Overcomplete Deep Subspace Clustering Networks

Jeya Maria Jose Valanarasu Vishal M. Patel
Department of Electrical and Computer Engineering,
Johns Hopkins University, 3400 N. Charles St, Baltimore, MD 21218, USA
{jvalana1, vpatel36}@jhu.edu

Abstract

Deep Subspace Clustering Networks (DSC) provide an efficient solution to the problem of unsupervised subspace clustering by using an undercomplete deep auto-encoder with a fully-connected layer to exploit the self expressiveness property. This method uses undercomplete representations of the input data which makes it not so robust and more dependent on pre-training. To overcome this, we propose a simple yet efficient alternative method - Overcomplete Deep Subspace Clustering Networks (ODSC) where we use overcomplete representations for subspace clustering. In our proposed method, we fuse the features from both undercomplete and overcomplete auto-encoder networks before passing them through the self-expressive layer thus enabling us to extract a more meaningful and robust representation of the input data for clustering. Experimental results on four benchmark datasets show the effectiveness of the proposed method over DSC and other clustering methods in terms of clustering error. Our method is also not as dependent as DSC is on where pre-training should be stopped to get the best performance and is also more robust to noise. Code - https://github.com/jeya-maria-jose/Overcomplete-Deep-Subspace-Clustering

1. Introduction

Subspace Clustering [44] is a learning paradigm which involves grouping a set of similar data points in an unsupervised way. Let $X \in \mathbb{R}^{D\times N}$ be a matrix such that its columns are chosen from a union of k subspaces of $\mathbb{R}^D$, $\cup_{i=1}^{k}\{S_i\}$ of dimensions d_i where $d_i << min\{D, N\}$, then the task of subspace clustering is to categorize the columns of X into their corresponding subspaces. Subspace clustering is widely popular for its use in image clustering [9, 50], image segmentation [49, 31], motion segmentation [10, 18], face clustering [14] and various other computer vision tasks. As most of the data in real-world are high dimensional, there exists a requirement to convert the high dimensional data into meaningful subspace representations before clustering. A union of multiple subspaces can then be clustered together into a single category. The use of multiple subspaces is what differentiates subspace clustering from principal component analysis (PCA) related methods where it is assumed that data is drawn from a single low-dimensional subspace.

Several methods have been proposed in the literature for solving the problem of subspace clustering using conventional methods [10, 53, 28, 54, 11, 29, 30, 37, 3, 6]. The first deep learning-based solution to the problem of subspace clustering, called Deep Subspace Clustering Networks (DSC), was introduced in [17]. DSC achieved a huge boost in performance when compared to previous conventional methods. DSC uses a convolutional autoencoder to learn a non-linear representation of the data. The network has an undercomplete ("encoder-decoder") architecture where the encoder gets trained to learn an abstract low-dimensional representation of the input image while the decoder learns to reconstruct the original image from those representations. This deep representation learned by the encoder explicitly performs a non-linear mapping of the data which helps in performing better subspace clustering. Although DSC performs very well and has a huge margin of improvement in terms of performance over the previous methods, there exists two main issues with the method. This method is not so robust and its performance drops considerably when there are potential degradations (i.e. noise) in the data leading to noisy representations [61]. Another major issue with DSC is that it depends a lot on where pre-training is stopped. Even if the pre-training is stopped some epochs before or after the correct epoch where clear reconstructions start to appear, the clustering becomes unstable and the performance drops significantly.

In this paper, we propose an alternative solution which improves the performance while being able to obtain a robust representation and a stable training. We propose Overcomplete Deep Subspace Clustering Networks (ODSC) where we make use of overcomplete representations which has greater robustness in the presence of noise and has a flexibility to match structures in the data. We induce overcomplete representations here by introducing an overcomplete convolutional autoencoder which is trained in parallel to the undercomplete autoencoder as in DSC. We then combine both the representations and use a self-expressive layer to learn pairwise affinities between the data points. This simple trick of fusing both overcomplete and undercomplete representations make the training stable and not be over-dependent on pre-training. We extensively analyze this issue by conducting various experiments. The pre-trained ODSC autoencoder is also able to obtain far better reconstructions compared to the pre-trained DSC autoencoder which only shows that better representations are learned by the overcomplete network. Even while maintaining the number of parameters to be the same as that of DSC, we get a good improvement over the performance of DSC with added advantages of more robustness and stable training. We evaluate our method on four different benchmark datasets: MNIST [21], COIL20 [33], ORL [40] and Extended Yale B [22]. Our experiments demonstrate that ODSC significantly outperforms DSC and other conventional subspace clustering methods by a large margin.

2. Related Work

Subspace clustering was initially solved by methods which relied on linear methods. These methods first construct an affinity matrix by measuring the affinity for every pair of data points. Then methods like NCut (normalized cuts) [41] or spectral clustering [34] were used on the affinity matrix. These two problems are either solved sequentially [9, 11, 28, 29, 30] or in multiple passes in an alternate manner [12, 13, 24, 25, 56]. An affinity matrix can be built by exploiting the self-expressiveness property of data [9, 12, 24, 28, 30, 15], using factorization methods [8, 16, 18, 32, 46] or by using high-order model-based methods [7, 35, 39]. Out of these, self-expressiveness property-based methods are more robust to corruption by noise. The property of self-expressiveness corresponds to the ability to represent data points as a linear combination of other points in the same subspace. Self-expressiveness can be formulated as follows: Given a set of data points $\{x_i\}_{i=1,\dots,N}$ which are taken from a collection of linear subspaces $\{S_i\}_{i=1,\dots,N}$, define a matrix X whose columns are stacked up with x_i. The self-expressiveness property can then be represented as $X = XC$, where C is the self-representation matrix. It has been shown that if the subspaces are independent, then C is guaranteed to have a block diagonal structure [15]. This means that if points x_i and x_j lie in the same subspace then the corresponding coefficient c_{ij} in matrix C cannot be zero. To build an affinity matrix for spectral clustering, matrix C can be used. This idea can be mathematically represented as an optimization problem as follows:

