	Classify ACC	NMI	Clustering ACC	ARI	Classify ACC
RotNet [10]	0.316	0.389	0.139	0.755	0.208	0.225	0.070	0.453
DeepCluster [5]	<0.3	<0.3	<0.1	<0.75	<0.2	<0.2	<0.07	<0.45
DAC [6]	0.439	0.514	0.335	0.787	0.228	0.254	0.121	0.485
DCCM (ours)	**0.496**	**0.623**	**0.408**	**0.818**	**0.285**	**0.327**	**0.173**	**0.512**

Table 3. Classification accuracy of different pair-sampling strategies on CIFAR-10.

	Methods	Classification ACC(Y64)
V1	nearest pos + random* neg	0.744
V2	nearest pos + farthest neg	0.713
V3	random* pos + random* neg	0.731
V4	top-n pos + random* neg	0.698

pair, while farthest means taking the farthest one to construct a negative pair.

- random* means that we randomly take a positive sample that satisfies $W_{ij} = 1$ as a positive pair or a negative sample that satisfies $W_{ij} = 0$ as a negative pair.

- top-n pos means that we select the top n confident pairs from the graph W to construct positive pairs.

For each strategy, we take n positive pairs and n negative pairs into account, where n is our batch size. This is to make sure that the computational complexity of each approach is nearly the same for fair comparison, while we also have explored more costly approaches and find that the improvement is negligible.

To clearly illustrate how L_{MI} is effected, here we set a fixed model trained with only $\widehat{L_{\text{PG}}} + \widehat{L_{\text{PL}}}$. Then with the pseudo-graph $\mathbf{W}$ generated by it, we train a new model using only L_{MI} from scratch. It can be concluded that the positive pairs are sensitive to noise since strategy V1 achieves better results than V3, and harder negative pairs are beneficial for training as strategy V1 also achieves better results than V2. Besides, we also notice the importance of uniform sampling within the minibatch, as the top-n pairs in V4 has higher confidence than that in V1, but the training collapses since only part of samples in the batch are included in the top-n strategy.

References

[1] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *NIPS*, pages 153–160, 2007.

[2] P. Bojanowski and A. Joulin. Unsupervised learning by predicting noise. In *ICML*, pages 517–526, 2017.

[3] D. Cai, X. He, and J. Han. Document clustering using locality preserving indexing. *IEEE Transactions on Knowledge and Data Engineering*, 17(12):1624–1637, 2005.

[4] D. Cai, X. He, X. Wang, H. Bao, and J. Han. Locality preserving nonnegative matrix factorization. In *IJCAI*, 2009.

[5] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018.

[6] J. Chang, L. Wang, G. Meng, S. Xiang, and C. Pan. Deep adaptive image clustering. In *IEEE ICCV*, pages 5879–5887, 2017.

[7] A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, pages 215–223, 2011.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE CVPR*, 2009.

[9] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. In *ICLR*, 2017.

[10] S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018.

[11] K. C. Gowda and G. Krishna. Agglomerative clustering using the concept of mutual nearest neighbourhood. *Pattern recognition*, 10(2):105–112, 1978.

[12] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, A. Trischler, and Y. Bengio. Learning deep representations by mutual information estimation and maximization. In *ICLR*, 2019.

[13] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

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

[15] A. Krizhevsky, V. Nair, and G. Hinton. Cifar-10 and cifar-100 datasets. *URl: https://www. cs. toronto. edu/kriz/cifar. html*, 6, 2009.

[16] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[17] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[18] 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*, 11(Dec):3371–3408, 2010.

[19] J. Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, pages 478–487, 2016.

[20] J. Yang, D. Parikh, and D. Batra. Joint unsupervised learning of deep representations and image clusters. In *IEEE CVPR*, pages 5147–5156, 2016.

[21] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *IEEE CVPR*, pages 2528–2535, 2010.

[22] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *NIPS*, pages 1601–1608, 2005.