Neural Collaborative Subspace Clustering

Tong Zhang [1 2] **Pan Ji** [3] **Mehrtash Harandi** [4] **Wenbing Huang** [5] **Hongdong Li** [2]

Abstract

We introduce the Neural Collaborative Subspace Clustering, a neural model that discovers clusters of data points drawn from a union of low-dimensional subspaces. In contrast to previous attempts, our model runs without the aid of spectral clustering. This makes our algorithm one of the kinds that can gracefully scale to large datasets. At its heart, our neural model benefits from a classifier which determines whether a pair of points lies on the same subspace or not. Essential to our model is the construction of two affinity matrices, one from the classifier and one based on a notion of subspace self-expressiveness, to supervise training in a collaborative scheme. We thoroughly assess and contrast the performance of our model against various state-of-the-art clustering algorithms including deep subspace-based ones.

1. Introduction

In this paper, we tackle the problem of subspace clustering, where the aim is to cluster data points drawn from a union of low-dimensional subspaces in an unsupervised manner. Subspace Clustering (SC) has achieved great success in various computer vision tasks, such as motion segmentation (Kanatani, 2001; Elhamifar & Vidal, 2009; Ji et al., 2014a; 2016), face clustering (Ho et al., 2003; Elhamifar & Vidal, 2013) and image segmentation (Yang et al., 2008; Ma et al., 2007).

Majority of the SC algorithms (Yan & Pollefeys, 2006; Chen & Lerman, 2009; Elhamifar & Vidal, 2013; Liu et al., 2013; Wang et al., 2013; Lu et al., 2012; Ji et al., 2015; You et al., 2016a) rely on the linear subspace assumption to construct the affinity matrix for spectral clustering. However, data do not naturally conform to linear models, which in turns results in the development of non-linear SC techniques. Kernel methods (Chen et al., 2009; Patel et al., 2013; Patel & Vidal, 2014; Yin et al., 2016; Xiao et al., 2016; Ji et al., 2017a) can be employed to implicitly map data to higher dimensional spaces, hoping that data fit better to linear models in the resulting spaces. That said, it is not straightforward to identify the right kernel function and its parameters. The use of deep neural networks as non-linear mapping functions to determine subspace friendly latent spaces has formed the latest developments in the field with promising results (Ji et al., 2017b; Peng et al., 2016).

Despite significant improvements, SC algorithms still resort to spectral clustering which in hindsight requires constructing an affinity matrix. This step, albeit effective, hampers the scalability as it takes $O(n^2)$ memory and $O(kn^2)$ computations for storing and decomposing the affinity matrix for n data points and k clusters. There are several attempts to resolve the scalability issue. For example, You et al. (2016a) accelerate the construction of the affinity matrix using orthogonal matching pursuit; Zhang et al. (2018) resort to k-subspace clustering to avoid generating the affinity matrix. However, the issue is not fully addressed either due to the use of spectral clustering (You et al., 2016a), or mitigates but at the cost of performance (Zhang et al., 2018).

In this paper, we propose a neural structure to improve the performance of subspace clustering while being mindful to the scalablity issue. To this end, we first formulate subspace clustering as a classification problem, which in turn removes the spectral clustering step from the computations. Our neural model is comprised of two modules, one for classification and one for affinity learning. Both modules collaborate during learning, hence the name "Neural Collaborative Subspace Clustering". During training and in each iteration, we use the affinity matrix generated by the subspace self-expressiveness to supervise the affinity matrix computed from the classification part. Concurrently, we make use of the classification part to improve self-expressiveness to build a better affinity matrix through collaborative optimization.

We evaluate our algorithm on three datasets , namely MNIST (LeCun et al., 1998), Fashion-MNIST (Xiao et al., 2017), and the Stanford Online Products dataset (Oh Song et al., 2016) which exhibit different levels of difficulty. Our empirical study shows the superiority of the proposed algo-

[1]Motovis Australia Pty Ltd [2]Australian National University [3]NEC Labs America [4]Monash University [5]Tencent AI Lab. Correspondence to: Pan Ji <peterji1990@gmail.com>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

rithm over several state-of-the-art baselines including deep subspace clustering techniques.

2. Related Work

2.1. Subspace Clustering

Linear subspace clustering encompasses a vast set of techniques, among them, spectral clustering algorithms are more favored to cluster high-dimensional data (Vidal, 2011). One of the crucial challenges in employing spectral clustering on subspaces is the construction of an appropriate affinity matrix. We can categorize the algorithms based on the way the affinity matrix is constructed into three main groups: factorization based methods (Gruber & Weiss, 2004; Mo & Draper, 2012), model based methods (Chen & Lerman, 2009; Ochs et al., 2014; Purkait et al., 2014), self-expressiveness based methods (Elhamifar & Vidal, 2009; Ji et al., 2014b; Liu et al., 2013; Vidal & Favaro, 2014).

The latter, *i.e.*, self-expressiveness based methods have become dominant due to their elegant convex formulations and existence of theoretical analysis. The basic idea of subspace self-expressiveness is that one point can be represented in terms of a linear combination of other points from the same subspace. This leads to several advantages over other methods: (i) it is more robust to noise and outliers; (ii) the computational complexity of the self-expressiveness affinity does not grow exponentially with the number of subspaces and their dimensions; (iii) it also exploits the non-local information without the need of specifying the size of the neighborhood (*i.e.*, the number of nearest neighbors as usually used for identifying locally linear subspaces (Yan & Pollefeys, 2006; Zhang et al., 2012)).

The assumption of having linear subspaces does not necessarily hold for practical problems. Kernel sparse subspace clustering (KSSC) (Patel & Vidal, 2014) and Kernel Low-rank representation (Xiao et al., 2016) benefit from pre-defined kernel functions, such as Radial Basis Functions (RBF), to cast the problem in high-dimensional (possibly infinite) reproducing kernel Hilbert spaces. However, it is still not clear how to choose proper kernel functions for different datasets and there is no guarantee that the feature spaces generated by kernel tricks are well-suited to linear subspace clustering.

