Joint Unsupervised Learning of Deep Representations and Image Clusters Supplementary materials

Jianwei Yang, Devi Parikh, Dhruv Batra
Virginia Tech
{jw2yang, parikh, dbatra}@vt.edu

Abstract

This supplementary materials explain some implementation details and present additional experiments that are complementary to our main paper "Joint Unsupervised Learning of Deep Representations and Image Clusters". The source code for this work can be downloaded from https://github.com/jwyang/joint-unsupervised-learning.

1. Affinity Measure for Clusters

In this paper, we employ the affinity measure in [13]

$$\begin{aligned}\mathcal{A}(\mathcal{C}_i, \mathcal{C}_j) &= \mathcal{A}(\mathcal{C}_j \rightarrow \mathcal{C}_i) + \mathcal{A}(\mathcal{C}_i \rightarrow \mathcal{C}_j) \\ &= \frac{1}{|\mathcal{C}_i|^2} \mathbf{1}^T_{|\mathcal{C}_i|} \boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_j} \boldsymbol{W}_{\mathcal{C}_j,\mathcal{C}_i} \mathbf{1}_{|\mathcal{C}_i|} \\ &+ \frac{1}{|\mathcal{C}_j|^2} \mathbf{1}^T_{|\mathcal{C}_j|} \boldsymbol{W}_{\mathcal{C}_j,\mathcal{C}_i} \boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_j} \mathbf{1}_{|\mathcal{C}_j|}\end{aligned} \tag{1}$$

where $\boldsymbol{W}$ is the affinity matrix for samples, and $\boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_j} \in \mathbb{R}^{|\mathcal{C}_i| \times |\mathcal{C}_j|}$ is the submatrix in $\boldsymbol{W}$ pointing from samples in $\mathcal{C}_i$ to samples in $\mathcal{C}_j$, and $\boldsymbol{W}_{\mathcal{C}_j,\mathcal{C}_i} \in \mathbb{R}^{|\mathcal{C}_j| \times |\mathcal{C}_i|}$ is the one pointing from $\mathcal{C}_j$ to $\mathcal{C}_i$. $\mathbf{1}_{|\mathcal{C}_i|}$ and $\mathbf{1}_{|\mathcal{C}_j|}$ are two vectors with all $|\mathcal{C}_i|$ and $|\mathcal{C}_j|$ elements be 1, respectively. Therefore, we have $\mathcal{A}(\mathcal{C}_i, \mathcal{C}_j) = \mathcal{A}(\mathcal{C}_j, \mathcal{C}_i)$.

According to (1), we can derive

$$\mathcal{A}((\mathcal{C}_m \cup \mathcal{C}_n) \rightarrow \mathcal{C}_i) = \mathcal{A}(\mathcal{C}_m \rightarrow \mathcal{C}_i) + \mathcal{A}(\mathcal{C}_n \rightarrow \mathcal{C}_i) \tag{2}$$

which has also been shown in [13]. Meanwhile,

$$\begin{aligned}&\mathcal{A}(\mathcal{C}_i \rightarrow (\mathcal{C}_m \cup \mathcal{C}_n)) \\ &= \beta \mathbf{1}^T_{|\mathcal{C}_m|+|\mathcal{C}_n|} \boldsymbol{W}_{\mathcal{C}_m \cup \mathcal{C}_n,\mathcal{C}_i} \boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_m \cup \mathcal{C}_n} \mathbf{1}_{|\mathcal{C}_m|+|\mathcal{C}_n|} \\ &= \beta \mathbf{1}^T_{|\mathcal{C}_m|} \boldsymbol{W}_{\mathcal{C}_m,\mathcal{C}_i} \boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_m} \mathbf{1}_{|\mathcal{C}_m|} + \beta \mathbf{1}^T_{|\mathcal{C}_n|} \boldsymbol{W}_{\mathcal{C}_n,\mathcal{C}_i} \boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_n} \mathbf{1}_{|\mathcal{C}_n|} \\ &+ \beta \mathbf{1}^T_{|\mathcal{C}_m|} \boldsymbol{W}_{\mathcal{C}_m,\mathcal{C}_i} \boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_n} \mathbf{1}_{|\mathcal{C}_n|} + \beta \mathbf{1}^T_{|\mathcal{C}_n|} \boldsymbol{W}_{\mathcal{C}_n,\mathcal{C}_i} \boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_m} \mathbf{1}_{|\mathcal{C}_m|}\end{aligned} \tag{3}$$

where $\beta = 1/(|\mathcal{C}_m| + |\mathcal{C}_n|)^2$.

2. Approximated Affinity Measure

During agglomerative clustering, we need to re-compute the affinity between the merged cluster to all other clusters based on 2 and 3 repeatedly. It is simple to compute 2. However, to get $\mathcal{A}(\mathcal{C}_i \rightarrow (\mathcal{C}_m \cup \mathcal{C}_n))$, we need a lot of computations. These time costs become dominant and remarkable when we have a large-scale dataset. To accelerate the computations, we introduce an approximation method. At the right side of (3), we assume samples in $\mathcal{C}_m$ and $\mathcal{C}_n$ have similar affinities to $\mathcal{C}_i$. This assumption is mild because the condition to merge $\mathcal{C}_m$ and $\mathcal{C}_n$ is that they are similar to each other. In this case, the ratio between $\boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_m} \mathbf{1}_{|\mathcal{C}_m|}$ and $\boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_n} \mathbf{1}_{|\mathcal{C}_n|}$ is analogy to the ratio between the number of samples in two set, i.e.,

$$\boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_m} \mathbf{1}_{|\mathcal{C}_m|} = \frac{|\mathcal{C}_m|}{|\mathcal{C}_n|} \boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_n} \mathbf{1}_{|\mathcal{C}_n|} \tag{4}$$

Based on (4), we can re-formulate (3) to

$$\begin{aligned}&\mathcal{A}(\mathcal{C}_i \rightarrow (\mathcal{C}_m \cup \mathcal{C}_n)) \\ &= \frac{1}{(|\mathcal{C}_m|^2 + |\mathcal{C}_m||\mathcal{C}_n|)} \mathbf{1}^T_{|\mathcal{C}_m|} \boldsymbol{W}_{\mathcal{C}_m,\mathcal{C}_i} \boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_m} \mathbf{1}_{|\mathcal{C}_m|} \\ &+ \frac{1}{(|\mathcal{C}_m||\mathcal{C}_n| + |\mathcal{C}_n|^2)} \mathbf{1}^T_{|\mathcal{C}_n|} \boldsymbol{W}_{\mathcal{C}_n,\mathcal{C}_i} \boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_n} \mathbf{1}_{|\mathcal{C}_n|}\end{aligned} \tag{5}$$

