Spectral clustering via ensemble deep autoencoder learning (SC-EDAE)

Séverine Affeldt, Lazhar Labiod, Mohamed Nadif

Spectral Clustering via Ensemble Deep Autoencoder Learning (SC-EDAE)

Séverine Affeldt[a,*], Lazhar Labiod[a], Mohamed Nadif[a]

[a] *Université de Paris, LIPADE,*
45 rue des Saints Pères,
F-75006 Paris, France

Abstract

Several works have studied clustering strategies that combine classical clustering algorithms and deep learning methods. These strategies generally improve clustering performance, however deep autoencoder setting issues impede the robustness of these approaches. To alleviate the impact of hyperparameters setting, we propose a model which combines spectral clustering and deep autoencoder strengths in an ensemble framework. Our proposal does not require any pretraining and includes the three following steps: generating various deep embeddings from the original data, constructing a sparse and low-dimensional ensemble affinity matrix based on anchors strategy and applying spectral clustering to obtain the common space shared by multiple deep representations. While the anchors strategy ensures an efficient merging of the encodings, the fusion of various deep representations enables to mitigate the deep networks setting issues. Experiments on various benchmark datasets demonstrate the potential and robustness of our approach compared to state-of-the-art deep clustering methods.

Keywords: spectral clustering, unsupervised ensemble learning, autoencoder
2010 MSC:

*Corresponding author
Email addresses: severine.affeldt@u-paris.fr (Séverine Affeldt), lazhar.labiod@u-paris.fr (Lazhar Labiod), mohamed.nadif@u-paris.fr (Mohamed Nadif)

1. Introduction

Learning from large amount of data is a very challenging task. Several dimensionality reduction and clustering techniques that are well studied in the literature aim to learn a suitable and simplified data representation from original dataset; see for instance [1, 2, 3]. While many approaches have been proposed to address the dimensionality reduction and clustering tasks, deep learning-based methods recently demonstrate promising results. Motivated by the keen interest in deep learning, many authors tackle the objective of data representation and partitioning using jointly the autoencoders [4] and clustering approaches.

1.1. Deep Autoencoder: challenges and issues

Deep learning is a field of machine learning that is based on multi-level learning of data representations and where one passes from low level features to higher level features through the different layers. These deep architectures can automatically learn important features from images, sound or text data and have made significant progress in the field of computer vision. The autoencoder (AE) algorithm and its deep version (DAE), like the traditional methods of dimensionality reduction, has been a great success in recent years.

An autoencoder [4, 5] is a neural network which is trained to replicate its input at its output. Training an autoencoder is unsupervised in the sense that no labeled data is needed. The training process is still based on the optimization of a cost function. Autoencoders can be used as tools to train deep neural networks [6]. For the purpose of dimensionality reduction, an autoencoder can learn a representation (or *encoding*) for a set of data. If linear activations are used, or only a single sigmoid hidden layer, then the optimal solution to an autoencoder is strongly related to Principal Component Analysis (PCA). With appropriate dimensionality and sparsity constraints, autoencoders can learn data projections that are more interesting than other basic techniques such as PCA which only allows linear transformation of data vectors. By contrast, the autoencoders are non-linear by nature, and can learn more complex relations between visible and

hidden units. Moreover, they can be stacked, which makes them even more powerful.

Recently, a number of works have studied clustering strategies that combine classical clustering algorithms and deep learning methods. These approaches follow either a sequential way, where a deep representation is learned using a deep autoencoder before obtaining clusters using a clustering technique (e.g. *k-means*) [7, 8, 9, 10], or a simultaneous way, where deep representation and clusters are learned jointly by optimizing a single objective function [11, 12, 13]. Both strategies improve clustering performance. However, when dealing with real-world data, existing clustering algorithms based on deep autoencoders suffer from different issues which impede their robustness and ease-to-use, such as,

- **the weights initialisation**, as mentioned in [14], the training of a Deep Neural Network (DNN) still suffers from two major drawbacks, among which the weights initialisation. Indeed, initializing the weights with random values clearly adds randomness to the obtained results. The DNN pretraining [15], which is strongly related to the initialisation issue, has been used in an increasing number of studies [11, 16, 10]. While pretraining helps to improve clustering performance, it is usually computationally intensive and thus raises supplementary training issues.
- **the architecture (or** *structure***)**, the architecture (*i.e.*, number of layers and their width) forces the network to seek a different representation of the data while preserving the important information. However, we observe that in almost all recent papers on deep clustering [11, 12, 13, 9, 10, 17], a different structure is recommended by the authors for each studied dataset. In some studies, the DAE architecture can even lack of technical rationales. Most importantly, the clustering performance of the proposed methods usually strongly depends on a particular DAE structure.

