Locally Normalized Soft Contrastive Clustering for Compact Clusters

Xin Ma[1] and **Won Hwa Kim**[1,2,3]
[1]Computer Science and Engineering, University of Texas at Arlington, USA
[2]Computer Science and Engineering, POSTECH, South Korea
[3]Graduate School of AI, POSTECH, South Korea
xin.ma@mavs.uta.edu, wonhwa@postech.ac.kr

Abstract

Recent deep clustering algorithms take advantage of self-supervised learning and self-training techniques to map the original data into a latent space, where the data embedding and clustering assignment can be jointly optimized. However, as many recent datasets are enormous and noisy, getting a clear boundary between different clusters is challenging with existing methods that mainly focus on contracting similar samples together and overlooking samples near boundary of clusters in the latent space. In this regard, we propose an end-to-end deep clustering algorithm, i.e., Locally Normalized Soft Contrastive Clustering (LNSCC). It takes advantage of similarities among each sample's local neighborhood and globally disconnected samples to leverage positiveness and negativeness of sample pairs in a contrastive way to separate different clusters. Experimental results on various datasets illustrate that our proposed approach achieves outstanding clustering performance over most of the state-of-the-art clustering methods for both image and non-image data even without convolution.

1 Introduction

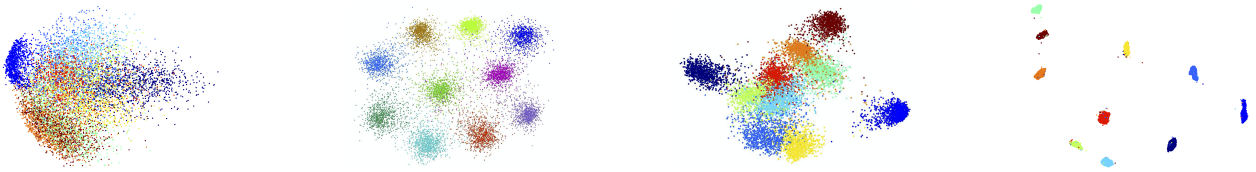
Clustering aims to separate scattered N data samples $\mathcal{X} = \{\mathbf{X}_i\}_{i=1}^{N}$ in a feature space into different groups (e.g., K number of clusters) in an unsupervised manner. In general, the priority is to gather the samples within the same group close and make the samples across different clusters distinct from each other. As a fundamental topic in machine learning, clustering has played critical roles in a broad range of fields including gene sequence clustering in bioinformatics [Petegrosso *et al.*, 2020; Zou *et al.*, 2020], creation of perfectionism profiles in social science [Bolin *et al.*, 2014], unsupervised image segmentation [Kanezaki, 2018; Ji *et al.*, 2019], document clustering in information retrieval [Xu *et al.*, 2003; Fard *et al.*, 2020; Costa and Ortale, 2020] and etc.

Traditional methods such as k-means [MacQueen and others, 1967], DenPeak [Rodriguez and Laio, 2014], DBSCAN [Ester *et al.*, 1996], and Spectral Clustering [Zelnik-Manor and Perona, 2005; Chen and Cai, 2011] have been effective when datasets used to be relatively small. As recent datasets become larger in both size and dimension, traditional shallow methods suffer from high computational complexity. Various Deep Clustering (DC) techniques have been recently developed to cope with issues of conventional methods. The core of DC consists of two components: 1) Dimension Reduction with Deep Learning (e.g., autoencoder, self-supervised learning) for mapping high-dimensional data onto a low-dimensional space and 2) Self-training the low-dimensional embedding to further improve clustering results from traditional clustering algorithms such as k-means [Xie *et al.*, 2016; Ghasedi Dizaji *et al.*, 2017].

The premise behind the DC algorithms is that suitable low-dimensional embeddings and cluster centers will assign each sample to a vivid individual cluster. These techniques focus on optimizing 'cluster assignment probability' — a likelihood of a sample belonging to each cluster. Many DC methods minimize Kullback–Leibler (KL) divergence between the distribution of the cluster assignment probability and an auxiliary target distribution directly computed from it, or directly use Fully Connected (FC) Layers to predict the cluster assignment probability. This process makes the assignment probability localized to a single cluster for each sample.

Unfortunately, many DC approaches suffer from two major bottlenecks. First, the initial clusters usually computed from a centroid-based clustering algorithm with random initialization, e.g., k-means, often cause stability issues when the low-dimensional embedding is not sufficiently effective for clustering. This problem can be restrictively relieved for image data using self-supervised learning (e.g., SimCLR, MoCo, SwAV) to learn a better feature space for the initial clustering [Van Gansbeke *et al.*, 2020; Caron *et al.*, 2020], but not for other types of data. The second issue, perhaps even more critical, stems from the self-training process and FC layers where the update of cluster assignment probability is mainly focused on clustering easy samples close to the centers and overlooks the samples near the "boundary" of clusters.

In fact, for clustering, what matters over the variation in the data is the relationships between data samples (i.e., structure) typically represented as a graph, e.g., k-nearest neighbor (kNN) graph. Also, a sound clustering method should provide compact clusters that are distinct from each other. This can be done by pulling similar samples together and pushing dissimilar samples apart in a latent space in a contrastive way.



(a) Raw Data. (b) JULE. (c) DEPICT. (d) LNSCC (Ours).

Figure 1: Visualization of embedding in subspaces optimized by different methods on MNIST-test dataset. (a) The raw data in 2D with PCA. (b) The embedding subspace of JULE [Yang *et al.*, 2016]. (c) The embedding subspace of joint DEPICT [Ghasedi Dizaji *et al.*, 2017]. (d) The embedding subspace of LNSCC by exploring the relationships among samples in a contrastive way.

While many contrastive learning methods have been successful, they mainly work on identifying similar sample pairs (i.e., positive pairs) for optimization. However, focusing only on the positive pairs, a.k.a., "hard contrastive learning", may not be effective in practice as seen in k-Nearest Neighbors classification on ImageNet [Van Gansbeke *et al.*, 2020] when the data are in high-dimensional space with large variations.