Therefore, we have

$$\begin{aligned}\mathcal{A}(\mathcal{C}_i \rightarrow (\mathcal{C}_m \cup \mathcal{C}_n)) &= \frac{|\mathcal{C}_m|}{|\mathcal{C}_m| + |\mathcal{C}_n|} \mathcal{A}(\mathcal{C}_i \rightarrow \mathcal{C}_m) \\ &+ \frac{|\mathcal{C}_n|}{|\mathcal{C}_m| + |\mathcal{C}_n|} \mathcal{A}(\mathcal{C}_i \rightarrow \mathcal{C}_n)\end{aligned} \tag{6}$$

Consequently, we have

$$\begin{aligned}\mathcal{A}(\mathcal{C}_m \cup \mathcal{C}_n, \mathcal{C}_i) &= \mathcal{A}(\mathcal{C}_m \rightarrow \mathcal{C}_i) + \mathcal{A}(\mathcal{C}_n \rightarrow \mathcal{C}_i) \\ &+ \frac{|\mathcal{C}_m|}{|\mathcal{C}_m| + |\mathcal{C}_n|} \mathcal{A}(\mathcal{C}_i \rightarrow \mathcal{C}_m) \\ &+ \frac{|\mathcal{C}_n|}{|\mathcal{C}_m| + |\mathcal{C}_n|} \mathcal{A}(\mathcal{C}_i \rightarrow \mathcal{C}_n)\end{aligned} \tag{7}$$

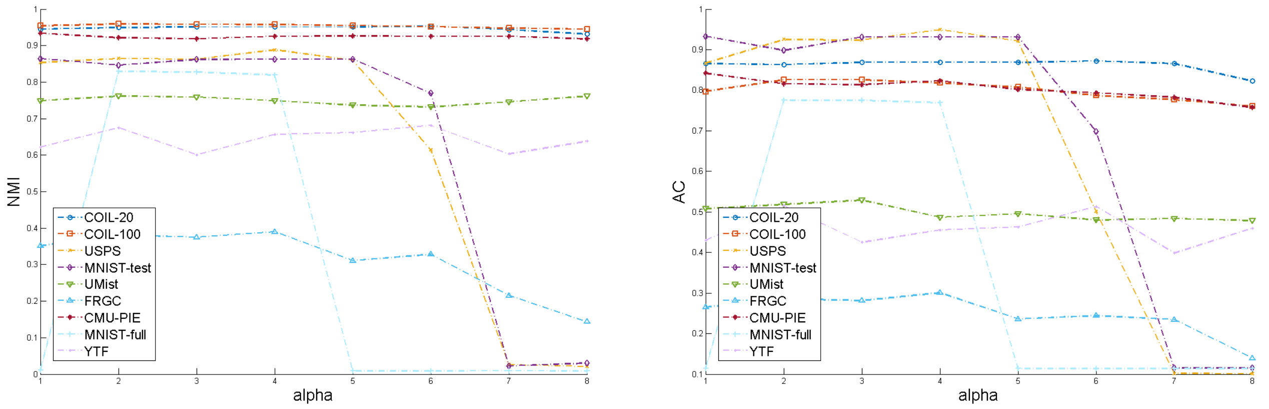


Figure 1: Performance of agglomerative clustering with approximations. Left one is NMI metric, and right one is AC metric. The first column is without acceleration. For the other columns from left to right, $\alpha = \{-0.2, -0.1, 0, 0.1, 0.2, 0.3, 0.5\}$.

Above approximation provides us a potential way to reduce the computational complexity of agglomerative clustering. Though we computed $\mathcal{A}(\mathcal{C}_i \rightarrow (\mathcal{C} \cup \mathcal{C}_n))$ based on (3) in all our experiments, we found the approximation version achieves analogy performance while costs much less time than the original one. In our experiments, we further simplify the computation by assuming a constant ratio α between the terms in 3:

$$\mathbf{1}^T_{|\mathcal{C}_m|}\boldsymbol{W}_{\mathcal{C}_m,\mathcal{C}_i}\boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_n}\mathbf{1}_{|\mathcal{C}_n|} = \alpha\mathbf{1}^T_{|\mathcal{C}_m|}\boldsymbol{W}_{\mathcal{C}_m,\mathcal{C}_i}\boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_m}\mathbf{1}_{|\mathcal{C}_m|} \tag{8a}$$

$$\mathbf{1}^T_{|\mathcal{C}_n|}\boldsymbol{W}_{\mathcal{C}_n,\mathcal{C}_i}\boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_m}\mathbf{1}_{|\mathcal{C}_m|} = \alpha\mathbf{1}^T_{|\mathcal{C}_n|}\boldsymbol{W}_{\mathcal{C}_n,\mathcal{C}_i}\boldsymbol{W}_{\mathcal{C}_i,\mathcal{C}_n}\mathbf{1}_{|\mathcal{C}_n|} \tag{8b}$$

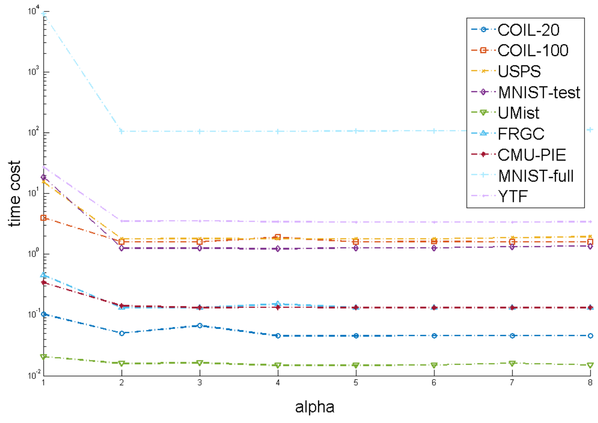


Figure 2: Time cost for different values of α. The first column is the time cost without acceleration. For the other columns from left to right, $\alpha = \{-0.2, -0.1, 0, 0.1, 0.2, 0.3, 0.5\}$.