Recently, Deep Subspace Clustering Networks (DSC-Net) (Ji et al., 2017b) are introduced to tackle the non-linearity arising in subspace clustering, where data is non-linearly mapped to a latent space with convolutional auto-encoders and a new *self-expressive layer* is introduced between the encoder and decoder to facilitate an end-to-end learning of the affinity matrix. Although DSC-Net outperforms traditional subspace clustering methods by large, its computational cost and memory footprint can become overwhelming even for mid-size problems.

There are a few attempts to tackle the scalability of subspace clustering. The SSC-Orthogonal Matching Pursuit (SSC-OMP) (You et al., 2016b) replaces the large scale convex optimization procedure with the OMP algorithm to represent the affinity matrix. However, SSC-OMP sacrifices the clustering performance in favor of speeding up the computations, and it still may fail when the number of data points is very large. k-Subspace Clustering Networks (k-SCN) (Zhang et al., 2018) is proposed to make subspace clustering applicable to large datasets. This is achieved via bypassing the construction of affinity matrix and consequently avoiding spectral clustering, and introducing the iterative method of k-subspace clustering (Tseng, 2000; Bradley & Mangasarian, 2000) into a deep structure. Although k-SCN develops two approaches to update the subspace and networks, it still shares the same drawbacks as iterative methods, for instance, it requires a good initialization, and seems fragile to outliers.

2.2. Model fitting

In learning theory, distinguishing outliers and noisy samples from clean ones to facilitate training is an active research topic. For example, Random Sample Consensus (RANSAC) (Fischler & Bolles, 1981) is a classical and well-received algorithm for fitting a model to a cloud of points corrupted by noise. Employing RANSAC on subspaces (Yang et al., 2006) in large-scale problems does not seem to be the right practice, as RANSAC requires a large number of iterations to achieve an acceptable fit.

Curriculum Learning (Bengio et al., 2009) begins learning a model from easy samples and gradually adapting the model to more complex ones, mimicking the cognitive process of humans. Ensemble Learning (Dietterich, 2000) tries to improve the performance of machine learning algorithms by training different models and then to aggregate their predictions. Furthermore, distilling the knowledge learned from large deep learning models can be used to supervise a smaller model (Hinton et al., 2015). Although Curriculum Learning, Ensemble Learning and distilling knowledge are notable methods, adopting them to work on problems with limited annotations, yet aside the unlabeled scenario, is far-from clear.

2.3. Deep Clustering

Many research papers have explored clustering with deep neural networks. Deep Embedded clustering (DEC) (Xie et al., 2016) is one of the pioneers in this area, where the authors propose to pre-train a stacked auto-encoder (SAE) (Bengio et al., 2007) and fine-tune the encoder with a regularizer based on the student-t distribution to achieve cluster-friendly embeddings. On the downside, DEC seems

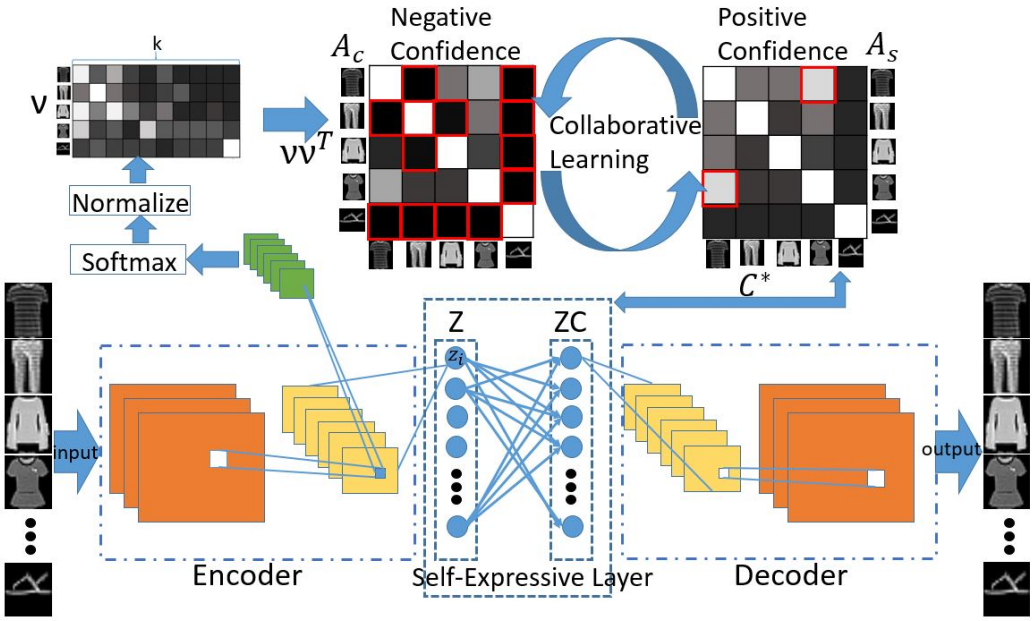

Figure 1. The Neural Collaborative Subspace Clustering framework. The affinity matrix generated by self-expressive layer, $\mathbf{A}_s$, and classifier, $\mathbf{A}_c$, supervise each other by selecting the high confidence parts for training. Red squares in $\mathbf{A}_s$ highlight positive pairs (belonging to the same subspace). Conversely, red squares in $\mathbf{A}_c$ highlight the negative pairs (belonging to different subspace). Affinities are coded with shades, meaning that light gray denotes large affinity while dark shades representing small affinities.

to be sensitive to the structure and initialization of the network. Various forms of Generative Adversarial Network (GAN) are employed for clustering such as Info-GAN (Chen et al., 2016) and ClusterGAN (Mukherjee et al., 2018), both of which intend to encourage discriminative features in the latent space to simultaneously generate and cluster images. The Deep Adaptive image Clustering (DAC) (Chang et al., 2017) uses fully convolutional neural nets (Springenberg et al., 2014) as initialization to perform self-supervised learning, and achieves remarkable results on various clustering benchmarks. However, sensitivity to the network structure seems again to be a concern for DAC.