$$\min_C \|C\|_p + \frac{\lambda}{2}\|X - XC\|_F^2 \quad s.t. \quad (\mathrm{diag}(C) = 0), \quad (1)$$

where $p = 0, 1$ or nuclear norm. The diagonal constraint here avoids trivial solutions for sparsity inducing norms. A major drawback of this clustering approach is that this holds true only for linear subspaces. For solving cases where data points do not lie in the linear subspace, several non-linear kernel-based methods have also been proposed [36, 38, 48, 52, 37] which require a pre-defined kernel. However, these predefined kernels cannot be assured to provide feature spaces that are well suited for subspace clustering.

Following the popularity of deep learning methods in various tasks of computer vision and machine learning like image segmentation, image restoration, medical image analysis, etc., deep learning was explored for subspace clustering in DSC [17]. DSC also introduced a novel self-expressive layer for deep autoencoders so as to train an autoencoder such that its latent representation is well-suited for subspace clustering. This idea of harnessing the self-expressiveness property using a deep autoencoder resulted in a huge boost in performance when compared to other conventional methods [2]. Based on this work, several extensions were proposed very recently exploring adversarial learning [61, 4], multimodal inputs [5], multi-level representations [19], and self-supervised learning [55] for subspace clustering. Other spectral clustering free techniques like distribution preserving DSC [60] and [57] have also been proposed. More recently, works like Deep Latent Low-Rank Fusion Network [59], Block-diagonal Adaptive representation [58], Multilayer Collaborative Low-Rank Coding Network [26] have also been proposed for subspace clustering. In contrast to these methods, we focus completely on DSC and show how it can be easily made very efficient and robust using overcomplete representations.

3. Overcomplete Deep Subspace Clustering Networks (ODSC)

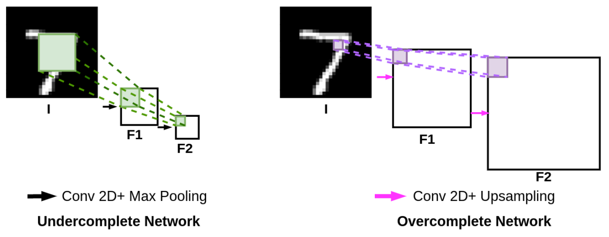
The proposed approach makes use of overcomplete representations to improve the clustering performance. In this section, we first briefly describe the concept of overcomplete representations before explaining our proposed network architecture, clustering method and training strategy.

3.1. Overcomplete Representations

Overcomplete Representations [23] were first introduced as an alternative and a more general method for signal representation. It involved using overcomplete bases (overcomplete dictionaries) so that the number of basis functions is more than the number of input signal samples. This enables a higher flexibility for capturing structure in data and thus it is shown to be more robust. From [47], we can see that overcomplete auto-encoders acted as better feature extractors for denoising. Interestingly, the idea of overcomplete representations have been very much under-explored in deep learning. All the major architectures widely used in deep learning use a generic "encoder-decoder" architecture in which the encoder tries to extract an abstract version of the input data and the decoder learns to take the latent low-dimensional representation back to a high-dimensional output depending on the task at hand. This generic "encoder-decoder" model is an example of undercomplete representations as the number of spatial dimensions is less in the latent space when compared to the input data. This is accomplished in a deep convolutional undercomplete auto-encoder where the convolutional layers are followed by max-pooling layers in the encoder and by upsampling layer in the decoder. Max-pooling reduces the spatial dimensionality of the feature maps while upsampling does the opposite. Note that this arrangement of an undercomplete network's encoder is what forces the initial layers of a deep network to learn low-level features and the deep layers to learn high-level features. This happens because the receptive field of the filters increases after every max-pooling layer. With an increased receptive field, the filters in the deep layers have access to more pixels in the initial image thus enabling them to learn high-level features. Recently, overcomplete representations have been explored for medical image segmentation [43, 42] and image-deraining [51].

In this work, we propose using an overcomplete deep autoencoder, where the encoder takes the input data to a higher spatial dimension. This is achieved by using an upsampling layer after every convolutional layer in the encoder. Note that, the dimensionality of the latent representations for any convolutional network depends on the number of filters and the feature map size used in a network. In [47], an overcomplete fully-connected network is defined as a network which has more number of neurons for representation in its hidden layers than in the initial layers. Similarly in this paper, we define an overcomplete CNN as a network that takes the input to a higher dimension in its deeper layers (spatially).

Replacing max-pooling layers with upsampling layers in the encoder causes the receptive field size to be constrained in the deeper layers making the deeper layers learn more fine details than the initial layers as seen in Fig 1. To understand this further, let I be the input image, F_1 and F_2 be the feature maps extracted from the conv layers 1 and 2, respectively. The max-pooling layer present in these conv layers of the undercomplete architecture (as seen in Fig 1(a)) is the main reason why the receptive field is large in the successive layers. Let the initial receptive field of the conv filter be $k \times k$ on the image. The receptive field size change due to



(a) (b)

Figure 1. Explanation of how receptive field changes in an (a) undercomplete network architecture, and in an (b) overcomplete network architecture.