Based on above assumption,

$$\begin{aligned}\mathcal{A}(\mathcal{C}_m \cup \mathcal{C}_n, \mathcal{C}_i) &= \mathcal{A}(\mathcal{C}_m \rightarrow \mathcal{C}_i) + \mathcal{A}(\mathcal{C}_n \rightarrow \mathcal{C}_i) \\ &+ \frac{(1+\alpha)|\mathcal{C}_m|^2}{(|\mathcal{C}_m| + |\mathcal{C}_n|)^2}\mathcal{A}(\mathcal{C}_i \rightarrow \mathcal{C}_m) \\ &+ \frac{(1+\alpha)|\mathcal{C}_n|^2}{(|\mathcal{C}_m| + |\mathcal{C}_n|)^2}\mathcal{A}(\mathcal{C}_i \rightarrow \mathcal{C}_n)\end{aligned} \tag{9}$$

We test various values for α, which are $\{-0.2, -0.1, 0, 0.1, 0.2, 0.3, 0.5\}$. By conducting experiments on various datasets, we find a valid range $[0, 0.1]$ for α which helps achieve analogous or even better performance to the one without acceleration. We show the quantitative comparison in Fig. 1. We use image intensities as input to rule out all random factors. The original AC-GDL algorithm is used as the baseline. Also, we compare the time cost between original AC-GDL algorithm and accelerated one in Fig. 2. It is clear that our approximation algorithm has much lower computational complexity.

3. Cluster-based to Sample-based Loss

In this part, we explain how to convert cluster-based loss to sample-based loss. Because it depends on specific agglomerative clustering processes, we use a toy example in Fig. 3 for illustration. We set K_c be 2 for simplicity. In Fig. 3, there are six time steps, and thus $T = 6$. We assume they are in a single partial unrolled period. The leaf nodes represent single samples. For simplicity, we omit $\frac{\lambda}{K_c - 1}$ in (10) in the main paper, obtaining the overall loss

$$\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{\mathcal{Y}}_*, \boldsymbol{I}) = \sum_{t=1}^{6}\left(\lambda'\mathcal{A}(\mathcal{C}^t_*, \mathcal{N}^{K_c}_{\mathcal{C}^t_*}[1]) - \mathcal{A}(\mathcal{C}^t_*, \mathcal{N}^{K_c}_{\mathcal{C}^t_*}[2])\right) \tag{10}$$

Given above loss function, we decompose it from first time step ($t = 1$) to the most recent time step ($t = 6$):

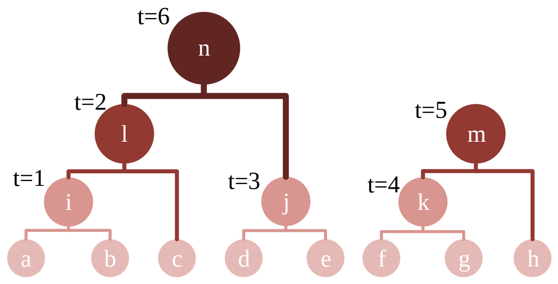


Figure 3: A illustration of agglomerative clustering.

- **t=1**: $\mathcal{C}_*^1 = \mathcal{C}_a, \mathcal{N}^2_{\mathcal{C}_*^1}[1] = \mathcal{C}_b$ and $\mathcal{N}^2_{\mathcal{C}_*^1}[2] = \mathcal{C}_c$. We have

$$\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{y}_*^1, I) = -(\lambda'\boldsymbol{\mathcal{A}}(\mathcal{C}_a, \mathcal{C}_b) - \boldsymbol{\mathcal{A}}(\mathcal{C}_a, \mathcal{C}_c)) \tag{11}$$

Clearly, above is sample-based weighted triplet loss function, where samples a and b are positive pair and a and c are negative pair.

- **t=2**: $\mathcal{C}_*^2 = \mathcal{C}_i, \mathcal{N}^2_{\mathcal{C}_*^2}[1] = \mathcal{C}_c$ and $\mathcal{N}^2_{\mathcal{C}_*^2}[2] = \mathcal{C}_d$. We have

$$\mathcal{L}(\boldsymbol{\theta}|\{\boldsymbol{y}_*^1, \boldsymbol{y}_*^2\}, I) = \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{y}_*^1, I) - (\lambda'\boldsymbol{\mathcal{A}}(\mathcal{C}_i, \mathcal{C}_c) - \boldsymbol{\mathcal{A}}(\mathcal{C}_i, \mathcal{C}_d)) \tag{12}$$

Since $\mathcal{C}_i = \mathcal{C}_a \cup \mathcal{C}_b$, we use Eq. (7) for approximation

$$\begin{aligned}\boldsymbol{\mathcal{A}}(\mathcal{C}_i, \mathcal{C}_c) &= \boldsymbol{\mathcal{A}}(\mathcal{C}_a \to \mathcal{C}_c) + \boldsymbol{\mathcal{A}}(\mathcal{C}_b \to \mathcal{C}_c) \\ &+ \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_c \to \mathcal{C}_a) + \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_c \to \mathcal{C}_b)\end{aligned} \tag{13}$$

$$\begin{aligned}\boldsymbol{\mathcal{A}}(\mathcal{C}_i, \mathcal{C}_d) &= \boldsymbol{\mathcal{A}}(\mathcal{C}_a \to \mathcal{C}_d) + \boldsymbol{\mathcal{A}}(\mathcal{C}_b \to \mathcal{C}_d) \\ &+ \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_d \to \mathcal{C}_a) + \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_d \to \mathcal{C}_b)\end{aligned} \tag{14}$$

Thus,

$$\begin{aligned}&\mathcal{L}(\boldsymbol{\theta}|\{\boldsymbol{y}_*^1, \boldsymbol{y}_*^2\}, I) \\ &= -\lambda'\boldsymbol{\mathcal{A}}(\mathcal{C}_a, \mathcal{C}_b) - (\lambda' - 1)\boldsymbol{\mathcal{A}}(\mathcal{C}_a \to \mathcal{C}_c) - \lambda'\boldsymbol{\mathcal{A}}(\mathcal{C}_b \to \mathcal{C}_c) \\ &- (\frac{\lambda'}{2} - 1)\boldsymbol{\mathcal{A}}(\mathcal{C}_c \to \mathcal{C}_a) - \frac{\lambda'}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_c \to \mathcal{C}_b) \\ &+ \boldsymbol{\mathcal{A}}(\mathcal{C}_a \to \mathcal{C}_d) + \boldsymbol{\mathcal{A}}(\mathcal{C}_b \to \mathcal{C}_d) \\ &+ \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_d \to \mathcal{C}_a) + \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_d \to \mathcal{C}_b)\end{aligned} \tag{15}$$

At current time step, sample a, b and c belong to the same cluster $\mathcal{C}_l$, while sample d is from another cluster. (15) computes the sample-based weighted triplet loss for samples in $\mathcal{C}_l$ and sample d. Except $\mathcal{C}_l$, the other clusters all have merely one sample. No need to compute triplet loss for them. It should be pointed out that λ' in above loss function should be not less than 2 so that the affinities for all pairs in $\mathcal{C}_l$ are enlarged.