In this regards, for effective compact clustering, we propose the following hypothesis: the method should be able to 1) accurately separate samples near cluster boundaries where the similarities among the neighborhood have large variation, 2) compactly contract similar samples near cluster centers, 3) completely disconnected samples should be expanded far apart. For this, we propose an end-to-end clustering algorithm that implements our ideas above in three different phases. The key is to consider both positiveness and negativeness for each pair of samples together in a "soft contrastive" way rather than the hard contrastive scheme. Our work demonstrates the following **contributions:**

- **(i)** we propose a novel method "Locally Normalized Soft Contrastive Clustering" (LNSCC) that achieve *very compact clusters* by introducing the concept of contrastive learning to the clustering task,
- **(ii)** LNSCC performs dimension reduction, self-training and clustering simultaneously as a unified framework,
- **(iii)** we carry extensive empirical validation of LNSCC with various independent datasets, demonstrating competitive qualitative and quantitative performances.

Fig. 1 (d) is a teasing result from LNSCC achieving localized clusters in a learnt 2D space using high-dimensional MNIST-test data, and its details are introduced in the following.

2 Related Work

Deep Clustering introduces deep learning into clustering to learn effective representations and cluster assignments [Guo *et al.*, 2019; Xie *et al.*, 2016; Yang *et al.*, 2019; Huang *et al.*, 2020]. In [Ghasedi Dizaji *et al.*, 2017], a denoised autoencoder was used to further improve the low-dimensional embedding and increased clustering performance. Some DC methods demonstrated more powerful results with data augmentation. SCAN [Van Gansbeke *et al.*, 2020], GCC [Zhong *et al.*, 2021], SwAV [Caron *et al.*, 2020] and MiCE [Tsai *et al.*, 2020] recently achieved state-of-the-art clustering performance on image datasets using contrastive learning. SCAN takes advantage of the nearest neighbors and proposes that the sample and its nearest neighbors should be assigned to the same cluster, and GCC assumes that the transform of an image and its neighbors' transformer should also be similar further improving clustering performance on image data.

Unlike other DC algorithms which treat nearest neighbors of each sample equally as similar pairs, we take advantage of the local structure (i.e., the distribution of neighbors) of each sample in a dataset with both positiveness (similar) and negativeness (dissimilar/disconnected) with soft contrastive learning. Moreover, the input of our framework is not limit to images, and the good features learned by self-supervised models (e.g., SimCLR, MoCo) can also be utilized to boost the clustering performance further.

3 LNSCC: Locally Normalized Soft Contrastive Clustering

Given a dataset $\mathcal{X} = \{\mathbf{X}_i \in \mathbb{R}^D\}_{i=1}^N$ with N samples in D-dimensional space, the principle of clustering is to separate $\mathcal{X}$ into K clusters such that intra-cluster samples stay compact while inter-cluster samples stay far apart. LNSCC aims to obtain compact clusters with a low-dimensional embedding $\mathcal{Z}$ by mining the relationships between samples via positiveness and negativeness of each sample pair for clustering.

3.1 Locally Normalized Soft Contrastive Learning

Contrastive learning is a representation learning technique that aims at learning good representations by contracting positive (i.e., similar) pairs and expanding negative (i.e., different) pairs without labels. The key is to discriminate positive sample pairs: SimCLR [Chen *et al.*, 2020] considers positive sample pairs in a latent space, and SCAN [Van Gansbeke *et al.*, 2020] and GCC [Zhong *et al.*, 2021] discover positive sample pairs using a kNN graph. In all those methods, discriminating good positive sample pairs is known as "Hard Contrastive Learning". However, the assumption that a sample and its neighbors belong to the same category may not be true as shown in the misleading k-NN classification on ImageNet [Van Gansbeke *et al.*, 2020], which indicates that the neighbors of each sample should be considered separately.

Unlike the previous contrastive learning which considers positive pairs only, in this paper, we consider both positive and negative sample pairs jointly. We explore both positive and negative relations among the neighbors of each sample and propose a locally normalized soft contrastive learning (LNSCL) approach with two components: *Locally Normalized Structure* and *Soft Contrastive Assignment*. The former builds a locally normalized kNN graph via three phases $\mathcal{P}$:

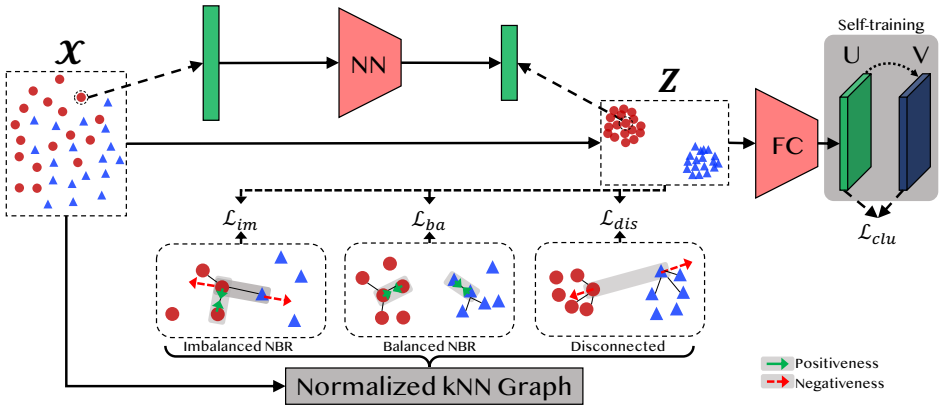


Figure 2: The overview of LNSCC model. Two models are trained in an end-to-end scheme: Neural Network (NN) that embeds $\mathcal{X}$ to $\mathcal{Z}$ and Fully Connected (FC) layer that assigns clusters. Loss functions for different phases for contrastive learning (in Section3.2) are defined using similarities and connectivity of samples.

- $\mathcal{P}_{im}$: focusing on samples with a skewed distribution of similarities among its neighbors with high variance (i.e., imbalanced neighbors) near cluster boundaries,
- $\mathcal{P}_{ba}$: focusing on samples with similarities among its neighbors close to uniform distribution (i.e., balanced neighbors) near cluster centers,
- $\mathcal{P}_{dis}$: focusing on disconnected samples,

and the overview of the framework including the ideas of the three $\mathcal{P}$ is shown in Fig. 2. The latter assigns both positiveness and negativeness scores instead of choosing either positive or negative for each sample pair by exploring the structure of locally normalized kNN.