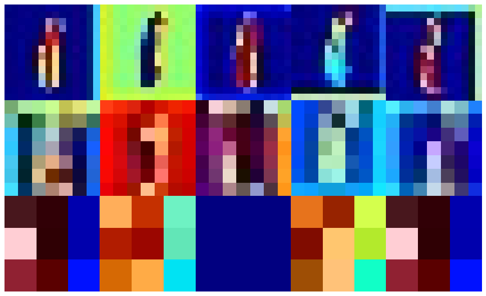
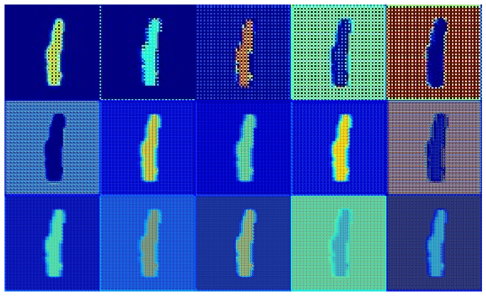
max-pooling layer is dependent on two variables- pooling coefficient and stride of the pooling filter. For convenience, the pooling coefficient and stride is both set as 2 (as in most of the networks). Considering this configuration, the receptive field of conv layer 2 (to which F_1 is forwarded) on the input image would be $2 \times k \times 2 \times k$. Similarly, the receptive field of conv layer 3 (to which F_2 is forwarded) would be $4 \times k \times 4 \times k$. This increase in receptive field can be generalized for the i^{th} layer in an undercomplete network as follows:

$$RF(w.r.t\ I) = 2^{2*(i-1)} \times k \times k.$$

For the proposed overcomplete network, we have upsampling layer of coefficient 2 replacing the max-pooling layer. As the upsampling layer actually works exactly opposite to that of max-pooling layer, the receptive field of conv layer 2 on the input image now would be $\frac{1}{2} \times k \times \frac{1}{2} \times k$. Similarly, the receptive field of conv layer 3 now would be $\frac{1}{4} \times k \times \frac{1}{4} \times k$. This increase in receptive field can be generalized for the i^{th} layer in the overcomplete branch as follows:

$$RF(w.r.t\ I) = \left(\frac{1}{2}\right)^{2*(i-1)} \times k \times k.$$

This helps in the overcomplete network learn more low-level information like edges and other finer details better. In Figure 2, we visualize some of the feature maps learned by the undercomplete and overcomplete networks while trained on the MNIST dataset. We can observe that the learned features in the overcomplete network are more detailed and capture the edges perfectly when compared to the features extracted from the undercomplete network. Also, we can see that the features in the deep layers of an undercomplete network are more coarse when compared to overcomplete network. In fact the feature maps in the deep layers of an overcomplete network are more fine detailed due to the large feature size. These differences show the superiority of overcomplete representations from a convolutional neural network perspective.

(a) (b)

Figure 2. Feature maps captured using (a) Undercomplete network architecture. (b) Overcomplete network architecture. The rows represent the layer from which the feature maps were taken. Row 1 corresponds to layer 1, Row 2 corresponds to layer 2 and Row 3 corresponds to layer 3.

3.2. Network Architecture

Now that we have established how an overcomplete deep network can be made to learn overcomplete latent representations of the data, we discuss how it can be designed to solve subspace clustering.

In ODSC, we propose using two encoders which get trained in parallel. One is a generic encoder which has max-pooling layers after every convolution layer. Another is an overcomplete encoder where we have an upsampling layer after every convolution layer. For upsampling, after exploring both learned convolution and using a bilinear interpolation method, we found that both the methods resulted in equal performance. So, we chose to use bilinear interpolation for upsampling in our architecture for simplicity. In the latent space, we fuse both the latent representations before passing them to the self-expressive layer. The latent overcomplete representations are passed through a max-pooling layer before being fused with the latent representations of the undercomplete encoder. The reason behind this fusion approach is that even though overcomplete representations are better and more meaningful for clustering when compared to the undercomplete representations, they are relatively larger in the spatial sense and so we would need more parameters in our self-expressive layer to accomodate them. This makes the training of the network difficult as when we have more number of parameters we need a lot of data to prevent overfitting. So, using fusion we are able to maintain less number of parameters in the self-expressive layer as in DSC while also being able to take the advantages of overcomplete representations. We pass this latent space representation to the self-expressive layer.

The self-expressive layer is a fully-connected linear layer where the weights of it correspond to the coefficients in the self-expression representation matrix C. This layer learns the affinity matrix directly. For the decoder part, we have a common decoder that consists of convolutional layers followed by upsampling layers. We do not choose to use an overcomplete decoder here because it leads to more parameters and does not contribute much in the performance as the latent representations are what matters more for self-expressive layer. The number of convolutional layers we use in both the encoder and decoder varies with respect to the dataset. We decide on that based on the total number of data available in each dataset. Using the affinity matrix learned, we perform spectral clustering to get the clusters. Figure 3 illustrates the proposed network architecture and the ODSC method.

3.3. Training Details

The autoencoder part of our network is initially trained separately for the task of reconstruction in an unsupervised way. It is trained with a reconstruction loss which is just the mean squared error (MSE) calculated between the reconstruction by the autoencoder $\hat{X}$ and the input image X. The reconstruction loss $\mathcal{L}_r$ is formulated as follows:

$$\mathcal{L}_r = \|X - \hat{X}\|_F^2. \tag{2}$$

We use Adam optimizer [20] with a learning rate of 0.001 to train the reconstruction network for all the experiments. Note that the self-expressive layer is not trained in this part. For subspace clustering, we start by loading these pre-trained weights into the network. Then, we fine-tune the network by using the self-expressive layer and a self-expressive loss term $\mathcal{L}_{self}$, which can be formulated as follows:

$$\mathcal{L}_{self}(\theta, C) = \lambda_2 \|C\|_p + \frac{\lambda_3}{2}\|Z_{\theta_e} - Z_{\theta_e}C\|_F^2, \tag{3}$$

where θ represents the parameters of the network and θ_e specifically represents the parameters of the encoder. Z represents the latent representations found in the self-expressive layer of the network and C represents the self-representation coefficient matrix. We use L2 regularization on C ($p = 2$). For training the network in the fine-tuning