In this paper, we formulate subspace clustering as a binary classification problem through collaborative learning of two modules, one for image classification and the other for subspace affinity learning. Instead of performing spectral clustering on the whole dataset, we train our model in a stochastic manner, leading to a scalable paradigm for subspace clustering.

3. Proposed Method

To design a scalable SC algorithm, our idea is to identify whether a pair of points lies on the same subspace or not. Upon attaining such knowledge (for a large-enough set of pairs), a deep model can optimize its weights to maximize such relationships (being on subspaces or not). This can be nicely cast as a binary classification problem. However, since ground-truth labels are not available to us, it is not obvious how such a classifier should be built and trained.

In this work, we propose to make use of two confidence maps (see Fig. 1 for a conceptual visualization) as a supervision signal for SC. To be more specific, we make use of the concept of self-expressiveness to identify positive pairs, *i.e.*, pairs that lie on the same subspaces. To identify negative pairs, pairs that do not belong to the same subspace, we benefit from a negative confidence map. This, as we will show later, is due to the fact that the former can confidently mine positive pairs (with affinity close to 1) while the latter is good at localizing negative pairs (with affinity close to 0). The two confidence maps, not only provide the supervision signal to optimize a deep model, but act collaboratively as partial supervisions for each other.

3.1. Binary Classification

Given a dataset with n points $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$ from k clusters, we aim to train a classifier to predict class labels for data points without using any ground-truth labels. To this end, we propose to use a multi-class classifier which consists of a few convolutional layers (with non-linear rectifiers) and a softmax output layer. We then convert the resulting outputs to an affinity-based binary classifier by

$$\mathbf{A}_c(i,j) = \nu_i \nu_j^\top, \tag{1}$$

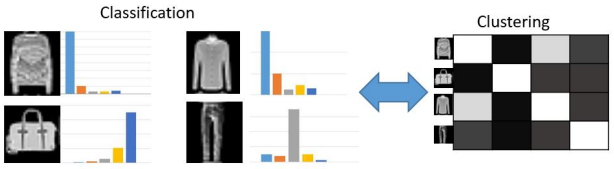

Figure 2. By normalizing the feature vectors after softmax function and computing their inner product, an affinity matrix can be generated to encode the clustering information.

where $\nu_i \in \mathbb{R}^k$ is a k dimensional prediction row vector after ℓ_2 normalization. Ideally, when ν_i is one-hot, $\mathbf{A}_c$ is a binary matrix encoding the confidence of data points belonging to the same cluster. So if we supervise the classifier using $\mathbf{A}_c$, we will end up with a binary classification problem. Also note that $\mathbf{A}_c(i,j)$ can be interpreted as the cosine similarity between softmax prediction vectors of $\mathbf{x}_i$ and $\mathbf{x}_j$, which has been widely used in different contexts (Nguyen & Bai, 2010). However, unlike the cosine similarity which lies in $[-1,1]$, $\mathbf{A}_c(i,j)$ lies within $[0,1]$, since the vectors are normalized by softmax and ℓ_2 norm. We illustrate this in Fig. 2.

3.2. Self-Expressiveness Affinity

Subspace self-expressiveness can be worded as: one data point $\mathbf{x}_i$ drawn from linear subspaces $\{\mathcal{S}_i\}_{i=1}^k$ can be represented by a linear combination of other points from the same subspace. Stacking all the points into columns of a data matrix $\mathbf{X}$, the self-expressiveness can be simply described as $\mathbf{X} = \mathbf{X}\mathbf{C}$, where $\mathbf{C}$ is the coefficient matrix.

It has been shown (*e.g.*, (Elhamifar & Vidal, 2009; Ji et al., 2014b)) that by minimizing certain norms of coefficient matrix $\mathbf{C}$, a block-diagonal structure (up to permutations) on $\mathbf{C}$ can be achieved. This translates into $c_{ji} \neq 0$ only if data points coming from the same subspace. Therefore, the loss function of learning the affinity matrix can be written as:

$$\min_{\mathbf{C}} \|\mathbf{C}\|_p \quad \text{s.t.} \quad \mathbf{X} = \mathbf{X}\mathbf{C},\ \text{diag}(\mathbf{C}) = \mathbf{0}, \tag{2}$$

where $\|\cdot\|_p$ is a matrix norm. For example, Sparse Subspace Clustering (SSC) (Elhamifar & Vidal, 2009) uses the ℓ_1 norm, Low Rank Representation (LRR) models (Liu & Yan, 2011; Vidal & Favaro, 2014) opt for the nuclear norm, and Efficient Dense Subspace Clustering (Ji et al., 2014b) utilizes the ℓ_2 norm. To handle data corruption, a relaxed version can be derived as:

$$\begin{aligned} \mathbf{C}^* = &\arg\min_{\mathbf{C}} \|\mathbf{C}\|_p + \frac{\lambda}{2}\|\mathbf{X} - \mathbf{X}\mathbf{C}\|_F^2 \\ &\text{s.t.} \quad \text{diag}(\mathbf{C}) = \mathbf{0}. \end{aligned} \tag{3}$$

Here, λ is a weighting parameter balancing the regularization term and the data fidelity term.