1.2. Spectral clustering: limitations and strengths

Spectral clustering is a popular clustering method that uses eigenvectors of a symmetric matrix derived from the distance between datapoints. It has the

advantage of being applicable to a wide variety of data types and similarity functions, and requires weak assumptions for the cluster shapes [18, 19]. By contrast, the standard `k-means` algorithm, which can be shown to be a version of the classification EM algorithm [20], considers the uniform spherical Gaussian mixture model with equal proportions. Thereby, when one departs from these assumptions and when clusters are not easily separable, `k-means` has difficulty to detect the underlying clusters. Despite its advantages, the spectral clustering is limited in practice due to its computational complexity of $\mathcal{O}(n^3)$. Recently, the *anchor* strategy has been proposed [21, 22] to enable an efficient and scalable spectral clustering approach.

1.3. Our paper's contribution and structure

To address the above mentioned deep learning challenging issues, we propose a Spectral Clustering via Ensemble Deep Autoencoder's algorithm (`SC-EDAE`) which combines the advantages and strengths of spectral clustering, deep embedding models and ensemble paradigm. Ensemble learning has been considered in different machine learning contexts where it generally helps in improving results by combining several models. The ensemble approach allows a better predictive performance and a more robust clustering as compared to the results obtained with a single model [23, 24, 25]. In `SC-EDAE`, the *anchor* strategy [21, 22] mitigates the computational complexity issue of spectral clustering and bridges the gap between large-scale spectral clustering and deep learning in an ensemble approach.

Following the ensemble paradigm, we first used several DAE with different hyperparameters settings to generate m encodings. In a second step, each encoding is projected in a higher features space based on the *anchors* strategy to construct m graph affinity matrices. Finally, we apply spectral clustering on an ensemble graph affinity matrix to have the common space shared by all the m encodings, before we run *k-means* in this common subspace to produce the final clustering (see Fig. 1 for a summary diagram). The *fusion* of m different encodings enables to alleviate the above-mentioned deep neural networks

challenges. Besides, the common space shared by the m encodings is obtained without any heavily time-consuming pre-training. Furthermore, the *anchors* strategy ensures an efficient merging of all the encodings. All in all, `SC-EDAE` results in a effective and robust deep ensemble clustering method.

The outline of the paper is as follows. In Section 2 we present the related work. In Section 3, some notations and preliminaries are given. In Section 4, we present and discuss our approach in full details. In Section 5, the evaluations of the proposed method and comparisons with several related approaches available in the literature are presented. The conclusion of the paper is given in Section 6.

2. Related Work

Despite their success, most existing clustering methods are severely challenged by the data generated with modern applications, which are typically high-dimensional, noisy, heterogeneous and sparse. This has driven many researchers to investigate new clustering models to overcome these difficulties. One promising category of such models relies on data embedding.

Within this framework, classical dimensionality reduction approaches, e.g., Principal Component Analysis (PCA), have been widely considered for the embedding task. However, the linear nature of such techniques makes it challenging to infer faithful representations of real-world data, which typically lie on highly non-linear manifolds. This motivates the investigation of deep learning models (e.g., autoencoders, convolutional neural networks), which have been shown so far to be successful in extracting highly non-linear features from complex data, such as text, images or graphs [4, 26, 5].