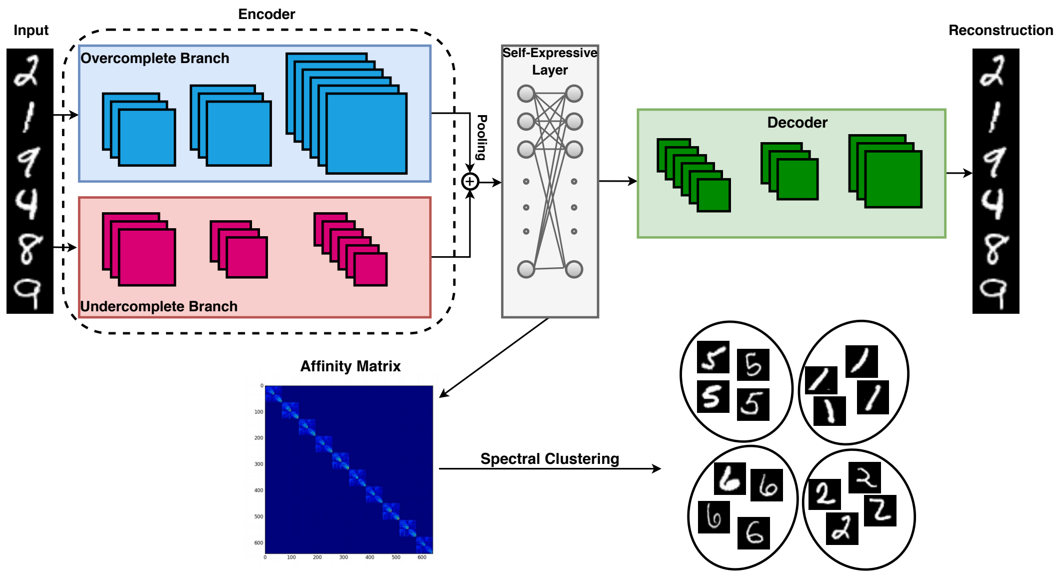


Figure 3. Overall approach for the proposed ODSC method.

stage, we optimize the loss using Adam optimizer on the combination of both of these losses. The final loss $\mathcal{L}_{Total}$ is defined as follows:

$$\mathcal{L}_{Total} = \frac{\lambda_1}{2}\mathcal{L}_r + \mathcal{L}_{self}$$

$$\mathcal{L}_{Total} = \frac{\lambda_1}{2}\|X - \hat{X}\|_F^2 + \lambda_2\|C\|_p + \frac{\lambda_3}{2}\|Z_{\theta_e} - Z_{\theta_e}C\|_F^2, \quad (4)$$

where λ_1, λ_2 and λ_3 are the hyperparameters that control the amount of effect each separate loss term can have over the total loss. These hyperparameter settings are discussed in the next section separately for each dataset. The whole training process is unsupervised as we do not make use of any labels. We make use of only the input data and latent representations derived from the input for training our network. After fine-tuning, we use the parameters of the self-expressive layer to construct the affinity matrix for spectral clustering [34]. We follow the same heuristics used by SSC [10] for this step.

4. Experiments

We perform all our experiments in Python using Tensorflow-1.14 [1] and evaluate our method using four datasets - MNIST [21], COIL20 [33], ORL [40] and Extended Yale B [22]. Our method is compared with the following baselines: Low Rank Representation (LRR) [27], Low Rank Subspace Clustering (LRSC) [45], Sparse Subspace Clustering (SSC) [10], Kernel Sparse Subspace Clustering (KSSC) [38], SSC by Orthogonal Matching Pursuit (SSCOMP) [54], Efficient Dense Subspace Clustering (EDSC) [15], SSC with the pre-trained convolutional auto-encoder features (AE+SSC), EDSC with the pre-trained convolutional auto-encoder features (AE+EDSC) and Deep Subspace Clustering (DSC) Networks [17]. In the following sections, we will discuss in detail the hyperparameters we use for each dataset and the improvements we obtain over the existing methods. The hyperparameters vary for each dataset because the number of data in each dataset varies. Note that for fair comparison, in all the experiments we either matched or used less number of parameters in the self-expressive layer when compared to DSC by lowering the number of filters used in a layer even though we use two encoders.

4.1. MNIST Dataset

The MNIST dataset has a collection of hand-written digit images from 0 to 9. We randomly pick 100 images out of each class and use this collection of 1000 images for the task of subspace clustering. The size of the images is 28×28. It can be noted that the MNIST dataset accounts for many deformations caused by the style of hand-writing even among a single class making the task of clustering difficult as it is an unsupervised setting. The network architecture (ODSC) we use for this dataset has 2 convolutional blocks in the overcomplete branch of encoder, 3 in the undercomplete branch of encoder and 3 in the decoder. The details of what is present in each convolutional block were explained in the previous section. The kernel size of the convolutional layer is 5×5 in the first layer and 3×3 in every other layer in the encoder and vice- versa in the decoder. The number of filters is 20, 10 in the overcomplete branch of encoder and 20, 10, 5 in the undercomplete branch in order, respectively.

In decoder, the number of filters are 5, 10, 20 in each convolutional block, respectively. We set $\lambda_1 = 1.00, \lambda_2 = 20.00$ and $\lambda_3 = 0.1$. The network is fine-tuned for 100 epochs. The results in terms of clustering error are tabulated in Table 8 where it can be seen that our ODSC method achieves an improvement of 6.2 % when compared to DSC.