- **t=3**: $\mathcal{C}_*^3 = \mathcal{C}_d$, $\mathcal{N}^2_{\mathcal{C}_*^3}[1] = \mathcal{C}_e$ and $\mathcal{N}^2_{\mathcal{C}_*^3}[2] = \mathcal{C}_f$. We have

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}|\{\boldsymbol{y}_*^1, \boldsymbol{y}_*^2, \boldsymbol{y}_*^3\}, I) &= \mathcal{L}(\boldsymbol{\theta}|\{\boldsymbol{y}_*^1, \boldsymbol{y}_*^2\}, I) \\ &- \lambda'(\boldsymbol{\mathcal{A}}(\mathcal{C}_d, \mathcal{C}_e) - \boldsymbol{\mathcal{A}}(\mathcal{C}_d, \mathcal{C}_f))\end{aligned} \tag{16}$$

Besides the loss $\mathcal{L}(\boldsymbol{\theta}|\{\boldsymbol{y}_*^1, \boldsymbol{y}_*^2\}, I)$ for $\mathcal{C}_l$, we also compute the loss for $\mathcal{C}_j$ in (16) because it contains two samples, d and e.

- **t=4**: $\mathcal{C}_*^4 = \mathcal{C}_f$, $\mathcal{N}^2_{\mathcal{C}_*^4}[1] = \mathcal{C}_g$ and $\mathcal{N}^2_{\mathcal{C}_*^4}[2] = \mathcal{C}_h$. We have

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}|\{\boldsymbol{y}_*^1, ..., \boldsymbol{y}_*^4\}, I) &= \mathcal{L}(\boldsymbol{\theta}|\{\boldsymbol{y}_*^1, \boldsymbol{y}_*^2, \boldsymbol{y}_*^3\}, I) \\ &- (\lambda'\boldsymbol{\mathcal{A}}(\mathcal{C}_f, \mathcal{C}_g) - \boldsymbol{\mathcal{A}}(\mathcal{C}_f, \mathcal{C}_h))\end{aligned} \tag{17}$$

Here, we additionally compute the weighted triplet loss for cluster $\mathcal{C}_k$ since it contains two samples.

- **t=5**: $\mathcal{C}_*^5 = \mathcal{C}_k$, $\mathcal{N}^2_{\mathcal{C}_*^5}[1] = \mathcal{C}_h$ and $\mathcal{N}^2_{\mathcal{C}_*^5}[2] = \mathcal{C}_j$. We have

$$\begin{aligned}&\mathcal{L}(\boldsymbol{\theta}|\{\boldsymbol{y}_*^1, ..., \boldsymbol{y}_*^5\}, I) \\ &= \mathcal{L}(\boldsymbol{\theta}|\{\boldsymbol{y}_*^1, ..., \boldsymbol{y}_*^4\}, I) - (\lambda'\boldsymbol{\mathcal{A}}(\mathcal{C}_k, \mathcal{C}_h) - \boldsymbol{\mathcal{A}}(\mathcal{C}_k, \mathcal{C}_j))\end{aligned} \tag{18}$$

Because $\mathcal{C}_k = \mathcal{C}_f \cup \mathcal{C}_g$, we have

$$\begin{aligned}\boldsymbol{\mathcal{A}}(\mathcal{C}_k, \mathcal{C}_h) &= \boldsymbol{\mathcal{A}}(\mathcal{C}_f \to \mathcal{C}_h) + \boldsymbol{\mathcal{A}}(\mathcal{C}_g \to \mathcal{C}_h) \\ &+ \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_h \to \mathcal{C}_f) + \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_h \to \mathcal{C}_g)\end{aligned} \tag{19}$$

$$\begin{aligned}\boldsymbol{\mathcal{A}}(\mathcal{C}_k, \mathcal{C}_j) &= \boldsymbol{\mathcal{A}}(\mathcal{C}_f \to \mathcal{C}_j) + \boldsymbol{\mathcal{A}}(\mathcal{C}_g \to \mathcal{C}_j) \\ &+ \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_j \to \mathcal{C}_f) + \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_j \to \mathcal{C}_g)\end{aligned} \tag{20}$$

Since $\mathcal{C}_j = \mathcal{C}_d \cup \mathcal{C}_e$, we further transform above equation to

$$\begin{aligned}\boldsymbol{\mathcal{A}}(\mathcal{C}_k, \mathcal{C}_j) &= \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_f \to \mathcal{C}_d) + \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_f \to \mathcal{C}_e) \\ &+ \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_g \to \mathcal{C}_d) + \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_g \to \mathcal{C}_e) \\ &+ \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_d \to \mathcal{C}_f) + \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_e \to \mathcal{C}_f) \\ &+ \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_d \to \mathcal{C}_g) + \frac{1}{2}\boldsymbol{\mathcal{A}}(\mathcal{C}_e \to \mathcal{C}_g)\end{aligned} \tag{21}$$

Similar to the relation between sample a and c at time steps $t = 1, 2$, sample f and h belong to the same cluster $\mathcal{C}_m$ at current time step while they are from different clusters at time step $t = 4$. Based on the approximation, the terms $\boldsymbol{\mathcal{A}}(\mathcal{C}_f \to \mathcal{C}_h)$ and $\boldsymbol{\mathcal{A}}(\mathcal{C}_h \to \mathcal{C}_f)$ in two time steps will be merged. As a result, the final loss is computed on intra-cluster pairs and inter-cluster pairs sampled from three clusters $\mathcal{C}_l$, $\mathcal{C}_j$ and $\mathcal{C}_m$.

- **t=6**: $\mathcal{C}_*^6 = \mathcal{C}_l, \mathcal{N}^2_{\mathcal{C}_*^6}[1] = \mathcal{C}_j$ and $\mathcal{N}^2_{\mathcal{C}_*^6}[2] = \mathcal{C}_m$. Thus

$$\mathcal{L}(\boldsymbol{\theta}|\{\boldsymbol{y}_*^1, ..., \boldsymbol{y}_*^6\}, I) = \mathcal{L}(\boldsymbol{\theta}|\{\boldsymbol{y}_*^1, ..., \boldsymbol{y}_*^5\}, I) - (\lambda' \boldsymbol{\mathcal{A}}(\mathcal{C}_l, \mathcal{C}_j) - \boldsymbol{\mathcal{A}}(\mathcal{C}_l, \mathcal{C}_m)) \tag{22}$$

  Similar to the decomposition procedures above, both $\boldsymbol{\mathcal{A}}(\mathcal{C}_l, \mathcal{C}_j)$ and $\boldsymbol{\mathcal{A}}(\mathcal{C}_l, \mathcal{C}_m)$ can be transformed to sample-based affinities. Because $\mathcal{C}_l$ and $\mathcal{C}_j$ are regarded as different clusters previously, sample pairs from both of them are with positive weights in the loss function. However, it will be diminished by positive pairs (with negative weights) at current time step.