Locally Normalized Structure. A naive weighted kNN often has large edge weights (i.e., high similarities) among samples in densely populated regions mostly near the center of sample clusters, and small weights among samples near cluster boundaries where samples exist sparsely (e.g., **E** and **F** in Fig. 3 respectively). Because of such tendency, a downstream clustering algorithm mostly focuses on the already well compactly grouped samples with large edge weights and ignores the samples sparsely distributed at the boundary of clusters. To fairly separate the samples at the cluster boundary, we introduce a locally normalized kNN graph which normalizes the neighbors' weights at each sample to fully take advantage of neighbor structure of each sample independently.

Consider a distance matrix $\mathbf{D}_{N\times M}$ computed from $\mathcal{X}$ where M is the predefined number of nearest neighbors. The elements in each row $\mathbf{D}_{i:}, i \in [1, 2, ..., N]$ of $\mathbf{D}$ are distance metrics (e.g., Euclidean distance) of the i-th sample to its M different nearest neighbors (we use M instead of k to avoid confusion with the number of clusters K). To normalize the neighbors at each sample, the softmax function is used at each row of $\mathbf{D}$ to achieve a normalized weight matrix $\mathbf{W}$.

$$\mathbf{W}_{i:} = \text{softmax}(-\mathbf{D}_{i:}). \tag{1}$$

Based on $\mathbf{W}$, a directed weight matrix $\hat{\mathbf{W}}_{N\times N}$ can be derived, and a symmetric adjacency matrix $\mathbf{A}$ is defined as

$$\mathbf{A}_{ij} = \hat{\mathbf{W}}_{ij} + \hat{\mathbf{W}}_{ji} - \hat{\mathbf{W}}_{ij}\hat{\mathbf{W}}_{ji}, \tag{2}$$

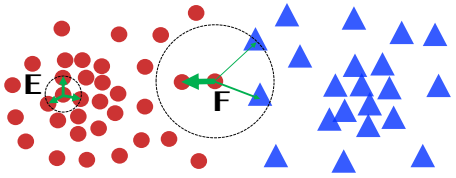


Figure 3: Illustration of edge weight distribution of samples with imbalanced neighbors (i.e., **F**) and balanced neighbors (i.e., **E**) in a locally normalized 3-NN graph. Line width denotes the edge weight.

which defines edge weights of a latent graph $\mathcal{G}_{lg}$. The $\mathbf{A}$ imputes edges that are one-sided in $\mathbf{W}$ and makes the separation of cluster boundaries more effective.

In a latent graph $\mathcal{G}_{lg}$, small edge weights typically exist on the samples with neighbors with large variations in their similarities as they are normalized per sample. At the central region of each cluster, samples stay close to each other and the weights in their sub-graphs have low variation. However, at the cluster boundaries, samples are sparsely distributed and it is highly likely to obtain imbalanced sub-graphs with high average and variations in their distances. Fig. 3 illustrates such behavior, where samples around **F** are sparse with different similarities (edge thickness) while samples around an interior point **E** are populated with low variation in their similarities.

Soft Contrastive Assignment. Instead of discriminating positive and negative pairs as in other works, we assign positiveness and negativeness scores for each sample pair to represent the similarity and dissimilarity. We observed that connected samples in a kNN graph do not always belong to the same category, and the difference between positive pairs and negative pairs may not be clear in a naive kNN.

We propose a soft contrastive assignment scheme which derives positiveness score $\mathbf{s}^{+}_{ij}$ and negativeness score $\mathbf{s}^{-}_{ij}$ to quantify the relationships between samples $\mathbf{X}_i$ and $\mathbf{X}_j$.

$$\mathbf{s}^{+}_{ij} = \phi_{\mathcal{P}}(\mathbf{A}_{ij}) \text{ and } \mathbf{s}^{-}_{ij} = \varphi_{\mathcal{P}}(\mathbf{A}_{ij}), \tag{3}$$

where $\phi_{\mathcal{P}}(\cdot)$ and $\varphi_{\mathcal{P}}(\cdot)$ are score functions with a clustering rule that similar samples have large positiveness score while

dissimilar samples have with large negativeness score, and $\mathcal{P} \in \{\mathcal{P}_{im}, \mathcal{P}_{ba}, \mathcal{P}_{dis}\}$ represents different learning phases.

Batch-wise Training with Over/Under Sampling. Operating directly on all edges in a graph for training a model can be computationally challenging since the number of edges dramatically increases along with the number of samples. For efficient training, we take traditional batch-wise training by randomly selecting a batch of sample pairs. As seen in Fig. 2, there are two Networks to be trained using this batch training scheme, where the Neural Network (NN) is a backbone network used to obtain embedding $\mathcal{Z}$ of $\mathcal{X}$ and FC is used to obtain localized clustering assignment matrix $\mathbf{U}$ from $\mathcal{Z}$.

Also, instead of directly defining ϕ and φ, we create over- and under-sampled datasets whose elements are augmented sample pairs according to the ideas of ϕ and φ depending on $\mathbf{A}$. Different datasets are curated for each $\mathcal{P}_{im}, \mathcal{P}_{ba}, \mathcal{P}_{dis}$ such that the notion of $\mathbf{s}_{ij}^{+}$ and $\mathbf{s}_{ij}^{-}$ are inherited as the proportion of desirable and undesirable sample pairs for different phases. This is done by designing a quantity augmentation (QA) matrix $\mathbf{Q}$ as a function of $\mathbf{A}$ (and thereby $\mathbf{s}^{+}$ and $\mathbf{s}^{-}$) that determines how many times each sample pair should be included in the augmented dataset. This way, we indirectly incorporate the notion of the scores.

3.2 Learning Clustering-oriented Latent Space

Let $\mathcal{Z}$ be an unknown embedding of samples in a low-dimensional latent space. Given two samples $\mathbf{Z}_i$ and $\mathbf{Z}_j$ from $\mathcal{Z}$, their similarity $\mathbf{O}_{ij}$ in the embedding space should be antidependent on their distance $d(\mathbf{Z}_i, \mathbf{Z}_j)$, and we use an exponential function to define $\mathbf{O}_{ij}$ as

$$\mathbf{O}_{ij} = e^{-d(\mathbf{Z}_i, \mathbf{Z}_j)}. \tag{4}$$

Here, two close samples with small $d(\mathbf{Z}_i, \mathbf{Z}_j)$ lead to a large $\mathbf{O}_{ij}$ which indicates that "attractive force" between those two samples is strong in the latent space, and vice versa for two far samples. This $\mathbf{O}_{ij}$ will be used to define loss functions together with an augmented dataset using $\mathbf{A}$.