4.2. ORL Dataset

The ORL dataset has a collection of face images corresponding to 40 subjects with 10 samples for each person. The whole collection of 400 images is used in the experiment for subspace clustering. The images are resized to 32×32. It can be noted that the ORL dataset consists of images under different lighting conditions and with different facial expressions. Also, this dataset is relatively smaller with just 400 images. The network architecture (ODSC) we use for this dataset has 2 convolutional blocks in the overcomplete branch of encoder, 3 in the undercomplete branch of encoder and 3 in the decoder. The kernel size of the convolutional layer is 3×3 for all the convolution layers in both encoder and decoder. The number of filters is 3, 3, and 6 for each block in encoder and in 6, 3 and 3 for each block in the decoder in the same order respectively. We set $\lambda_1 = 1.00, \lambda_2 = 2.00$ and $\lambda_3 = 0.1$. The network is fine-tuned for 800 epochs. The results in terms of clustering error are tabulated in Table 2 where it can be seen that our proposed method ODSC achieves an improvement of 2.00 % compared to DSC.

4.3. COIL20 Dataset

The COIL20 dataset has a collection of different object images. It consists of 20 classes and 1440 images. The images are resized to 32×32. The main challenge of this dataset is that different samples of the same object are captured in different angles which makes even the images of the same object look very different. The network architecture (ODSC) proposed for this dataset has 1 convolutional block in both the undercomplete and overcomplete branches of encoder, 1 convolutional block in the decoder. The kernel size of the convolutional layer is 3×3 and the number of filters is 15 for all the convolution layers in both encoder and decoder. We set $\lambda_1 = 1.00, \lambda_2 = 1.00$ and $\lambda_3 = 15.00$. The network is fine-tuned for 40 epochs. The results in terms of clustering error for COIL20 are tabulated in Table 3. Our proposed method ODSC achieves an improvement of 2.64 % over DSC for COIL20 dataset.

4.4. Extended YaleB Dataset

The Extended YaleB Dataset has a collection of face images which are taken under varying light conditions. It consists of 38 classes with 64 images per class resulting in a total of 2432 images. The images are resized to 48×42 for easy comparison with baselines. The network architecture (ODSC) we use for this dataset has 2 convolutional blocks in the overcomplete branch of encoder, 3 in the undercomplete branch of encoder and 3 in the decoder. The kernel size of the convolutional layer is 5×5 in the first layer and 3×3 in every other layer in the encoder and vice-versa in the decoder. The number of filters is 2 across all the convolutional layers in the dataset. We set $\lambda_1 = 1.00, \lambda_2 = 1.00$ and $\lambda_3 = 6.30$. The network is fine-tuned for 800 epochs. The results in terms of clustering error are tabulated in Table 4. We note here that even though ODSC achieves a better performance than other methods, this dataset has already reached saturation in terms of performance so our margin of improvement here is not as signifiacnt as that of the other datasets.

5. Discussion

In this section, we first report a detailed ablation study and then discuss other characteristics of the proposed method.

Ablation Study

For ablation study, we start with DSC which uses an undercomplete deep auto-encoder. We represent this setup as DSC-U. Then, we change the deep-autoencoder architecture to overcomplete and represent this setup as DSC-O. The architectures of DSC-U and DSC-O are visualized in Figure 4. We then show how our proposed method ODSC which has a fused encoder architecture of both undercomplete and overcomplete encoder fares compared to the other two. The results for all the four datsets can be found in Table 5. As can be seen from this table, in general DSC-O performs better than DSC-U on all datasets. The best performance is obtained when both DSC-U and DSC-O are fused (i.e. ODSC). This experiment clearly shows the significance of the proposed subspace clustering method.

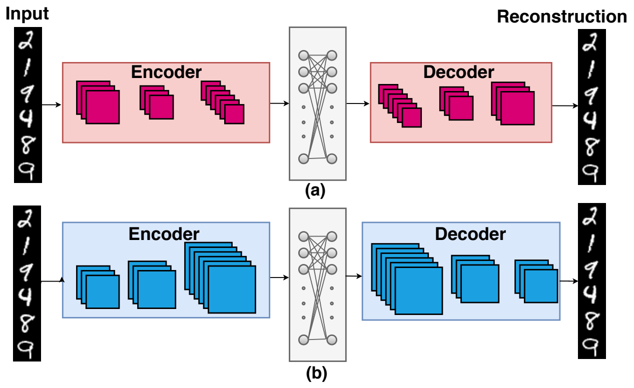


Figure 4. Architecture for ablation study: (a) DSC - Undercomplete (DSC-U). (b) DSC - Overcomplete (DSC-O).

Method	SSC	ENSC	KSSC	SSC-OMP	EDSC	LRR	LRSC	AE+SSC	DSC-L1	DSC-L2	ODSC
Error	54.70	50.17	47.80	66.00	43.50	46.14	48.60	51.60	27.20	25.00	**18.80**

Table 1. Comparison of clustering error with recent methods for the MNIST dataset.

Method	SSC	ENSC	KSSC	SSC-OMP	EDSC	LRR	LRSC	AE+SSC	DSC-L1	DSC-L2	ODSC
Error	32.50	24.75	34.25	36.00	27.25	38.25	32.5	26.75	14.25	14.00	**12.00**

Table 2. Comparison of clustering error with recent methods for the ORL dataset.

Method	SSC	ENSC	KSSC	SSC-OMP	EDSC	LRR	LRSC	AE+SSC	DSC-L1	DSC-L2	ODSC
Error	14.86	12.40	24.65	45.90	14.86	31.01	31.25	22.08	6.95	5.14	**2.5**

Table 3. Comparison of clustering error with recent methods for the COIL20 dataset.