The deep autoencoders (DAE) have proven to be useful for dimensionality reduction [4] and image denoising. In particular, the autoencoders (AE) can non-linearly transform data into a latent space. When this latent space has lower dimension than the original one [4], this can be viewed as a form of non-linear PCA. An autoencoder typically consists of an *encoder* stage, that can provide an encoding of the original data in lower dimension, and a *decoder* part,

to define the data reconstruction cost. In clustering context, the general idea is to embed the data into a low dimensional latent space and then perform clustering in this new space. The goal of the embedding here is to learn new representations of the objects of interest (e.g., images) that encode only the most relevant information characterizing the original data, which would for example reduce noise and sparsity.

Several interesting works have recently combined embedding learning and clustering. The proposed methods generally conduct both clustering and deep embedding in two different ways. First, some works proposed to combine deep embedding and clustering in a sequential way. In [7] the authors use a stacked autoencoder to learn a representation of the affinity graph, and then run *k-means* on the learned representations to obtain the clusters. In [10], it has been proposed to train a deep network by iteratively minimizing a Kullback-Leibler (KL) divergence between a centroid based probability distribution and an auxiliary target distribution.

More recently, in [16] the authors propose to incorporate an autoencoder into the Deep Embedded Clustering (DEC) framework [10]. Then, the proposed framework can jointly perform clustering and learn representative features with local structure preservation. A novel non-linear reconstruction method which adopt deep neural networks for representation based community detection has been proposed in [13]. The work presented in [17] combines deep learning with subspace clustering such that the network is designed to directly learn the affinities matrix. Finally, a novel algorithm was introduced in [9] that uses *landmarks* and deep autoencoders, to perform efficient spectral clustering.

Since the embedding process is not guaranteed to infer representations that are suitable for the clustering task, several authors recommend to perform both tasks jointly so as to let clustering govern feature extraction and vice-versa. In [12], the authors propose a general framework, so-called *DeepCluster*, to integrate the traditional clustering methods into deep learning models and adopt Alternating Direction of Multiplier Method to optimize it. In [11], a joint dimensionality reduction and *k-means* clustering approach in which dimensionality

reduction is accomplished via learning a deep neural network is proposed.

Beyond the joint and sequential ways to combine clustering and deep embedding, it appears that the connection between autoencoder and ensemble learning paradigm has not been explored yet. In this paper, we aim to fill the gap between ensemble deep autoencoders and spectral clustering in order to propose a robust approach that takes simultaneously advantage of several deep models with various hyperparameter settings. In particular, we apply spectral clustering on an ensemble of *fused* encodings obtained from m different deep autoencoders. To our knowledge, the adoption of deep learning in an ensemble learning paradigm has not been adequately investigated yet. The goal of this work is to conduct investigations along this direction.

3. Preliminaries

3.1. Notation

Throughout the paper, we use bold uppercase characters to denote matrices, bold lowercase characters to denote vectors. For any matrix $\mathbf{M}$, $\mathbf{m}_j$ denotes the j-th column vector of $\mathbf{M}$, $\mathbf{y}_i$ means the i-th row vector of $\mathbf{Y}$, m_{ij} denotes the $(i,j)-$ element of $\mathbf{M}$ and $Tr[\mathbf{M}]$ is the trace of $\mathbf{M}$ whether $\mathbf{M}$ is a square matrix; $\mathbf{M}^\top$ denotes the transpose matrix of $\mathbf{M}$. We consider the Frobenius norm of a matrix $\mathbf{M} \in \mathbb{R}^{n\times d}$: $||\mathbf{M}||^2 = \sum_{i=1}^{n}\sum_{j=1}^{d} m_{ij}^2 = Tr[\mathbf{M}^\top\mathbf{M}]$. Furthermore, let $\mathbf{I}$ be the identity matrix with appropriate size.