$\mathcal{P}_{im}$: Focusing on Samples with Imbalanced Neighbors

The goal of $\mathcal{P}_{im}$ is to attract each sample to its similar neighbors and expand it from the farther neighbors, and the contrastive assignments are given as

$$\mathbf{s}_{ij}^{+} = \mathbf{A}_{ij} \text{ and } \mathbf{s}_{ij}^{-} = 1 - \mathbf{A}_{ij}. \tag{5}$$

Given the observation that the samples with imbalanced neighbors are highly associated with small edge weights which are near cluster boundaries, we use the following QA matrices, $\mathbf{Q}^{im+}$ and $\mathbf{Q}^{im-}$, to encode the above $\mathbf{s}^{+}$ and $\mathbf{s}^{-}$:

$$\mathbf{Q}_{ij}^{im+} = \left\lfloor \frac{\alpha \mathbf{A}_{max}}{\mathbf{s}_{ij}^{-}} \right\rfloor, \quad \mathbf{Q}_{ij}^{im-} = \left\lfloor \frac{(\mathbf{1} - \mathbf{A})_{max}}{\mathbf{s}_{ij}^{+}} \right\rfloor, \tag{6}$$

where $\lfloor \cdot \rfloor$ is the floor function, $\mathbf{A}_{max}$ and $(\mathbf{1} - \mathbf{A})_{max}$ are the largest values in $\mathbf{A}$ and $\mathbf{1} - \mathbf{A}$ respectively, and α balances the over-sampling ratio for positiveness and negativeness.

The introduction of α lies in two reasons: 1) $\mathbf{A}$ is not uniformly distributed in the range $[0, 1]$ and along with the increase of the number of nearest neighbors M, the mean of $\mathbf{A}$ is decreasing; 2) each sample pair $(\mathbf{X}_i, \mathbf{X}_j)$ in $\mathcal{P}_{im}$ is in the top M nearest neighbors of $\mathbf{X}_i$, which means the expansion between $(\mathbf{X}_i, \mathbf{X}_j)$ should be much weaker.

The $\mathbf{Q}^{im+}$ lets the edges with large weights become more likely to be selected to compute $\mathcal{L}_{im}$, while $\mathbf{Q}^{im-}$ behaves exactly the opposite. Then, the augmented sample pair datasets, $\mathcal{E}^{im+}$ and $\mathcal{E}^{im-}$ are given as

$$\mathcal{E}^{im+} = \bigcup_{(\mathbf{X}_i, \mathbf{X}_j) \in \mathcal{G}_{lg}} \bigcup_{r=1}^{\mathbf{Q}_{ij}^{im+}} \{(\mathbf{X}_i, \mathbf{X}_j)\}, \tag{7}$$

$$\mathcal{E}^{im-} = \bigcup_{(\mathbf{X}_i, \mathbf{X}_j) \in \mathcal{G}_{lg}} \bigcup_{r=1}^{\mathbf{Q}_{ij}^{im-}} \{(\mathbf{X}_i, \mathbf{X}_j)\}. \tag{8}$$

The loss $\mathcal{L}_{im}$ for a batch $\mathcal{E}_{bat}^{im+}$, i.e., a subset of $\mathcal{E}^{im+}$, and a batch $\mathcal{E}_{bat}^{im-}$, i.e., a subset of $\mathcal{E}^{im-}$, is defined as

$$\mathcal{L}_{im} = -\sum_{(\mathbf{X}_i, \mathbf{X}_j) \in \mathcal{E}_{bat}^{im+}} \log \mathbf{O}_{ij} - \sum_{(\mathbf{X}_i, \mathbf{X}_j) \in \mathcal{E}_{bat}^{im-}} \log(1 - \mathbf{O}_{ij}). \tag{9}$$

Here, $\log \mathbf{O}_{ij}$ attracts sample pairs in $\mathcal{E}^{im+}$, while $\log(1 - \mathbf{O}_{ij})$ expands sample pairs in $\mathcal{E}^{im-}$.

$\mathcal{P}_{ba}$: Focusing on Samples with Balanced Neighbors

While $\mathcal{P}_{im}$ focused on separating samples near cluster boundaries, the goal of $\mathcal{P}_{ba}$ is to contract those samples and their neighbors to achieve compact clusters. Therefore, the contrastive assignments for $\mathcal{P}_{ba}$ are defined as

$$\mathbf{s}_{ij}^{+} = \mathbf{A}_{ij} \text{ and } \mathbf{s}_{ij}^{-} = 0. \tag{10}$$

Assigning positiveness scores $\mathbf{s}_{ij}^{+} = \mathbf{A}_{ij}$ and extremely low scores to denote negativeness $\mathbf{s}_{ij}^{-} = 0$ for sample pairs let samples within the same cluster contract. To encode them, the QA matrix, $\mathbf{Q}^{ba}$, for $\mathcal{P}_{ba}$ is given by

$$\mathbf{Q}_{ij}^{ba} = \left\lfloor \frac{\mathbf{s}_{ij}^{+}}{\mathbf{A}_{mean}} \right\rfloor, \tag{11}$$

where $\mathbf{A}_{mean}$ is the mean value in $\mathbf{A}$. This way, $\mathbf{Q}^{ba}$ assigns more numbers of the sample pairs with relatively large positiveness scores (i.e., above average) to be included in the augmented sample pair dataset $\mathcal{E}^{ba}$ as

$$\mathcal{E}^{ba} = \bigcup_{(\mathbf{X}_i, \mathbf{X}_j) \in \mathcal{G}_{lg}} \bigcup_{r=1}^{\mathbf{Q}_{ij}^{ba}} \{(\mathbf{X}_i, \mathbf{X}_j)\}. \tag{12}$$

Then, the loss $\mathcal{L}_{ba}$ computed for a batch $\mathcal{E}_{bat}^{ba}$ of training data per epoch is computed as

$$\mathcal{L}_{ba} = -\sum_{(\mathbf{X}_i, \mathbf{X}_j) \in \mathcal{E}_{bat}^{ba}} \log \mathbf{O}_{ij}. \tag{13}$$

By contracting similar sample pairs only, $\mathcal{L}_{ba}$ helps achieve very compact clusters with samples near cluster centers.