Method	SSC	ENSC	KSSC	SSC-OMP	EDSC	LRR	LRSC	AE+SSC	DSC-L1	DSC-L2	ODSC
Error	27.51	12.40	27.75	24.71	11.64	34.81	29.89	25.33	3.33	2.67	**2.22**

Table 4. Comparison of clustering error with recent methods for the Extended Yale B dataset.

Method	DSC (U)	DSC (O)	ODSC
MNIST	25.00	21.60	**18.8**
ORL	14.00	12.75	**12.00**
COIL20	5.14	3.20	**2.50**
EYaleB	2.67	2.35	**2.22**

Table 5. Ablation study. The numbers correspond to the error in terms of percentage.

Reconstruction

In both DSC and ODSC, the networks are first trained for reconstruction. Only from this process are the latent representations learned and are later fed into the self-expressive layer during fine-tuning. It is evident that better the latent representations, better the reconstructions. This is because only if the latent representations are meaningful, the decoder will be able to make a good reconstruction. In case of overcomplete architecture, as we project the images to a higher dimension, the feature maps learn very fine details. When visualized, these feature maps have a very good representation of the input image (please refer Fig 2). Thus, for the overcomplete architecture we were able to get far better reconstruction when compared to undercomplete architecture. The reconstructions of both the types of architectures for the MNIST and COIL20 datasets are visualized in Figure 5. From this figure, we can see that the overcomplete network is able to achieve better reconstructions and as a result the corresponding latent representations are more meaningful than undercomplete representations.

Robustness

The performance of DSC significantly depends on the right place where the pretraining is stopped. The authors of DSC follow an approach of stopping the pretraining process when the reconstructions start to look reasonable. Stopping the training based on the reconstruction quality is not efficient. Moreover, stopping pretraining a few epochs before or after the desired instant leads to poor results in their case. A network with high dependence on this would be unstable and inefficient during real-world implementation. To this end, we show that overcomplete representations are more robust and so do not depend much on things like where pretraining is stopped. Table 9 shows the results for an experiment we carried out where we stopped the pretraining at different epochs for both DSC and ODSC for the MNIST dataset. It can be observed that the change in the performance for ODSC is minimal when compared to that of DSC.

In real-world, data is often noisy. Hence we compare the performance of different mehtods on noisy data. We add random noise to the MNIST dataset and carry out clustering with both DSC and ODSC. The random noise is added in different levels so as to study the increase in error for the methods with respect to increase in noise. The results corresponding to this experiment are tabulated in Table 8. It can be seen that ODSC gives a better performance compared to DSC for all levels of noise. One major observation is that when the noise level is taken from 0% to 50%, the error for DSC goes from 25.00 to 28.10, causing an increase of 3.10% error when the noise level is increased by 50%. However for the same increase in noise level, the increase in error for ODSC is just 1.10%. This shows us that ODSC is more robust to addition of noise when compared to DSC.

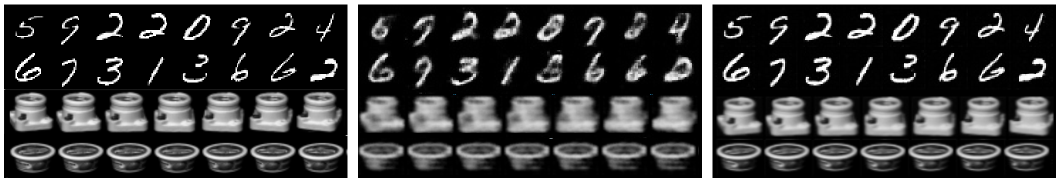

(a) (b) (c)

Figure 5. (a) Input data. (b) Reconstructions using an undercomplete deep auto-encoder. (c) Reconstructions using an overcomplete deep auto-encoder.

layer	enc-1	enc-2	enc-3	self-expressive	dec-1	dec-2	dec-3	Total
kernel size	5×5	3×3	3×3	-	3×3	3×3	5×5	-
channels	5	3	3	-	3	3	5	-
# Parameters	130	138	84	160000	84	140	126	160702

Table 6. Details of the DSC network.

layer	enc-1 (U)	enc-2 (U)	enc-3 (U)	enc-1 (O)	enc-2 (O)	self-expressive	dec-1	dec-2	dec-3	Total
kernel size	3×3	3×3	3×3	3×3	3×3	-	3×3	3×3	3×3	-
channels	3	3	3	3	3	-	3	3	3	-
# Parameters	30	84	84	30	84	160000	84	84	30	160510

Table 7. Details of the ODSC network.

Method	0%	10%	20%	30%	40%	50%
DSC	25.00	26.60	27.00	27.20	27.40	28.10
ODSC	**18.80**	**19.60**	**19.70**	**20.00**	**19.90**	**19.90**

Table 8. Comparison of DSC and ODSC in terms of clustering error percentage for MNIST data when added with different levels of noise. The first row corresponds to the amount of noise added to the data.

Pretraining Epoch	50	100	150	Avg
DSC	27.90	26.10	32.5	28.33
ODSC	21.50	18.80	21.80	20.7

Table 9. Comparison of DSC and ODSC in terms of clustering error percentage while stopping pretraining at different epochs for MNIST dataset.

Number of Parameters

Since ODSC has feature maps of a larger size when compared to DSC and also has two branches in the encoder, it will contain more number of parameters if we have the same number of filters as that of DSC. So, we reduce the number of filters across each layer in ODSC to match the number of parameters of DSC. We analyze this in detail for the architecture we used for ORL dataset as seen in Tables 6 and 7. In Table 7, enc (U) corresponds to the undercomplete branch of encoder while enc (O) corresponds to the overcomplete branch of encoder. It can be seen that the number of parameters of ODSC is actually less than that of DSC. From this, we show that ODSC giving better results is not due to more number of parameters but due to the better properties of overcomplete representations. Also, as we make sure that the number of parameters in the self-expressive layer is the same for DSC and ODSC, so that the change in parameters is not that different for any experiment. For real-time usage, we note that we can have more number of parameters for ODSC and might be able to get even better performance.