Though we use a toy example to show that the cluster-based loss can be transformed to sample-based loss above, the reduction is general to any possible agglomerative clustering processes because the loss for clusters at high-level can always be decomposed to the losses on clusters at low-level until it reaches to single samples. The difference among various processes lies on the different weights associated with sample-based affinities. At this point, we should know that sample pairs from the same cluster may be with positive weights. One way to avoid this is increase λ'. In our implementation, we aim to increase affinities between samples from the same clusters, while decrease the affinities between samples from different clusters. And the clusters are determined by cluster ids at current step. Therefore, we assign a consistent weight γ to any affinities from the same cluster and 1 to any affinities from different clusters. Because we use SGD for batch optimization, the scales for affinities do not affect much on the performance. It is the signs affect much. Accordingly, at any given time step T, the overall loss is approximated to

$$\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{y}_*^T, \boldsymbol{I}) = -\frac{\lambda}{K_c - 1} \sum_{i,j,k} (\gamma \boldsymbol{\mathcal{A}}(\boldsymbol{x}_i, \boldsymbol{x}_j) - \boldsymbol{\mathcal{A}}(\boldsymbol{x}_i, \boldsymbol{x}_k)) \tag{23}$$

Note that we replace $\boldsymbol{\mathcal{Y}}_*$ in (23) by $\boldsymbol{y}_*^T$ in (23) because it is merely determined by current $\boldsymbol{y}^T$, regardless of $\{\boldsymbol{y}_*^1, ..., \boldsymbol{y}_*^{T-1}\}$. As a result, we do not need to record $\{\boldsymbol{y}_*^1, ..., \boldsymbol{y}_*^{T-1}\}$. This simplifies the batch optimization for CNN. Concretely, given a sample $\boldsymbol{x}_i$, we randomly select a sample $\boldsymbol{x}_j$ which belongs to the same cluster, while select neighbours of $\boldsymbol{x}_i$ that from other clusters to be $\boldsymbol{x}_k$. To omit the case that $\boldsymbol{\mathcal{A}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is much larger than $\boldsymbol{\mathcal{A}}(\boldsymbol{x}_i, \boldsymbol{x}_k)$, we also add a margin threshold like the triplet loss function used in [8, 11].

4. Detailed CNN Architectures in our Paper

In this paper, the CNN architectures vary from dataset to dataset. As we mentioned in the main paper, we stacked different number of layers for different datasets so that the size of most top layer response map is about 10×10. In Table 1, we list the architectures for the datasets used in our paper. ”conv” means convolutional layer. ”bn” means batch normalization layer. ”wt-loss” means weighted triplet loss layer. ✓ means the layer is used, while $-$ means the layer is not used.

5. Performance Evaluated by Accuracy

In this section, we evaluate the performance of different algorithms based on clustering accuracy (AC) metric, as a supplement to the NMI metric used in our main paper. As we can see from table 2, the proposed method outperform other methods on all datasets, which has similar trend as evaluated using NMI. Meanwhile, according to table 3, all other clustering algorithms are boosted after using the learned representation as evaluated on AC. These results further prove the proposed method is superior to other clustering algorithms and also learns powerful deep representations that generalize well across different clustering algorithms.

6. Robustness Analysis

We choose the two most important parameters: unfolding rate η and K_s for evaluating the robustness of our approach to variations in these parameters. In these experiments, we set all the other parameters except for the target one to default values listed in Table 2 in the main paper. As we can see from Fig. 4, when the unfolding rate increases, the performance is not affected much for most of the datasets. For K_s, the performance is stable when $K_s <= 50$ for all datasets. It drops with larger values of K_s for a few datasets. Increasing K_s also result in similar degradation in the agglomerative clustering algorithms we compare to. This suggests that K_s should not be set to very large value in general.

7. Reliability Analysis

We evaluate the reliability by measuring the purity of samples at the beginning of our algorithm. Because we use agglomerative clustering, there are very few samples in each cluster at the beginning (average is about 4 in our experiments). Most samples in the same cluster tend to belong to the same category. Quantitatively, for each sample in a dataset, we count the number of samples (K_m) that belong to the same category within its K nearest neighbours, and then compute the precision K_m/K for it. In Fig. 5, we report the average precision across all samples. As we can see, based on raw image data, all datasets have high ratios when K is smaller, and the ratios increase further when using our learned deep representations. Consequently, when K is small, the pseudo-labels are reliable enough to learn plausible deep representations.

Table 1: CNN architectures for different datasets in our paper.

Dataset	*COIL20*	*COIL100*	*USPS*	*MNIST-test*	*MNIST-full*	UMist	*FRGC*	*CMU-PIE*	*YTF*
conv1	✓	✓	✓	✓	✓	✓	✓	✓	✓
bn1	✓	✓	✓	✓	✓	✓	✓	✓	✓
relu1	✓	✓	✓	✓	✓	✓	✓	✓	✓
pool1	✓	✓	✓	✓	✓	✓	✓	✓	✓
conv2	✓	✓	–	✓	✓	✓	✓	✓	✓
bn2	✓	✓	–	✓	✓	✓	✓	✓	✓
relu2	✓	✓	–	✓	✓	✓	✓	✓	✓
pool2	✓	✓	–	–	–	✓	✓	✓	✓
conv3	✓	✓	–	–	–	✓	–	–	–
bn3	✓	✓	–	–	–	✓	–	–	–
relu3	✓	✓	–	–	–	✓	–	–	–
pool3	✓	✓	–	–	–	✓	–	–	–
conv4	✓	✓	–	–	–	–	–	–	–
bn4	✓	✓	–	–	–	–	–	–	–
relu4	✓	✓	–	–	–	–	–	–	–
pool4	✓	✓	–	–	–	–	–	–	–
ip1	✓	✓	✓	✓	✓	✓	✓	✓	✓
l2-norm	✓	✓	✓	✓	✓	✓	✓	✓	✓
wt-loss	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 2: Quantitative clustering performance (AC) for different algorithms using image intensities as input.