$\mathcal{P}_{dis}$: Focusing on Disconnected Samples

In contrast to optimizing connected samples, to make better clusters, disconnected samples should be dramatically separated in $\mathcal{Z}$. Notice that similarity is not needed here; the

model only needs to know if two samples are disconnected (possibly from different clusters) to push them apart. Therefore, based on local similarity from $\mathbf{A}$, a binary disconnectivity matrix $\mathbf{B}_{N\times N}$ is computed with a sign function as

$$\mathbf{B}_{ij} = 1 - \text{sign}(\mathbf{A}_{ij}) = \begin{cases} 0, & \text{if } \mathbf{A}_{ij} > 0 \\ 1, & \text{otherwise} \end{cases}. \tag{14}$$

Note that each element in $\mathbf{B}$ indicates if two samples are disconnected in the kNN graph. To expand the disconnected samples far away, the contrastive assignment is given as

$$\mathbf{s}^{+}_{ij} = 0 \text{ and } \mathbf{s}^{-}_{ij} = \mathbf{B}_{ij}, \tag{15}$$

where only the negativeness scores are considered.

The number of disconnected sample pairs (i.e., non-zero elements in $\mathbf{B}$) is often very large, so using the entire disconnected samples can be inefficient. Therefore, we use a subset of disconnections in the graph as the sample pair dataset $\mathcal{E}^{dis}$.

Let $\mathcal{B}$ be the set of disconnected sample pairs in $\mathbf{B}$. We create $\mathcal{E}^{dis}$ by undersampling the disconnected sample pairs from $\mathcal{B}$ according to the following condition

$$|\mathcal{E}^{dis}| = \beta|\mathcal{E}^{im+}|, \tag{16}$$

where $|\cdot|$ represents the size of the dataset, and β is the hyperparameter to balance the expansion among disconnected samples and the contraction among connected samples from other $\mathcal{P}$. Intuitively, without a sufficient force to separate dissimilar samples from $\mathcal{P}_{dis}$, $\mathcal{P}_{im}$ and $\mathcal{P}_{ba}$ will contract all samples as close as possible without distinct clusters.

Finally, the loss $\mathcal{L}_{dis}$ for a batch $\mathcal{E}^{dis}_{bat}$ is computed as

$$\mathcal{L}_{dis} = -\sum_{(\mathbf{X}_i,\mathbf{X}_j)\in\mathcal{E}^{dis}_{bat}} \log(1-\mathbf{O}_{ij}). \tag{17}$$

Unlike $\mathcal{L}_{im}$ and $\mathcal{L}_{ba}$ which inherit positiveness and negativeness of sample pairs inside local neighborhood, $\mathcal{L}_{dis}$ provides a global view of exploring the negativeness of disconnected sample pairs to avoid all samples from being merged.

3.3 Cluster Assignment with Self-training

Similar to [Ghasedi Dizaji *et al.*, 2017; Van Gansbeke *et al.*, 2020], the clustering head in our framework is implemented with a FC layer, Θ, followed by the softmax function. Given an embedding $\mathbf{Z}_i \in \mathcal{Z}$, the FC generates its cluster assignment probability $\mathbf{U}^k_i$ for each cluster $k = [1, 2, \cdots, K]$.

To iteratively make cluster assignment probability $\mathbf{U}$ localized to a specific cluster, we adopt the self-training process which first calculates a target cluster assignment $\mathbf{V}$ from $\mathbf{U}$ by applying a specific normalization on $\mathbf{U}$, and then minimize the clustering loss $\mathcal{L}_{clu}$, i.e., the Kullback–Leibler (KL) divergence between the cluster assignment matrix $\mathbf{U}$ and $\mathbf{V}$:

$$\mathcal{L}_{clu} = KL(\mathbf{V}||\mathbf{U}) + \gamma\sum_{k\in\mathcal{K}}\bar{\mathbf{U}}^k\log(\bar{\mathbf{U}}^k), \tag{18}$$

where $\mathbf{V}^k_i = \frac{(\mathbf{U}^k_i)^2/\sum_{i'}\mathbf{U}^k_{i'}}{\sum_j(\mathbf{U}^j_i)^2/\sum_{i'}\mathbf{U}^j_{i'}}$, $\bar{\mathbf{U}}^k = \frac{1}{|\mathcal{Z}|}\sum_{\mathbf{Z}_i\in\mathcal{Z}}\mathbf{U}^k_i$ and γ is a hyperparameter to regularize the distribution of clusters to avoid all the predictions being assigned to the same cluster. If the prior distribution of clusters is known, the second term can be replaced with a KL-divergence loss. The overall objective function combines $\mathcal{L}_{im}$, $\mathcal{L}_{ba}$, $\mathcal{L}_{dis}$ and $\mathcal{L}_{clu}$ as

$$\mathcal{L} = \mathcal{L}_{im} + \mathcal{L}_{ba} + \mathcal{L}_{dis} + \eta\mathcal{L}_{clu}, \;\; \eta = \min(\delta \times n_i, \xi) \tag{19}$$

to achieve LNSCC, an end-to-end pipeline to achieve an optimal embedding space and compact clustering of samples. δ is a hyperparameter to balance the clustering and embedding, n_i represents the i-th iteration, and ξ is the upper limit of η.

4 Experiments

In this section, we compare LNSCC with various baseline clustering methods. We first evaluate LNSCC on relatively simple (image and non-image) datasets and then test it on complex image datasets that are adopted by SOTA methods.

4.1 Experimental Setup

Datasets. To evaluate performances of different clustering methods, five popular public datasets were used: REUTERS-10K [Guo *et al.*, 2017], MNIST, MNIST-test, USPS, and Fashion-MNIST. Each dataset contains several classes with ground truths. The datasets are summarized in TABLE 1.

Dataset	# Samples	# Classes	Sample Dimension	Type
REUTERS-10K	10,000	4	2000	non-image
MNIST	70,000	10	28×28	image
MNIST-test	10,000	10	28×28	image
USPS	9,298	10	16×16	image
Fashion-MNIST	10,000	10	28×28	image

Table 1: Summary of Datasets.