6. Conclusion

In this paper, we proposed a new solution to the problem of subspace clustering using overcomplete representations. Using a combination of overcomplete and undercomplete networks, we build an affinity matrix and perform spectral clustering to get the clusters. We performed extensive experiments on four benchmark datasets where our proposed method ODSC achieved better results. Also, we show that the latent representations learned by our method is more meaningful as we get better reconstructions and is also more robust to noise. ODSC is also not that dependent on pre-training as DSC.

Acknowledgment

This work was supported by the NSF grant 1910141.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy

Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] M. Abavisani, A. Naghizadeh, D. Metaxas, and V. M. Patel. Deep subspace clustering with data augmentation. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[3] M. Abavisani and V. M. Patel. Domain adaptive subspace clustering. In *British Machine Vision Conference*, 2016.

[4] M. Abavisani and V. M. Patel. Adversarial domain adaptive subspace clustering. In *IEEE 4th International Conference on Identity, Security, and Behavior Analysis (ISBA)*, pages 1–8, 2018.

[5] Mahdi Abavisani and Vishal M Patel. Deep multimodal subspace clustering networks. *IEEE Journal of Selected Topics in Signal Processing*, 12(6):1601–1614, 2018.

[6] M. Abavisani and V. M. Patel. Multimodal sparse and low-rank subspace clustering. *Information Fusion*, 39:168–177, 2018.

[7] Guangliang Chen and Gilad Lerman. Spectral curvature clustering (scc). *International Journal of Computer Vision*, 81(3):317–330, 2009.

[8] João Paulo Costeira and Takeo Kanade. A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, 29(3):159–179, 1998.

[9] Ehsan Elhamifar and René Vidal. Sparse subspace clustering. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2797. IEEE, 2009.

[10] Ehsan Elhamifar and Rene Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2765–2781, 2013.

[11] Paolo Favaro, René Vidal, and Avinash Ravichandran. A closed form solution to robust subspace estimation and clustering. In *CVPR 2011*, pages 1801–1807. IEEE, 2011.

[12] Jiashi Feng, Zhouchen Lin, Huan Xu, and Shuicheng Yan. Robust subspace segmentation with block-diagonal prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3818–3825, 2014.

[13] Xiaojie Guo. Robust subspace segmentation by simultaneously learning data representations and their affinity matrix. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[14] Jeffrey Ho, Ming-Husang Yang, Jongwoo Lim, Kuang-Chih Lee, and David Kriegman. Clustering appearances of objects under varying illumination conditions. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 1, pages I–I. IEEE, 2003.

[15] Pan Ji, Mathieu Salzmann, and Hongdong Li. Efficient dense subspace clustering. In *IEEE Winter Conference on Applications of Computer Vision*, pages 461–468. IEEE, 2014.

[16] Pan Ji, Mathieu Salzmann, and Hongdong Li. Shape interaction matrix revisited and robustified: Efficient subspace clustering with corrupted and incomplete data. In *Proceedings of the IEEE International Conference on computer Vision*, pages 4687–4695, 2015.

[17] Pan Ji, Tong Zhang, Hongdong Li, Mathieu Salzmann, and Ian Reid. Deep subspace clustering networks. In *Advances in Neural Information Processing Systems*, pages 24–33, 2017.

[18] Ken-ichi Kanatani. Motion segmentation by subspace separation and model selection. In *Proceedings Eighth IEEE International Conference on computer Vision. ICCV 2001*, volume 2, pages 586–591. IEEE, 2001.

[19] Mohsen Kheirandishfard, Fariba Zohrizadeh, and Farhad Kamangar. Multi-level representation learning for deep subspace clustering. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 2039–2048, 2020.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[22] Kuang-Chih Lee, Jeffrey Ho, and David J Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on pattern analysis and machine intelligence*, 27(5):684–698, 2005.

[23] Michael S Lewicki and Terrence J Sejnowski. Learning overcomplete representations. *Neural computation*, 12(2):337–365, 2000.

[24] Chun-Guang Li and Rene Vidal. Structured sparse subspace clustering: A unified optimization framework. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 277–286, 2015.

[25] Chun-Guang Li, Chong You, and René Vidal. Structured sparse subspace clustering: A joint affinity learning and subspace clustering framework. *IEEE Transactions on Image Processing*, 26(6):2988–3001, 2017.

[26] Xianzhen Li, Zhao Zhang, Yang Wang, Guangcan Liu, Shuicheng Yan, and Meng Wang. Multilayer collaborative low-rank coding network for robust deep subspace discovery. *arXiv preprint arXiv:1912.06450*, 2019.

[27] Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, Yong Yu, and Yi Ma. Robust recovery of subspace structures by low-rank representation. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):171–184, 2012.

[28] Guangcan Liu, Zhouchen Lin, and Yong Yu. Robust subspace segmentation by low-rank representation. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 663–670, 2010.

[29] Canyi Lu, Jiashi Feng, Zhouchen Lin, and Shuicheng Yan. Correlation adaptive subspace segmentation by trace lasso. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1345–1352, 2013.

[30] Can-Yi Lu, Hai Min, Zhong-Qiu Zhao, Lin Zhu, De-Shuang Huang, and Shuicheng Yan. Robust and efficient subspace segmentation via least squares regression. In *European conference on computer vision*, pages 347–360. Springer, 2012.