3.2. Spectral clustering

Several spectral clustering algorithms have been proposed in the literature [18, 19], each using the eigenvectors in slightly different ways [27, 28]. The partition of the n datapoints of $\mathbf{X} \in \mathbb{R}^{n\times d}$ into k disjoint clusters is based on an objective function that favors low similarity between clusters and high similarity within clusters. In its normalized version, the spectral clustering algorithm exploits the top k eigenvectors of the normalized graph Laplacian $\mathbf{L}$ that are the relaxations of the indicator vectors which provide assignments of

each datapoint to a cluster. In particular, it amounts to maximize the following relaxed normalized association,

$$\max_{\mathbf{B}\in\mathbb{R}^{n\times k}} Tr(\mathbf{B}^\top \mathbf{S}\mathbf{B}) \quad s.t. \quad \mathbf{B}^\top\mathbf{B} = \mathbf{I} \tag{1}$$

with $\mathbf{S} = \mathbf{D}^{-1/2}\mathbf{K}\mathbf{D}^{-1/2} \in \mathbb{R}^{n\times n}$ is the normalized similarity matrix where $\mathbf{K} \in \mathbb{R}^{n\times n}$ is the similarity matrix and $D \in \mathbb{R}^{n\times n}$ is the diagonal matrix whose (i,i)-element of $\mathbf{X}$ is the sum of $\mathbf{X}$'s i-th row. The solution of (1) is to set the matrix $\mathbf{B} \in \mathbb{R}^{n\times k}$ equal to the k eigenvectors corresponding to the largest k eigenvalues of $\mathbf{S}$. After renormalization of each row of $\mathbf{B}$, a *k-means* assigns each datapoint $\mathbf{x}_i$ of $\mathbf{X}$ to the cluster that the row $\mathbf{b}_i$ of $\mathbf{B}$ is assigned to.

As opposed to several other clustering algorithms (e.g. *k-means*), spectral clustering performs well on arbitrary shaped clusters. However, a limitation of this method is the difficulty to handle large-scale datasets due to the high complexity of the graph Laplacian construction and the eigendecomposition.

Recently, a scalable spectral clustering approach, referred to as *Landmark-based Spectral Clustering* (*LSC*) [21] or *AnchorGraph* [22], has been proposed. This approach allows to efficiently construct the graph Laplacian and compute the eigendecomposition. Specifically, each datapoint is represented by a linear combination of p representative datapoints (or *landmarks*), with $p \ll n$. The obtained representation matrix $\hat{\mathbf{Z}} \in \mathbb{R}^{p\times n}$, for which the affinity is calculated between n datapoints and the p landmarks, is sparse which in turn ensures a more efficient eigendecomposition as compare to the above mentioned eigendecomposition of $\mathbf{S}$ (Eq. 1).

3.3. Deep autoencoders

An autoencoder [29] is a neural network that implements an unsupervised learning algorithm in which the parameters are learned in such a way that the output values tend to copy the input training sample. The internal hidden layer of an autoencoder can be used to represent the input in a lower dimensional space by capturing the most salient features.

Specifically, we can decompose an autoencoder in two parts, namely an *encoder*, f_θ, followed by a *decoder*, g_ψ. The first part allows the computation of a feature vector $\mathbf{y}_i = f_\theta(\mathbf{x}_i)$ for each input training sample, thus providing the *encoding* $\mathbf{Y}$ of the input dataset. The *decoder* part aims at transforming back the encoding into its original representation, $\hat{\mathbf{x}}_i = g_\psi(\mathbf{y}_i)$. The sets of parameters for the encoder f_θ and the decoder g_ψ are learned simultaneously during the reconstruction task while minimizing the *loss*, referred to as $\mathcal{J}$, where $\mathcal{L}$ is a cost function for measuring the divergence between the input training sample and the reconstructed data,

$$\mathcal{J}_{AE}(\theta, \psi) = \sum_{i=1}^{n} \mathcal{L}(\mathbf{x}_i, g_\psi(f_\theta(\mathbf{x}_i))). \tag{2}$$

The encoder and decoder parts can have several shallow layers, yielding a deep autoencoder (DAE) that enables to learn higher order features. The network architecture of these two parts usually mirrors each other.

It is remarkable that PCA can be interpreted as a linear AE with a single layer [4]. In particular, PCA can be seen as a linear autoencoder with $\mathbf{W} \in \mathbb{R}^{d \times k}$ where $k \leq d$. Taking $f_\theta(\mathbf{X}) = \mathbf{XW}$ and $g_\psi\big(f_\theta(\mathbf{X})\big) = \mathbf{XWW}^\top$ we find the objective function $||\mathbf{X} - \mathbf{XWW}^\top||^2$ optimized by PCA.