Dataset	*COIL20*	*COIL100*	*USPS*	*MNIST-test*	*MNIST-full*	*UMist*	*FRGC*	*CMU-PIE*	*YTF*
K-means [5]	0.665	0.580	0.467	0.560	0.564	0.419	0.327	0.246	0.548
SC-NJW [6]	0.641	0.544	0.413	0.220	0.502	0.551	0.178	0.255	0.551
SC-ST [12]	0.417	0.300	0.308	0.454	0.311	0.411	0.358	0.293	0.290
SC-LS [10]	0.717	0.609	0.659	0.740	0.714	0.568	0.407	0.549	0.544
N-Cuts [9]	0.544	0.577	0.314	0.304	0.327	0.550	0.235	0.155	0.536
AC-Link [3]	0.251	0.269	0.421	0.693	0.657	0.398	0.175	0.201	0.547
AC-Zell [15]	0.867	0.811	0.575	0.693	0.112	0.517	0.266	0.765	0.519
AC-GDL [13]	0.865	0.797	0.867	0.933	0.113	0.563	0.266	0.842	0.430
AC-PIC [14]	0.855	0.840	0.855	0.920	0.115	0.576	0.320	0.797	0.472
NMF-LP [1]	0.621	0.553	0.522	0.479	0.471	0.365	0.259	0.229	0.546
OURS-SF	**1.000**	0.894	0.922	0.940	0.959	**0.809**	**0.461**	0.980	**0.684**
OURS-RC	**1.000**	**0.916**	**0.950**	**0.961**	**0.964**	**0.809**	**0.461**	**1.000**	**0.684**

Table 3: Quantitative clustering performance (AC) for different algorithms using our learned representations as inputs.

Dataset	*COIL20*	*COIL100*	*USPS*	*MNIST-test*	*MNIST-full*	*UMist*	*FRGC*	*CMU-PIE*	*YTF*
K-means [5]	0.821	0.751	0.776	0.957	0.969	0.761	0.476	0.834	0.660
SC-NJW [6]	0.738	0.659	0.716	0.868	0.972	0.707	0.485	0.776	0.521
SC-ST [12]	0.851	0.705	0.661	0.960	0.958	0.697	0.496	0.896	0.575
SC-LS [10]	0.867	0.735	0.792	0.960	**0.973**	0.733	0.502	0.802	0.571
N-Cuts [9]	0.888	0.626	0.634	0.959	0.971	**0.798**	**0.504**	0.981	0.441
AC-Link [3]	0.678	0.539	0.773	0.955	0.964	0.795	0.495	0.947	0.602
AC-Zell [15]	**1.000**	0.931	0.879	0.879	0.969	0.790	0.449	**1.000**	0.644
AC-GDL [13]	**1.000**	0.920	0.949	**0.961**	0.878	0.790	0.461	**1.000**	**0.677**
AC-PIC [14]	**1.000**	**0.950**	**0.955**	0.958	0.882	0.790	0.438	**1.000**	0.652
NMF-LP [1]	0.769	0.603	0.778	0.955	0.970	0.725	0.481	0.504	0.575

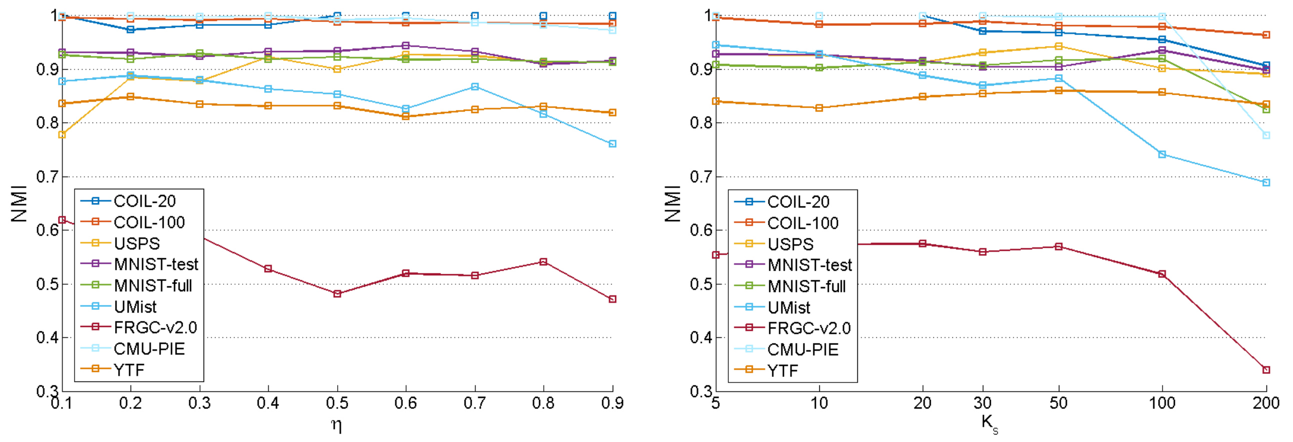


Figure 4: Clustering performance (NMI) with different η (left) and K_s (right).

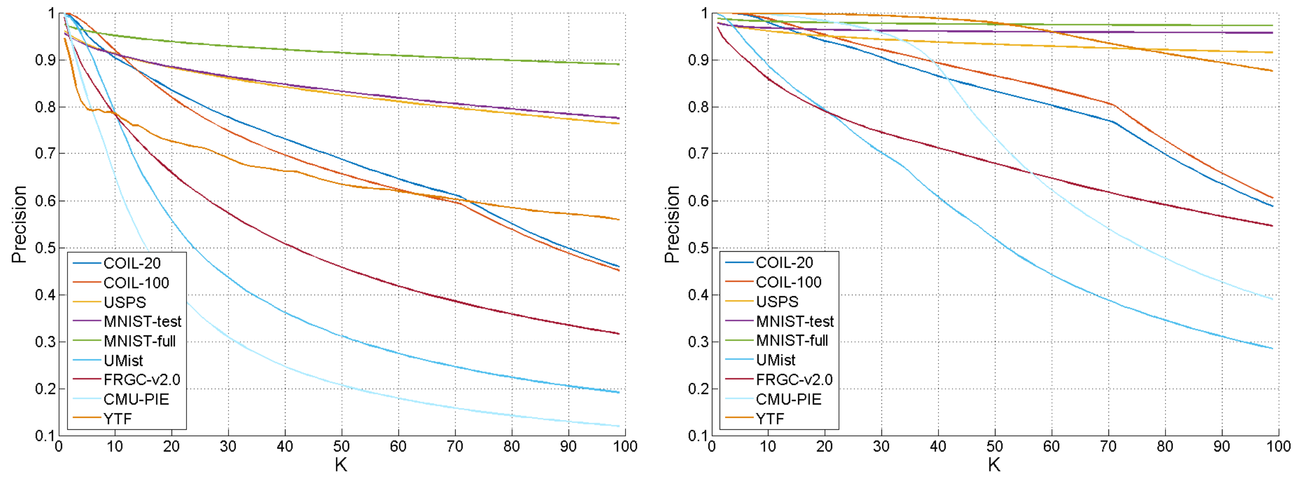


Figure 5: Average purity of K-nearest neighbour for varying values of K. Left is computed using raw image data, while right is computed using our learned representation.