[31] Yi Ma, Harm Derksen, Wei Hong, and John Wright. Segmentation of multivariate mixed data via lossy data coding and compression. *IEEE transactions on pattern analysis and machine intelligence*, 29(9):1546–1562, 2007.

[32] Quanyi Mo and Bruce A Draper. Semi-nonnegative matrix factorization for motion segmentation with missing data. In *European Conference on Computer Vision*, pages 402–415. Springer, 2012.

[33] Sameer A Nene, Shree K Nayar, Hiroshi Murase, et al. Columbia object image library (coil-20). 1996.

[34] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002.

[35] Peter Ochs and Thomas Brox. Higher order motion models and spectral clustering. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 614–621. IEEE, 2012.

[36] Vishal M Patel, Hien Van Nguyen, and René Vidal. Latent space sparse subspace clustering. In *Proceedings of the IEEE international conference on computer vision*, pages 225–232, 2013.

[37] V. M. Patel, H. Van Nguyen, and R. Vidal. Latent space sparse and low-rank subspace clustering. *IEEE Journal of Selected Topics in Signal Processing*, 9(4):691–701, 2015.

[38] Vishal M Patel and René Vidal. Kernel sparse subspace clustering. In *2014 ieee international conference on image processing (icip)*, pages 2849–2853. IEEE, 2014.

[39] Pulak Purkait, Tat-Jun Chin, Alireza Sadri, and David Suter. Clustering with hypergraphs: the case for large hyperedges. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1697–1711, 2016.

[40] Ferdinando S Samaria and Andy C Harter. Parameterisation of a stochastic model for human face identification. In *Proceedings of 1994 IEEE workshop on applications of computer vision*, pages 138–142. IEEE, 1994.

[41] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[42] Jeya Maria Jose Valanarasu, Vishwanath A. Sindagi, Ilker Hacihaliloglu, and Vishal M. Patel. Kiu-net: Overcomplete convolutional architectures for biomedical image and volumetric segmentation. arXiv:2010.01663, 2020.

[43] Jeya Maria Jose Valanarasu, Vishwanath A Sindagi, Ilker Hacihaliloglu, and Vishal M Patel. Kiu-net: Towards accurate segmentation of biomedical images using over-complete representations. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part IV 23*, pages 363–373. Springer, 2020.

[44] René Vidal. Subspace clustering. *IEEE Signal Processing Magazine*, 28(2):52–68, 2011.

[45] René Vidal and Paolo Favaro. Low rank subspace clustering (lrsc). *Pattern Recognition Letters*, 43:47–61, 2014.

[46] René Vidal, Roberto Tron, and Richard Hartley. Multiframe motion segmentation with missing data using powerfactorization and gpca. *International Journal of Computer Vision*, 79(1):85–105, 2008.

[47] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[48] Shijie Xiao, Mingkui Tan, Dong Xu, and Zhao Yang Dong. Robust kernel low-rank representation. *IEEE transactions on neural networks and learning systems*, 27(11):2268–2281, 2015.

[49] Allen Y Yang, John Wright, Yi Ma, and S Shankar Sastry. Unsupervised segmentation of natural images via lossy data compression. *Computer Vision and Image Understanding*, 110(2):212–225, 2008.

[50] Ming-Hsuan Yang and Jeffrey Ho. Clustering appearances of objects under varying illumination conditions, Sept. 5 2006. US Patent 7,103,225.

[51] Rajeev Yasarla, Jeya Maria Jose Valanarasu, and Vishal M. Patel. Exploring overcomplete representations for single image deraining using cnns. arXiv:2010.10661, 2020.

[52] Ming Yin, Yi Guo, Junbin Gao, Zhaoshui He, and Shengli Xie. Kernel sparse subspace clustering on symmetric positive definite manifolds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5157–5164, 2016.

[53] Chong You, Chun-Guang Li, Daniel P Robinson, and René Vidal. Oracle based active set algorithm for scalable elastic net subspace clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3928–3937, 2016.

[54] Chong You, Daniel Robinson, and René Vidal. Scalable sparse subspace clustering by orthogonal matching pursuit. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3918–3927, 2016.

[55] Junjian Zhang, Chun-Guang Li, Chong You, Xianbiao Qi, Honggang Zhang, Jun Guo, and Zhouchen Lin. Self-supervised convolutional subspace clustering network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5473–5482, 2019.

[56] Junjian Zhang, Chun-Guang Li, Honggang Zhang, and Jun Guo. Low-rank and structured sparse subspace clustering. In *2016 Visual Communications and Image Processing (VCIP)*, pages 1–4. IEEE, 2016.

[57] Tong Zhang, Pan Ji, Mehrtash Harandi, Wenbing Huang, and Hongdong Li. Neural collaborative subspace clustering. *arXiv preprint arXiv:1904.10596*, 2019.

[58] Zhao Zhang, Jiahuan Ren, Sheng Li, Richang Hong, Zhengjun Zha, and Meng Wang. Robust subspace discovery by block-diagonal adaptive locality-constrained representation. In *Proceedings of the 27th ACM international conference on multimedia*, pages 1569–1577, 2019.

[59] Zhao Zhang, Jiahuan Ren, Zheng Zhang, and Guangcan Liu. Deep latent low-rank fusion network for progressive subspace discovery. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, 2020.

[60] Lei Zhou, Bai Xiao, Xianglong Liu, Jun Zhou, Edwin R Hancock, et al. Latent distribution preserving deep subspace clustering. In *28th International Joint Conference on Artificial Intelligence*. York, 2019.

[61] Pan Zhou, Yunqing Hou, and Jiashi Feng. Deep adversarial subspace clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1596–1604, 2018.