To handle subspace non-linearity, one can employ convolutional auto-encoders to non-linearly map input data $\mathbf{X}$ to a latent space $\mathbf{Z}$, and transfer the self-expressiveness into a linear layer (without non-linear activation and bias parameters) named *self-expressive layer* (Ji et al., 2017b) (see the bottom part of Fig. 1). This enables us to learn the subspace affinity $\mathbf{A}_s$ in an end-to-end manner using the weight parameters $\mathbf{C}^*$ in the self-expressive layer:

$$\mathbf{A}_s(i,j) = \begin{cases} (|c_{ij}^*| + |c_{ji}^*|)/2c_{\max} & \text{if } i \neq j, \\ 1 & \text{if } i = j, \end{cases} \tag{4}$$

where $c_{\max}$ is the maximum absolute value of off-diagonal entries of the current row. Note that $\mathbf{A}_s(i,j)$ then lies within $[0,1]$.

3.3. Collaborative Learning

The purpose of collaborative learning is to find a principle way to benefit from the strengths of different modules. The classification module and self-expressive module distill different information in the sense that the former tends to extract more abstract and discriminative features while the latter focuses more on capturing the pairwise correlation between data samples. As alluded to earlier, ideally, the subspace affinity $\mathbf{A}_s(i,j)$ is nonzero only if $\mathbf{x}_i$ and $\mathbf{x}_j$ are from the same subspace, which means that $\mathbf{A}_s$ can be used to mine similar pairs (*i.e.*, positive samples). On the other hand, if the classification affinity $\mathbf{A}_c(i,j)$ is close to zero, it indicates strongly that $\mathbf{x}_i$ and $\mathbf{x}_j$ are dissimilar (*i.e.*, negative sample). Therefore, we carefully design a mechanism to let both modules collaboratively supervise each other.

Given $\mathbf{A}_c$ and $\mathbf{A}_s$, we pick up the high-confidence affinities as supervision for training. We illustrate this process in Fig. 1. The "positive confidence" in Fig. 1 denotes the ones from the same class, and the "negative confidence" represents the ones from different classes. As such, we select high affinities from $\mathbf{A}_s$ and small affinities from $\mathbf{A}_c$, and formulate the collaborative learning problem as:

$$\begin{aligned} &\min_{\mathbf{A}_s, \mathbf{A}_c} \mathbf{\Omega}(\mathbf{A}_s, \mathbf{A}_c, l, u) = \\ &L_{pos}(\mathbf{A}_s, \mathbf{A}_c, u) + \alpha L_{neg}(\mathbf{A}_c, \mathbf{A}_s, l), \end{aligned} \tag{5}$$

where the $L_{pos}(\mathbf{A}_s, \mathbf{A}_c, u)$ and $L_{neg}(\mathbf{A}_c, \mathbf{A}_s, l)$ denote the cross-entropy function with sample selection process, which can be defined as follows:

$$\begin{aligned} L_{pos}(\mathbf{A}_s, \mathbf{A}_c, u) &= H(\mathbf{M}_s || \mathbf{A}_c) \\ \text{s.t} \quad \mathbf{M}_s &= \mathbb{1}(\mathbf{A}_s > u), \end{aligned} \tag{6}$$

and

$$\begin{aligned} L_{neg}(\mathbf{A}_c, \mathbf{A}_s, l) &= H(\mathbf{M}_c || (\mathbf{1} - \mathbf{A}_s)) \\ \text{s.t} \quad \mathbf{M}_c &= \mathbb{1}(\mathbf{A}_c < l), \end{aligned} \tag{7}$$

where $\mathbb{1}(\cdot)$ is the indicator function returning 1 or 0, $\{u, l\}$ are thresholding parameters, and H is the entropy function, defined as $H(\mathbf{p}||\mathbf{q}) = \sum_j p_j \log(q_j)$.

Note that the cross-entropy loss is a non-symmetric metric function, where the former probability serves a supervisor to the latter. Therefore, in Eqn. (6), the subspace affinity matrix $\mathbf{A}_s$ is used as the "teacher" to supervise the classification part (the "student"). Conversely, in Eqn. (7), the classification affinity matrix $\mathbf{A}_c$ works as the "teacher" to help the subspace affinity learning module to correct negative samples. That said, to better facilitate gradient back-propagation between two modules, we can approximate indicator function by replacing $\mathbf{M}_s$ with $\mathbf{M}_s\mathbf{A}_s$ in Eqn. (6) and $\mathbf{M}_c$ with $\mathbf{M}_c(1-\mathbf{A}_c)$ in Eqn. (7). The weight parameter α in Eqn. 5, called collaboration rate, controls the contributions of L_{pos} and L_{neg}. It can be set as the ratio of the number of positive confident pairs and the negative confident pairs, or tuned for better performance.

3.4. Overall Model

After introducing all the building blocks of this work, we now explain how to jointly organize them in a network and train it with a carefully defined loss function. As shown in Fig. 1, our network is composed of four main parts: (i) a convolutional encoder that maps input data $\mathbf{X}$ to a latent representation $\mathbf{Z}$; (ii) a linear self-expressive layer which learns the subspace affinity through weights $\mathbf{C}$; (iii) a convolutional decoder that maps the data after self-expressive layer, *i.e.*, $\mathbf{ZC}$, back to the input space $\hat{\mathbf{X}}$; (iv) a multi-class classifier that outputs k dimensional prediction vectors, with which a classification affinity matrix can be constructed. Our loss function consists of two parts, *i.e.*, collaborative learning loss and subspace learning loss:

$$\begin{aligned}\mathbb{L}(\mathbf{X};\Theta) = &L_{sub}(\mathbf{X};\Theta,\mathbf{A}_s)\\ &+\lambda_{cl}\boldsymbol{\Omega}(\mathbf{X},u,l;\Theta,\mathbf{A}_s,\mathbf{A}_c).\end{aligned} \tag{8}$$

Here, Θ denotes the parameters of the neural network and λ_{cl} is the weight of the collaborative learning loss. The $L_{sub}(\mathbf{X};\Theta,\mathbf{A}_s)$ is the loss to train the affinity matrix through self-expressive layer. Combining Eqn. (3) and the reconstruction loss of the convolutional auto-encoder, we arrive at:

$$\begin{aligned}L_{sub}(\mathbf{X};\Theta,\mathbf{A}_s) = &\|\mathbf{C}\|_F^2 + \frac{\lambda_1}{2}\|\mathbf{Z}-\mathbf{ZC}\|_F^2\\ &+\frac{1}{2}\|\mathbf{X}-\hat{\mathbf{X}}\|_F^2\\ &\text{s.t.}\quad \text{diag}(\mathbf{C}) = \mathbf{0},\end{aligned} \tag{9}$$

where $\mathbf{A}_s$ is a function of $\mathbf{C}$ as defined in (4).