Baselines. We adopt 20 different shallow to deep clustering methods for comparison shown in TABLE 2. All shallow methods are listed in the top panel, and Deep methods in the middle and bottom panels are categorized by the use of convolutional techniques, which are suited for image data.

Evaluation Metrics. To evaluate the performance of various clustering algorithms, we adopt the two most common evaluation metrics (i.e., Normalized Mutual Information (NMI) and Clustering Accuracy (ACC)) in our experiments. Unlike ACC which is computed by finding the best match between cluster assignment and target labels, NMI captures the similarity between cluster assignment and target labels which is appropriate for evaluating clustering.

Implementation. The input data were projected onto a low-dimensional space with an MLP (with layers: 500, 500, 2000 nodes) which is widely used in other DC methods, while the clustering head is also implemented using an MLP (with one layer: 32 nodes). The dimension of $\mathcal{Z}$ was set to 64. When constructing the locally normalized kNN graph $\mathcal{G}_{lg}$, the number of nearest neighbors was set to 10 and 70 only for REUTERS-10k dataset, and Euclidean distance was used for distance metric. For the MLPs, we used Stochastic Gradient Descent (SGD) as the optimizer with learning rate 0.01, momentum 0.9, and weight decay 0.0005 and the batch size was set to 200. The α in (6) was 2, the β in (16) was 5, the γ in (18) was 1, and the (δ, ξ) were set as (0.05, 5) in (19). Pretrained VGG [Simonyan and Zisserman, 2015] was used to extract image features if convolution was separately needed.

	w/	MNIST		MNIST-test		USPS		Fashion-MNIST		REUTERS-10K	
	Conv.	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
k-means	✗	0.500	0.534	0.501	0.547	0.450	0.460	0.476	0.512	0.516	0.309*
DBSCAN	✗	–	–	0.114	0	0.167	0	0.100	0	0.403	0.003
DenPeak	✗	–	–	0.357	0.399	0.390	0.433	0.344	0.398	–	–
N-Cut	✗	0.411	0.327	0.753	0.304	0.675	0.314	–	–	–	–
LDMGI	✗	0.802	0.842	0.811	0.847	0.563	0.580	–	–	–	–
SC-ST	✗	0.416	0.311	0.756	0.454	0.726	0.308	–	–	–	–
SC-LS	✗	0.706	0.714	0.756	0.740	0.681	0.659	–	–	–	–
DEC	✗	0.849	0.816	0.856	0.830	0.758	0.769	0.591	0.618	0.737	0.497
IDEC	✗	0.881	0.867	0.846	0.802	0.759	0.777	0.523	0.600	0.756	0.498
DKM	✗	0.840	0.796	–	–	0.757	0.776	–	–	–	–
DCN	✗	0.830	0.810	0.802	0.786	0.688	0.683	–	–	–	–
UMAP^{kmeans}	✗	0.759	0.713	0.848	0.789	0.764	0.789	0.580	0.569	**0.781**	**0.571**
LNSCC (ours)	✗	0.964	0.921	0.946	0.889	0.970	0.937	**0.670**	0.656	**0.812**	**0.605**
k-means$_{VGG}$	✓	0.804	0.682	0.822	0.721	0.792	0.837	0.552	0.598	–	–
JULE	✓	0.964	0.913	0.961	0.915	0.950	0.913	–	–	–	–
DEPICT	✓	0.965	0.917	0.963	0.915	0.964	0.927	–	–	–	–
ConvDEC	✓	0.940	0.916	0.861	0.847	0.784	0.820	0.514	0.588	–	–
ConvDEC-DA	✓	**0.985**	**0.961**	0.955	**0.949**	0.970	**0.953**	0.570	0.632	–	–
DDC	✓	0.965	0.932	0.965	0.916	0.967	0.918	0.619	**0.682**	–	–
DDC-DA	✓	0.969	0.941	**0.970**	0.927	0.977	0.939	0.609	0.661	–	–
UMAP$^{kmeans}_{VGG}$	✓	0.909	0.867	0.962	0.916	**0.978**	0.945	0.593	0.605	–	–
LNSCC$_{VGG}$	✓	**0.983**	**0.971**	**0.982**	**0.969**	**0.981**	**0.969**	**0.665**	**0.703**	–	–

Table 2: Clustering performances on public datasets. Top: Shallow methods, Mid: Deep methods, Bottom: Methods with Convolution (or image features). The results of baselines taken from [Ren *et al.*, 2020; Yang *et al.*, 2019]. "–" and "∗"mean the results are not available or obtained by running provided codes, respectively.

4.2 Results and Discussions

Clustering on Public Datasets. TABLE 2 compares the clustering performances of baseline methods and LNSCC on various datasets. Top-2 algorithms on each dataset are highlighted in bold. TABLE 2 shows that most deep methods perform better than shallow models. Among the deep methods, comparing DEC and ConvDEC clearly tells that the convolution results in an increase in clustering performance for image data. Moreover, comparisons of DDC vs. DDC-DA and ConvDEC vs. ConvDEC-DA show that Data Augmentation (DA) provides an improvement as well.

Without Convolution. Shallow methods mostly did not perform well on these high-dimensional data, and Deep methods yielded reasonable results. On the other hand, LNSCC achieved the best performance in both ACC and NMI on all five datasets across image and non-image data. LNSCC's performances on image data were also comparable with or sometimes even better than the results from the baselines that utilize convolution to extract effective image representations. These results tell that LNSCC is able to learn very effective representation of the data regardless of its modality.

With Image Features. The LNSCC with VGG features achieved at least top-2 ACC and NMI (∼0.98) on MNIST and USPS datasets, and only ConvDEC with DA yielded a slightly better result than LNSCC on the accuracy. For Fashion-MNIST which is more complicated, LNSCC achieved good ACC (0.665) and NMI (0.703) which outperformed all the clustering baselines designed for images. One notable result is that LNSCC without VGG features yielded the highest accuracy on the Fashion-MNIST data. This may be because, while VGG features are effective, their dimensions are too high compared to the original image size.

(a) PCA. (b) UMAP. (c) LNSCC.