8. Clustering based on Hand-Crafted Features

We also evaluate the performance of clustering based on image features, instead of image intensities. We choose three different types of datasets for testing: COIL100, MNIST-test and UMist, and three types of clustering algorithms including SC-LS [10], N-Cuts [9] and AC-PIC [14] for comparison since their better performance among all the algorithms. For these three datasets, we use spatial pyramid descriptor [4][1], histogram of oriented gradient (HOG) [2][2] and local binary pattern (LBP) [7] for representation, respectively. We report the results in Table 4. $\downarrow$ means performance become worse, and $\uparrow$ means it become better. Almost all algorithms perform worse than using original image as input. It indicates hand-crafted features should be designed dataset by dataset. In contrast, directly learning from image intensities is more straightforward and also achieves better performance.

Table 4: Clustering performance (NMI) based on hand-crafted features.

Dataset	COIL100	MNIST-test	UMist	FRGC
SC-LS [10]	0.733$\downarrow$	0.625$\downarrow$	0.752$\downarrow$	0.338$\downarrow$
N-Cuts [9]	0.722$\downarrow$	0.423$\uparrow$	0.420$\downarrow$	0.238$\downarrow$
AC-PIC [14]	0.878$\downarrow$	0.735$\downarrow$	0.734$\downarrow$	0.322$\downarrow$

[1]http://slazebni.cs.illinois.edu/research/SpatialPyramid.zip

[2]http://www.robots.ox.ac.uk/~vgg/research/caltech/phog.html

9. Visualizing Learned Deep Representations

We show the first three principle components of learned representations in Fig. 6 and Fig. 7 at different stages. For comparison, we show the image intensities at the first column. We use different colors for representing different clusters that we predict during the algorithm. At the bottom of each plot, we give the number of clusters at the corresponding stage. At the final stage, the number of cluster is same to the number of categories in the dataset. After a number of iterations, we can learn more discriminative representations for the datasets, and thus facilitate more precise clustering results.

References

[1] D. Cai, X. He, X. Wang, H. Bao, and J. Han. Locality preserving nonnegative matrix factorization. In *IJCAI*, volume 9, pages 1010–1015, 2009. 5

[2] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005. 6

[3] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999. 5

[4] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2169–2178. IEEE, 2006. 6

[5] 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*, volume 1, pages 281–297. Oakland, CA, USA., 1967. 5

[6] A. Y. Ng, M. I. Jordan, Y. Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002. 5

[7] T. Ojala, M. Pietikäinen, and D. Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29(1):51–59, 1996. 6

[8] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. *arXiv preprint arXiv:1503.03832*, 2015. 4

[9] J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000. 5, 6

[10] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 5, 6

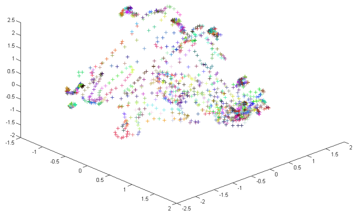
[11] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1386–1393. IEEE, 2014. 4

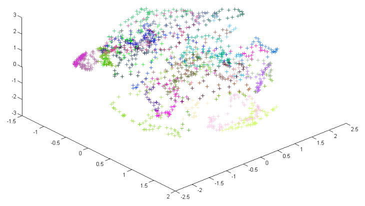
[12] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. pages 1601–1608, 2004. 5

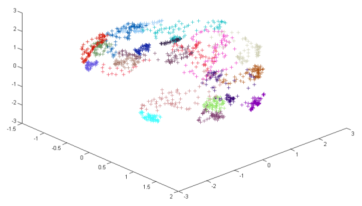
[13] W. Zhang, X. Wang, D. Zhao, and X. Tang. Graph degree linkage: Agglomerative clustering on a directed graph. In *Computer Vision–ECCV 2012*, pages 428–441. Springer, 2012. 1, 5

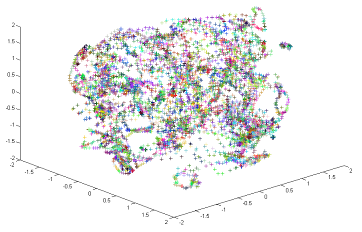
[14] W. Zhang, D. Zhao, and X. Wang. Agglomerative clustering via maximum incremental path integral. *Pattern Recognition*, 46(11):3056–3065, 2013. 5, 6

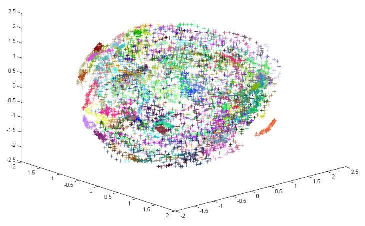
[15] D. Zhao and X. Tang. Cyclizing clusters via zeta function of a graph. In *Advances in Neural Information Processing Systems*, pages 1953–1960, 2009. 5

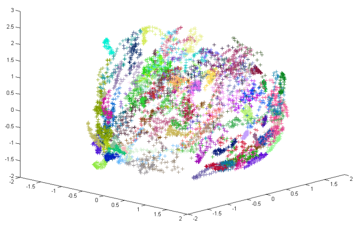

(a) Initial stage (421)

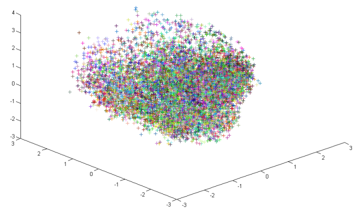

(b) Middle stage (42)

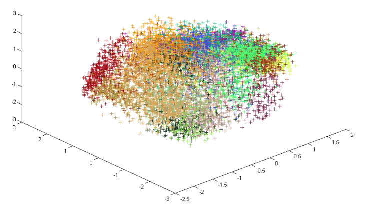

(c) Final stage (20)

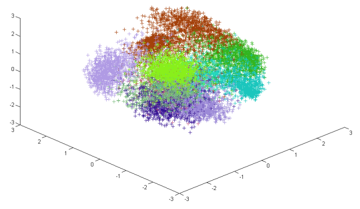

(d) Initial stage (2162)

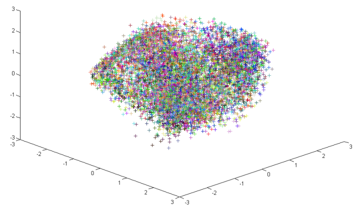

(e) Middle stage (216)