After the training stage, we no longer need to run the decoder and self-expressive layer to infer the labels. We can directly infer the cluster labels through the classifier output ν:

$$s_i = \arg\max_h \nu_{ih}, \quad h = 1,\ldots,k, \tag{10}$$

where s_i is the cluster label of image $\mathbf{x}_i$.

Algorithm 1 Neural Collaborative Subspace Clustering

Input: dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, number of clusters k, sample selection threshold u and l , learning rate of auto-encoder η_{ae}, and learning rate of other parts η
Initialization: Pre-train the Convolutional Auto-encoder by minimizing the reconstruction error.
repeat
  For every mini-batch data
  Train auto-encoder with *self-expressive layer* to minimize loss function in Eqn. (9) to update $\mathbf{A}_s$.
  Forward the batch data through the classifier to get $\mathbf{A}_c$.
  Do sample selection and collaborative learning through minimizing Eqn. (5) to update the classifier.
  Jointly update all the parameters by minimizing Eqn. (8).
until reach the maximum epochs
Output: Get the cluster s_i for all samples by Eqn. (10)

4. Training

In this section, we provide more details about how training will be done. Similarly to other auto-encoder based clustering methods, we pre-train the auto-encoder by minimizing the reconstruction error to get a good initialization of latent space for subspace clustering.

According to (Elhamifar & Vidal, 2009), the solution to (2) is guaranteed to have a block-diagonal structure (up to certain permutations) under the assumption that the subspaces are independent. To account for this, we make sure that the dimensionality of the latent space ($\mathbf{Z}$) is greater than (the subspace intrinsic dimension) $\times$ (number of clusters) [1]. In doing so, we make use of the stride convolution to downsample the images while increasing the number of channels over layers to keep the dimensionality of the latent space large. Since we have pre-trained the auto-encoder, we use a smaller learning rate in the auto-encoder when the collaborative learning is performed. Furthermore, compared to DSC-Net or other spectral clustering based methods which require to perform involved techniques to post process the affinity matrix, we only need to compute $\mathbf{A}_s = (|\mathbf{C}^*| + |\mathbf{C}^{*T}|)/2$ and normalize it (divided by the largest value in each row and assign 1.0 to the diagonal entries) to ensure the subspace

[1] Note that our algorithm does not require specifying subspace intrinsic dimensions explicitly. Empirically, we found a rough guess of the subspace intrinsic dimension would suffice, *e.g.*, in our experiments, we set it to 9.

affinity matrix lies in the same range with the classification affinity matrix.

We adopt a three-stage training strategy: first, we train the auto-encoder together with the self-expressive layer using the loss in (9) to update the subspace affinity $\mathbf{A}_s$; second, we train the classifier to minimize Eqn. (5); third, we jointly train the whole network to minimize the loss in (8). All these details are summarized in Algorithm 1.

5. Experiments

We implemented our framework with Tensorflow-1.6 (Abadi et al., 2016) on an Nvidia TITAN X GPU. We mainly evaluate our method on three standard datasets, *i.e.*, MNIST, Fashion-MNIST and a subset of the Stanford Online Products dataset. All of these datasets are considered challenging for subspace clustering as it is hard to perform spectral clustering on datasets of this scale. The number of clusters k is set to 10 as input to all competing algorithms. For all the experiments, we pre-train the convolutional auto-encoder for 60 epochs with a learning rate 1.0×10^{-3}.

The hyper parameters in our loss function are easy to tune. λ_1 in Eqn. (9) controls self-expressiveness, and it also affects the choice of u and l in Eqn. (8). If λ_1 is set large, the coefficient in the affinity matrix will become larger, and in that case the u should be increased. The other parameter λ_{cl} balances the cost of subspace clustering and collaborative learning, and we usually set it to keep these two terms in the same scale to treat them equally. We keep the $\lambda_1 = 10$ in all the experiments, and slightly change the l and u for each dataset.

In all experiments, we employ a special residual convolutional block (He et al., 2016a), which doesn't contain batch normalization layer. This is because we empirically observed that batch normalization would negatively affect the subspace structure in the latent space. We use the Rectified Linear Unit (ReLu) as the non-linear activation in the blocks. Hence, each residual block has two convolutional kernels and two ReLu layers as activation function.

Since no ground truth label is available to the algorithm, we chose to use a larger batch size as compared to the practice in supervised learning. This makes the training stable and robust. Specifically, we set the batch size to 5000, and used Adam (Kingma & Ba, 2014), an adaptive momentum based gradient descent method to minimize the loss in all our experiments. We set the learning rate to 1.0×10^{-5} for the auto-encoder and 1.0×10^{-3} for other parts of the network in all training stages. In each iteration, we train the self-expressiveness layer for 50 iterations, followed by 50 iterations' training along with classifier, and fine-tune the whole network by 10 iterations.