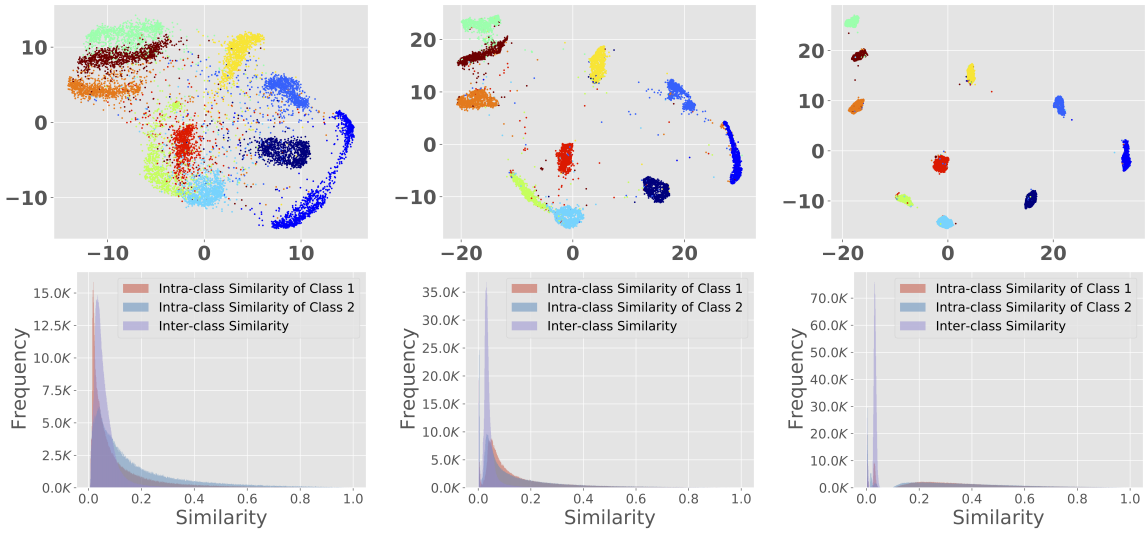
Figure 4: The visualization of embedding on CIFAR10 test dataset with a) PCA, b) UMAP and c) LNSCC in 2D space.

4.3 Evaluations on Challenging Image Datasets

TABLE 3 compares the performances of other recent image clustering methods with LNSCC on more complex datasets including CIFAR10 [Krizhevsky *et al.*, 2009], CIFAR100-20 [Krizhevsky *et al.*, 2009] and STL10 [Coates *et al.*, 2011]. We provide these results separately as their results on MNIST,

	CIFAR10		CIFAR100-20		STL10	
	ACC	NMI	ACC	NMI	ACC	NMI
DeepCluster [Caron *et al.*, 2018]	0.374	-	0.189	-	0.334	-
DAC [Chang *et al.*, 2017]	0.522	0.400	0.238	0.185	0.470	0.366
IIC [Ji *et al.*, 2019]	0.617	0.511	0.257	0.225	0.596	0.496
SCAN-Loss (SimCLR)	0.787	-	-	-	-	-
SCAN-Loss (RA)	0.818	0.712	0.422	0.441	**0.755**	0.654
LNSCC (ours)	**0.820**	**0.713**	**0.439**	**0.446**	0.738	**0.662**

Table 3: Comparisons with SOTA methods on challenging image datasets. LNSCC achieves competitive results.



(a) Beginning. (b) Middle. (c) Final.

Figure 5: Visualization of LNSCC's clustering assignment on MNIST-test data throughout training. Top: clustering results at different training stages, Bot: distribution of similarities between two closest clusters which become localized during training.

USPS, Fashion-MNIST and Reuters-10K have not been reported. The experimental settings follow [Van Gansbeke *et al.*, 2020] for SCAN which trains and evaluates the model using the train and validation splits, respectively.

TABLE 3 shows that SCAN [Van Gansbeke *et al.*, 2020] and our LNSCC outperform other methods with a large gap ($> 17\%$) in accuracy. SCAN uses self-supervised learning (i.e., SimCLR) from [Chen *et al.*, 2020] and a strong data augmentation for self-labeling, i.e., RandAugment (RA) [Cubuk *et al.*, 2020]. For fair comparisons, LNSCC here adopted the same embedding from SimCLR as an input. From the table, we can see SCAN with self-labeling performs much better than the original SCAN. However, our LNSCC can still outperforms SCAN on CIFAR10 and CIFAR100-20 datasets in both ACC and NMI. On STL10 dataset, our LNSCC is still competitive in ACC and achieves the best NMI. Finally, Fig. 4 visualizes the quality of low-dimensional embedding obtained with LNCSS on the CIFAR10 compared to PCA and UMAP, even when only 2D is used for the data embedding.

4.4 Ablation Study and Model Analysis

TABLE 4 shows our ablation study on $\mathcal{L}_{im}$ (separating boundary points), $\mathcal{L}_{ba}$ (contracting sample pairs) and $\mathcal{L}_{dis}$ (expanding sample pairs) using MNIST-test data. It shows that the performance of LNSCC drops (i.e., ACC/NMI drop from 0.94/0.89 to 0.78/0.83 after removing $\mathcal{L}_{im}$ and to 0.80/0.82 when $\mathcal{L}_{ba}$ and $\mathcal{L}_{dis}$ are ablated. It also shows that the expansion is critical in clustering accuracy emphasizing the importance of separating samples near cluster boundaries. The failure cases, i.e., indistinct clusters, were observed when $\mathcal{L}_{im}$ was removed or both $\mathcal{L}_{ba}$ and $\mathcal{L}_{dis}$ were not included.

	w/ $\mathcal{L}_{im}$	w/ $\mathcal{L}_{ba}$	w/ $\mathcal{L}_{dis}$	MNIST-test ACC	MNIST-test NMI
LNSCC	✓	✓	✓	0.946	0.889
	✗	✓	✓	0.784	0.835
	✓	✗	✓	0.915	0.837
	✓	✓	✗	0.810	0.852
	✓	✗	✗	0.801	0.820

Table 4: Ablation study on $\mathcal{L}_{im}$, $\mathcal{L}_{ba}$ and $\mathcal{L}_{dis}$ for LNSCC.