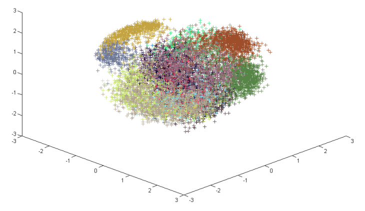

(f) Final stage (100)

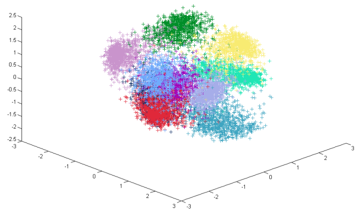

(g) Initial stage (2232)

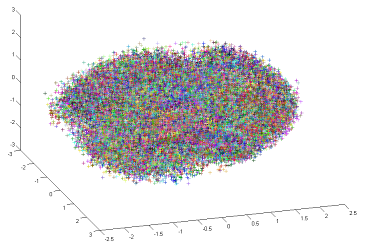

(h) Middle stage (22)

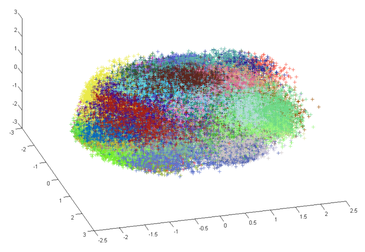

(i) Final stage (10)


(j) Initial stage (1762)


(k) Middle stage (22)


(l) Final stage (10)


(m) Initial stage (11521)


(n) Middle stage (115)

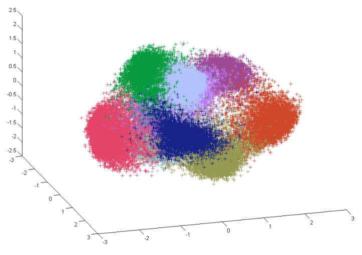
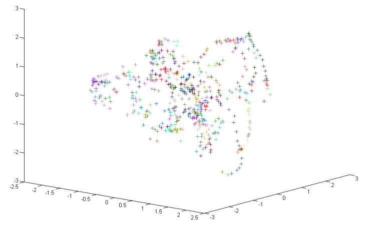
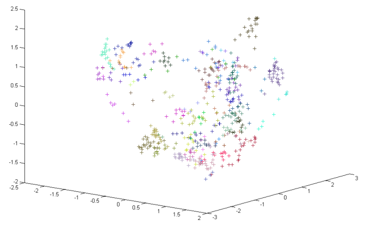
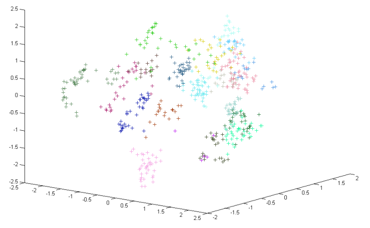
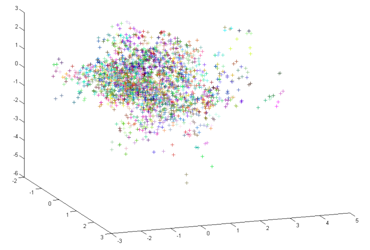

(o) Final stage (10)

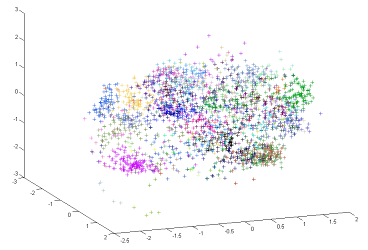
Figure 6: Learned representations at different stages on five datasets. From top to bottom, they are *COIL20*, *COIL100*, *USPS* and *MNIST-test* and *MNIST-full*. The first column are image intensities. For *MNIST-test*, we show another view point different from Fig.1 in the main paper.

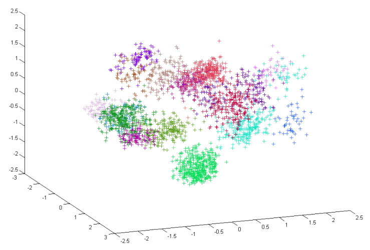

(a) Initial stage (188)

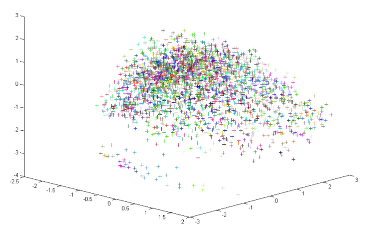

(b) Middle stage (60)

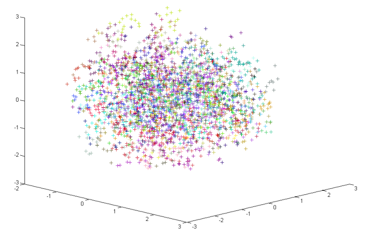

(c) Final stage (20)

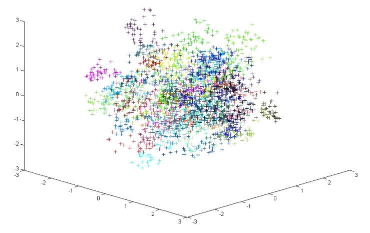

(d) Initial stage (775)

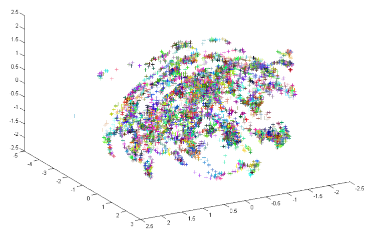

(e) Middle stage (128)

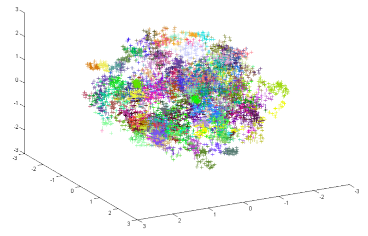

(f) Final stage (20)


(g) Initial stage (775)


(h) Middle stage (200)


(i) Final stage (68)


(j) Initial stage (2814)


(k) Middle stage (300)

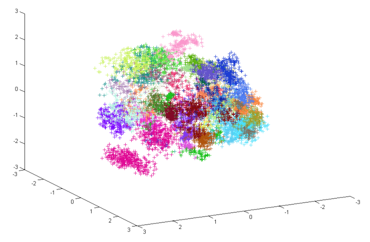

(l) Final stage (41)

Figure 7: Learned representations as different stages on four datasets. From top to bottom, they are *UMist*, *FRGC*, *CMU-PIE* and *YTF*. The first column are image intensities.