Baseline Methods. We use various clustering methods as the baseline methods including the classic clustering methods, subspace clustering methods, deep clustering methods, and GAN based methods. Specifically, we have the following baselines:

- classic methods: K-Means (Lloyd, 1982) (KM), K-Means with our CAE-feature (CAE-KM) and SAE-feature (SAE-KM);
- subspace clustering algorithms: sparse subspace clustering (SSC) (Elhamifar & Vidal, 2013), Low Rank Representation (LRR) (Liu et al., 2013), Kernel Sparse Subspace Clustering (KSSC) (Patel & Vidal, 2014), Deep Subspace Clustering Network (DSC-Net) (Ji et al., 2017b), and k-Subspace Clustering Network (k-SCN) (Zhang et al., 2018);
- deep clustering methods: Deep Embedded Clustering (DEC) (Xie et al., 2016), Deep Clustering Network (DCN) (Yang et al., 2017), and Deep Adaptive image Clustering (DAC) (Chang et al., 2017);
- GAN based clustering methods: Info-GAN (Chen et al., 2016) and ClusterGAN (Mukherjee et al., 2018).

Evaluation Metric. For all quantitative evaluations, we make use of the unsupervised clustering accuracy rate, defined as

$$\text{ACC } \% = \max_{\Pi} \frac{\sum_{i=1}^{n} \mathbb{1}(y_i = \Pi(c_i))}{n} \times 100\% \,. \quad (11)$$

where y_i is the ground-truth label, c_i is the subspace assignment produced by the algorithm, and Π ranges over all possible one-to-one mappings between subspaces and labels. The mappings can be efficiently computed by the Hungarian algorithm. We also use normalized mutual information (NMI) as the additional quantitative standard. NMI scales from 0 to 1, where a smaller value means less correlation between predicted labels and ground truth labels. Another quantitative metric is the adjusted Rand index (ARI), which is scaled between 0 and 1. It computes a similarity between two clusters by considering all pairs of samples and counting pairs that are assigned to the same or different clusters. The larger the ARI, the better the clustering performance.

5.1. MNIST

MNIST consists of 70000 hand-written digit images of size 28×28. Subspace non-linearity arises naturally for MNIST due to the variance of scale, thickness and orientation among all the images of each digit. We thus apply our method on this dataset to see how well it can handle this type of subspace non-linearity.

In this experiment, we use a convolutional layer followed by three pre-activation residual blocks (He et al., 2016b) without batch normalization as encoder and a self-expressive layer in between encoder and decoder for the subspace affin-

ity learning module. The kernel size is fixed as 3 and the number of channel is 20, 30 and 40 for each block. For the classification module, we connect three more convolutional layers after the encoder layers with kernel size 2, and one convolutional layer with kernel size 1 to output the feature vector. Meanwhile, we also tried simple version auto-encoder (same as DSC) and find that the classification module can not share the layers with the encoder, but need to use an independent small network because of its low capacity. For the threshold parameters u and l, we set them to 0.7 and 0.1 respectively in the first epoch of training, and increase u to 0.9 afterwards.

We report the clustering results of all competing methods in Table 1. Since spectral clustering based methods (*i.e.*, SSC-CAE, LRR-CAE, KSSC-CAE, DSC-Net) can not apply on the whole dateset (due to memory and computation issues), we only use the 10000 samples to show how they perform. As shown in Table 1, subspace algorithms do not perform very well even on 10000 samples. Although the DSC-Net is trapped by training the self-expressive layer, it outperforms other subspace clustering algorithm, which shows the potential of learning subspace structure using neural networks. On the other hand, DEC, DCN, k-SCN and our algorithm are all based on auto-encoder, which learn embeddings with different metrics to help clustering. However, our classification module boosts the performance by making the latent space of auto-encoder more discriminative. Therefore, our algorithm incorporates the advantage of different classes, *e.g.*, self-expressiveness, nonlinear mapping and discriminative features, and achieves the best results among all the algorithms thanks to the collaborative learning paradigm.

	ACC(%)	NMI(%)	ARI(%)
CAE-KM	51.00	44.87	33.52
SAE-KM	81.29	73.78	67.00
KM	53.00	50.00	37.00
DEC	84.30	80.00	75.00
DCN	83.31	80.86	74.87
SSC-CAE	43.03	56.81	28.58
LRR-CAE	55.18	66.54	40.57
KSSC-CAE	58.48	67.74	49.38
DSC-Net	65.92	73.00	57.09
k-SCN	87.14	78.15	75.81
Ours	**94.09**	**86.12**	**87.52**

Table 1. Clustering results of baseline methods and our collaborative scheme on the MNIST dataset. For all the metrics, the larger value is better. The best results are shown in bold.

5.2. Fashion-MNIST

Same as in MINIST, Fashion-MNIST also has 70000 images of size 28×28. It consists of various types of fashion products. Unlike MNIST, every class in Fashion-MNIST has different styles with different gender groups (*e.g.*, men, women, kids and neutral). As shown in Fig. 3, the high similarity between several classes (such as { Pullover, Coat, Shirt}, { T-shirt, Dress }) makes the clustering more difficult. Compared to MNIST, the Fashion-MNIST clearly poses more challenges for unsupervised clustering.

Figure 3. Examples of the Fashion-MNIST dataset

On Fashion-MNIST, we employ a network structure with one convolutional layer, followed by pre-activation residual blocks without batch normalization in the encoder, and with a symmetric structure in the decoder. As the complexity of dataset increases, we also raise the dimensionality of ambient space to better suit self-expressiveness, and increase capacity for the classification module. For all convolutional layers, we keep kernel size as 3 and set the number of channels to 10-20-30-40 in the encoder, where three residual blocks are employed for 20, 30 and 40 repsectively.

We report the clustering results of all methods in Table 2, where we can clearly see that our framework outperforms all the baselines by a large margin including the best-performing baseline k-SCN. Specifically, our method improves over the second best one (*i.e.*, k-SCN) by 8.36%, 4.2% and 9% in terms of accuracy, NMI and ARI. We can clearly observe from Fig. 4 that the latent space of our framework, which is collaboratively learned by subspace and classification modules, has strong subspace structure and also keeps each subspace discriminative. For subspace clustering methods and similar to the previous experiment, we used only 10000 samples. DSC-Net does not drop a lot while the performance of other subspace clustering algorithms decline sharply compared with their performance on MNIST.