Fig. 5 shows the training process of LNSCC. It is seen that clusters are gradually separated as the training progresses. At the final epoch, individual clusters are compactly clustered with very few false-positives. The bottom row shows intra- and inter-class similarity distributions of the two nearest clusters (i.e., classes) to demonstrate the quality of the clusters. Notice that the intra-class similarities in both clusters (red and blue) become highly localized to small values, and inter-class similarity distribution (purple) gets shifted to larger values. Such a behavior is exactly what we expected with LNSCC.

5 Conclusion

In this paper, we introduced a unified end-to-end clustering framework, i.e., LNSCC, which demonstrate very competitive clustering performance for both image and non-image data. The key idea is to consider both positiveness and negativeness for each pairs of samples in a locally normalized graph to learn an effective low-dimensional embedding of data, and propose a batch-wise training scheme to efficiently perform contrastive learning. The clustering results show very compact clusters, which demonstrate significant potentials to be deployed in various application fields.

Acknowledgments

This research was supported by NSF IIS CRII 1948510, NIH R03 AG070701, IITP-2019-0-01906 (AI Graduate Program at POSTECH) and IITP-2022-2020-0-01461 (ITRC) funded by Ministry of Science and ICT (MSIT), South Korea.

References

[Bolin *et al.*, 2014] Jocelyn H Bolin, Julianne M Edwards, W Holmes Finch, et al. Applications of cluster analysis to the creation of perfectionism profiles: a comparison of two clustering approaches. *Frontiers in psychology*, 5:343, 2014.

[Caron *et al.*, 2018] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, pages 132–149, 2018.

[Caron *et al.*, 2020] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *NeurIPS*, 33:9912–9924, 2020.

[Chang *et al.*, 2017] Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep adaptive image clustering. In *ICCV*, pages 5879–5887, 2017.

[Chen and Cai, 2011] Xinlei Chen and Deng Cai. Large scale spectral clustering with landmark-based representation. In *AAAI*. Citeseer, 2011.

[Chen *et al.*, 2020] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, pages 1597–1607. PMLR, 2020.

[Coates *et al.*, 2011] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, pages 215–223, 2011.

[Costa and Ortale, 2020] Gianni Costa and Riccardo Ortale. Document clustering meets topic modeling with word embeddings. In *ICDM*, pages 244–252. SIAM, 2020.

[Cubuk *et al.*, 2020] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPR Workshop*, pages 702–703, 2020.

[Ester *et al.*, 1996] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96 (34), pages 226–231, 1996.

[Fard *et al.*, 2020] Mazar Moradi Fard, Thibaut Thonet, and Eric Gaussier. Seed-guided deep document clustering. In *ECIR*, pages 3–16. Springer, 2020.

[Ghasedi Dizaji *et al.*, 2017] Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *ICCV*, pages 5736–5745, 2017.

[Guo *et al.*, 2017] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved deep embedded clustering with local structure preservation. In *IJCAI*, pages 1753–1759, 2017.

[Guo *et al.*, 2019] Xifeng Guo, Xinwang Liu, En Zhu, Xinzhong Zhu, Miaomiao Li, Xin Xu, and Jianping Yin. Adaptive self-paced deep clustering with data augmentation. *TKDE*, 32(9):1680–1693, 2019.

[Huang *et al.*, 2020] Jiabo Huang, Shaogang Gong, and Xiatian Zhu. Deep semantic clustering by partition confidence maximisation. In *CVPR*, pages 8849–8858, 2020.

[Ji *et al.*, 2019] Xu Ji, João F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *ICCV*, pages 9865–9874, 2019.

[Kanezaki, 2018] Asako Kanezaki. Unsupervised image segmentation by backpropagation. In *ICASSP*, pages 1543–1547. IEEE, 2018.

[Krizhevsky *et al.*, 2009] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *technical report*, 2009.

[MacQueen and others, 1967] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Berkeley symposium on mathematical statistics and probability*, volume 1(14), pages 281–297. Oakland, CA, USA, 1967.

[Petegrosso *et al.*, 2020] Raphael Petegrosso, Zhuliu Li, and Rui Kuang. Machine learning and statistical methods for clustering single-cell rna-sequencing data. *Briefings in bioinformatics*, 21(4):1209–1223, 2020.

[Ren *et al.*, 2020] Yazhou Ren, Ni Wang, Mingxia Li, and Zenglin Xu. Deep density-based image clustering. *Knowledge-Based Systems*, page 105841, 2020.

[Rodriguez and Laio, 2014] Alex Rodriguez and Alessandro Laio. Clustering by fast search and find of density peaks. *Science*, 344(6191):1492–1496, 2014.

[Simonyan and Zisserman, 2015] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[Tsai *et al.*, 2020] Tsung Wei Tsai, Chongxuan Li, and Jun Zhu. Mice: Mixture of contrastive experts for unsupervised image clustering. In *ICLR*, 2020.

[Van Gansbeke *et al.*, 2020] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Scan: Learning to classify images without labels. In *ECCV*, pages 268–285. Springer, 2020.

[Xie *et al.*, 2016] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, pages 478–487, 2016.

[Xu *et al.*, 2003] Wei Xu, Xin Liu, and Yihong Gong. Document clustering based on non-negative matrix factorization. In *SIGIR*, pages 267–273, 2003.

[Yang *et al.*, 2016] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *CVPR*, pages 5147–5156, 2016.

[Yang *et al.*, 2019] Xu Yang, Cheng Deng, Feng Zheng, Junchi Yan, and Wei Liu. Deep spectral clustering using dual autoencoder network. In *CVPR*, pages 4066–4075, 2019.

[Zelnik-Manor and Perona, 2005] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *NeurIPS*, pages 1601–1608, 2005.

[Zhong *et al.*, 2021] Huasong Zhong, Jianlong Wu, Chong Chen, Jianqiang Huang, Minghua Deng, Liqiang Nie, Zhouchen Lin, and Xian-Sheng Hua. Graph contrastive clustering. In *ICCV*, pages 9224–9233, October 2021.

[Zou *et al.*, 2020] Quan Zou, Gang Lin, Xingpeng Jiang, Xiangrong Liu, and Xiangxiang Zeng. Sequence clustering in bioinformatics: an empirical study. *Briefings in bioinformatics*, 21(1):1–10, 2020.