5.3. Stanford Online Products

The Stanford Online Products dataset is mainly used for supervised metric learning, and it is thus considered to be difficult for unsupervised tasks. Compared to the previous two datasets, the challenging aspects of this dataset include: (i) the product images contain various backgrounds, from

Figure 4. The visualization of the latent space of our collaborative scheme through dimensionality reduction by PCA.

	ACC(%)	NMI(%)	ARI(%)
SAE-KM	54.35	58.53	41.86
CAE-KM	39.84	39.80	25.93
KM	47.58	51.24	34.86
DEC	59.00	60.10	44.60
DCN	58.67	59.4	43.04
DAC	61.50	63.20	50.20
ClusterGAN	63.00	64.00	50.0
InfoGAN	61.00	59.00	44.20
SSC-CAE	35.87	18.10	13.46
LRR-CAE	34.48	25.41	10.33
KSSC-CAE	38.17	19.73	14.74
DSC-Net	60.62	61.71	48.20
k-SCN	63.78	62.04	48.04
Ours	**72.14**	**68.60**	**59.17**

Table 2. Clustering results of baseline methods and our collaborative scheme on the Fashion-MNIST dataset. For all the metrics, the larger value is better. The best results are shown in bold.

pure white to real world environments; (ii) each product has various shapes, colors, scales and view angles; (iii) products across different classes may look similar to each other. To create a manageable dataset for clustering, we manually pick 10 classes out of 12 classes, with around 1000 images per class (10056 images in total), and then re-scale them to 32×32 gray images, as shown in Fig. 5.

Our networks for this dataset has one convolutional layer with 10 channels, followed by three pre-activation residual blocks without batch normalization, which have 20, 30 and 10 channels respectively.

Table 3 shows the performance of all algorithms on this dataset. Due to the challenging nature of the dataset, most deep learning based methods fail to produce reasonable results. For example, DEC and DCN perform below their initialization, and DAC cannot self-supervise its model to achieve a better result. Similarly, infoGAN also fails to find clustering pattern. Along with KSSC and DSC-Net, our

	ACC (%)	NMI (%)	ARI (%)
DEC	22.89	12.10	3.62
DCN	21.30	8.40	3.14
DAC	23.10	9.80	6.15
InfoGAN	19.76	8.15	3.79
SSC-CAE	12.66	0.73	0.19
LRR-CAE	22.35	**17.36**	4.04
KSSC-CAE	26.84	15.17	7.48
DSC-Net	26.87	14.56	**8.75**
k-SCN	22.91	16.57	7.27
Ours	**27.5**	13.78	7.69

Table 3. Clustering results of baseline methods and our collaborative scheme on the Stanford product dataset.

Figure 5. Examples of the Stanford Online Products Dataset

algorithm achieves the best results compared, especially in comparison to the deep learning based algorithms, hinting a better handling of non-linearity. We can also observe that subspace based clustering algorithms perform better than clustering methods. This illustrates the benefits of the underlying subspace assumption in high dimensional regimes.

In summary, compared to other deep learning methods, our framework is not sensitive to the architecture of the neural network, as long as the dimensionality meets the requirement of subspace self-expressiveness. Furthermore, the two modules in our network progressively improve the performance in a collaborative way, which is both effective and efficient.

6. Conclusion

In this work, we have introduced a novel collaborative learning for unsupervised subspace clustering. To this end, we have made use of the complementary property of the classifier-induced affinities and the subspace-based affinities to train a deep model. Our network can be trained in stochastic manner and can directly predict the clustering labels (once trained) without the need of performing spectral clustering. The experiments in our paper have shown that the proposed method outperforms the-state-of-art algorithms by a large margin on image clustering tasks, validating the proposed framework.

Acknowledgement

HL was supported in part by Australia Centre for Robotic Vision (ACRV) , and the Australian Research Council (ARC) grants (CE140100016, DP190102261).

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv:1603.04467*, 2016.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. Greedy layer-wise training of deep networks. In *NeurIPS*, pp. 153–160, 2007.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *ICML*, pp. 41–48. ACM, 2009.

Bradley, P. S. and Mangasarian, O. L. K-plane clustering. *Journal of Global Optimization*, 16(1):23–32, 2000.

Chang, J., Wang, L., Meng, G., Xiang, S., and Pan, C. Deep adaptive image clustering. In *ICCV*, pp. 5880–5888. IEEE, 2017.

Chen, G. and Lerman, G. Spectral curvature clustering (SCC). *IJCV*, 81(3):317–330, 2009.

Chen, G., Atev, S., and Lerman, G. Kernel spectral curvature clustering (KSCC). In *ICCV Workshops*, pp. 765–772. IEEE, 2009.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NeurIPS*, pp. 2172–2180, 2016.

Dietterich, T. G. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pp. 1–15. Springer, 2000.

Elhamifar, E. and Vidal, R. Sparse subspace clustering. In *CVPR*, pp. 2790–2797, 2009.

Elhamifar, E. and Vidal, R. Sparse subspace clustering: Algorithm, theory, and applications. *TPAMI*, 35(11):2765–2781, 2013.

Fischler, M. A. and Bolles, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

Gruber, A. and Weiss, Y. Multibody factorization with uncertainty and missing data using the em algorithm. In *CVPR*, volume 1, pp. I–I. IEEE, 2004.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016a.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016b.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Ho, J., Yang, M.-H., Lim, J., Lee, K.-C., and Kriegman, D. Clustering appearances of objects under varying illumination conditions. In *CVPR*, volume 1, pp. 11–18. IEEE, 2003.

Ji, P., Li, H., Salzmann, M., and Dai, Y. Robust motion segmentation with unknown correspondences. In *ECCV*, pp. 204–219. Springer, 2014a.

Ji, P., Salzmann, M., and Li, H. Efficient dense subspace clustering. In *WACV*, pp. 461–468. IEEE, 2014b.

Ji, P., Salzmann, M., and Li, H. Shape interaction matrix revisited and robustified: Efficient subspace clustering with corrupted and incomplete data. In *ICCV*, pp. 4687–4695, 2015.

Ji, P., Li, H., Salzmann, M., and Zhong, Y. Robust multibody feature tracker: a segmentation-free approach. In *CVPR*, pp. 3843–3851, 2016.

Ji, P., Reid, I. D., Garg, R., Li, H., and Salzmann, M. Adaptive low-rank kernel subspace clustering. In *arXiv preprint arXiv:1707.04974v4*, 2017a.

Ji, P., Zhang, T., Li, H., Salzmann, M., and Reid, I. Deep subspace clustering networks. In *NeurIPS*, pp. 23–32, 2017b.

Kanatani, K.-i. Motion segmentation by subspace separation and model selection. In *ICCV*, volume 2, pp. 586–591. IEEE, 2001.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *ICML*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Liu, G. and Yan, S. Latent low-rank representation for subspace segmentation and feature extraction. In *ICCV*, pp. 1615–1622. IEEE, 2011.

Liu, G., Lin, Z., Yan, S., Sun, J., Yu, Y., and Ma, Y. Robust recovery of subspace structures by low-rank representation. *TPAMI*, 35(1):171–184, 2013.

Lloyd, S. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

Lu, C.-Y., Min, H., Zhao, Z.-Q., Zhu, L., Huang, D.-S., and Yan, S. Robust and efficient subspace segmentation via least squares regression. In *ECCV*, pp. 347–360. Springer, 2012.

Ma, Y., Derksen, H., Hong, W., and Wright, J. Segmentation of multivariate mixed data via lossy data coding and compression. *TPAMI*, 29(9), 2007.

Mo, Q. and Draper, B. A. Semi-nonnegative matrix factorization for motion segmentation with missing data. In *ECCV*, pp. 402–415. Springer, 2012.

Mukherjee, S., Asnani, H., Lin, E., and Kannan, S. Clustergan : Latent space clustering in generative adversarial networks. *CoRR*, abs/1809.03627, 2018.

Nguyen, H. V. and Bai, L. Cosine similarity metric learning for face verification. In *ACCV*, pp. 709–720. Springer, 2010.

Ochs, P., Malik, J., and Brox, T. Segmentation of moving objects by long term video analysis. *TPAMI*, 36(6):1187–1200, 2014.

Oh Song, H., Xiang, Y., Jegelka, S., and Savarese, S. Deep metric learning via lifted structured feature embedding. In *CVPR*, pp. 4004–4012, 2016.

Patel, V. M. and Vidal, R. Kernel sparse subspace clustering. In *ICIP*, pp. 2849–2853. IEEE, 2014.

Patel, V. M., Van Nguyen, H., and Vidal, R. Latent space sparse subspace clustering. In *ICCV*, pp. 225–232, 2013.

Peng, X., Xiao, S., Feng, J., Yau, W.-Y., and Yi, Z. Deep subspace clustering with sparsity prior. In *IJCAI*, 2016.

Purkait, P., Chin, T.-J., Ackermann, H., and Suter, D. Clustering with hypergraphs: the case for large hyperedges. In *ECCV*, pp. 672–687. Springer, 2014.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

Tseng, P. Nearest q-flat to m points. *Journal of Optimization Theory and Applications*, 105(1):249–252, 2000.

Vidal, R. Subspace clustering. *IEEE Signal Processing Magazine*, 28(2):52–68, 2011.

Vidal, R. and Favaro, P. Low rank subspace clustering (LRSC). 43:47–61, 2014.

Wang, Y.-X., Xu, H., and Leng, C. Provable subspace clustering: When LRR meets SSC. In *NeurIPS*, pp. 64–72, 2013.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

Xiao, S., Tan, M., Xu, D., and Dong, Z. Y. Robust kernel low-rank representation. *IEEE transactions on neural networks and learning systems*, 27(11):2268–2281, 2016.

Xie, J., Girshick, R., and Farhadi, A. Unsupervised deep embedding for clustering analysis. In *ICML*, 2016.

Yan, J. and Pollefeys, M. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In *ECCV*, pp. 94–106. Springer, 2006.

Yang, A. Y., Rao, S. R., and Ma, Y. Robust statistical estimation and segmentation of multiple subspaces. In *null*, pp. 99. IEEE, 2006.

Yang, A. Y., Wright, J., Ma, Y., and Sastry, S. S. Unsupervised segmentation of natural images via lossy data compression. *CVIU*, 110(2):212–225, 2008.

Yang, B., Fu, X., Sidiropoulos, N. D., and Hong, M. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *ICML*, volume 70, pp. 3861–3870, 2017.

Yin, M., Guo, Y., Gao, J., He, Z., and Xie, S. Kernel sparse subspace clustering on symmetric positive definite manifolds. In *CVPR*, pp. 5157–5164, 2016.

You, C., Li, C.-G., Robinson, D. P., and Vidal, R. Oracle based active set algorithm for scalable elastic net subspace clustering. In *CVPR*, pp. 3928–3937, 2016a.

You, C., Robinson, D., and Vidal, R. Scalable sparse subspace clustering by orthogonal matching pursuit. In *CVPR*, pp. 3918–3927, 2016b.

Zhang, T., Szlam, A., Wang, Y., and Lerman, G. Hybrid linear modeling via local best-fit flats. *IJCV*, 100(3): 217–240, 2012.

Zhang, T., Ji, P., Harandi, M., Hartley, R. I., and Reid, I. D. Scalable deep k-subspace clustering. *CoRR*, abs/1811.01045, 2018.