4. Spectral Clustering via Ensemble DAE

4.1. Problem formulation

Given an $n \times d$ data matrix $\mathbf{X}$, the goal is to first obtain a set of m encodings $\{\mathbf{Y}_\ell\}_{\ell \in [1,m]}$ using m DAE trained with different hyperparameters settings. In a second step, we construct a graph matrix $\mathbf{S}_\ell$ associated to each embedding $\mathbf{Y}_\ell$, and then *fuse* the m graph matrices in an ensemble graph matrix $\mathbf{S}$ (equivalently named *affinity* or *similarity* matrix in the following) which contains information provided by the m embeddings. Finally, to benefit from the common subspace shared by the m deep embeddings, spectral clustering is applied to $\mathbf{S}$. The challenges of the problem are threefold,

1. generate m deep embeddings,

2. integrate the clustering in an ensemble learning framework,

3. solve the clustering task in a highly efficient way.

Each of the above mentioned issues is discussed in the separate subsections 4.2, 4.3 and 4.4 respectively. Most importantly, the `SC-EDAE` approach is provided with an ensemble optimization which is detailed in subsection 4.5.

4.2. Deep embeddings generation

The cost function of an autoencoder, with an encoder f_θ and a decoder g_ψ, measures the error between the input $\mathbf{x} \in \mathbb{R}^{d\times 1}$ and its reconstruction at the output $\hat{\mathbf{x}} \in \mathbb{R}^{d\times 1}$. The encoder f_θ and decoder g_ψ can have multiple layers of different widths. To generate m deep representations or encodings $\{\mathbf{Y}_\ell\}_{\ell\in[1,m]}$, the DAE is trained with different hyperparameter settings (e.g., initialisation, layer widths) by optimizing the following cost function.

$$||\mathbf{X} - g_{\psi_\ell}\big(f_{\theta_\ell}(\mathbf{X})\big)||^2 \tag{3}$$

where g_{ψ_ℓ} and f_{θ_ℓ} are learned with the hyperparameter setting ℓ, and $\mathbf{Y}_\ell = f_{\theta_\ell}(\mathbf{X})$ (Fig. 1, (a)).

4.3. Graph matrix construction

To construct the graph matrix $\mathbf{S}_\ell$, we use an idea similar to that of *Landmark Spectral Clustering* [21] and the *Anchor-Graphs* [22], where a smaller and sparser representation matrix $\mathbf{Z}_\ell \in \mathbb{R}^{n\times p}$ that approximates a full $n \times n$ affinity matrix is built between the landmarks $\{\mathbf{u}_j^\ell\}_{j\in[1,p]}$ and the encoded points $\{\mathbf{y}_i^\ell\}_{i\in[1,n]}$ (Fig. 1, (a)). Specifically, a set of p points ($p \ll n$) are obtained through a k-means clustering on the embedding matrix $\mathbf{Y}_\ell$. These points are the landmarks which approximate the neighborhood structure. Then a non-linear mapping from data to landmark is computed as follows,

$$z_{ij}^\ell = \Phi(\mathbf{y}_i^\ell) = \frac{\mathcal{K}(\mathbf{y}_i^\ell, \mathbf{u}_j^\ell)}{\sum_{j'\in N_{(i)}} \mathcal{K}(\mathbf{y}_i^\ell, \mathbf{u}_{j'}^\ell)}; \quad j' \in N_{(i)} \tag{4}$$

where $N_{(i)}$ indicates the r ($r < p$) nearest landmarks around $\mathbf{y}_i^\ell$. As proposed in [21], we set z_{ij}^ℓ to zero when the landmark $\mathbf{u}_j^\ell$ is not among the nearest neighbor of $\mathbf{y}_i^\ell$, leading to a sparse affinity matrix $\mathbf{Z}_\ell$. The function $\mathcal{K}(.)$ is used to measure the similarity between data $\mathbf{y}_i^\ell$ and anchor $\mathbf{u}_j^\ell$ with L_2 distance in Gaussian kernel space $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma^2)$, and σ is the bandwidth parameter. The normalized matrix $\hat{\mathbf{Z}}_\ell \in \mathbb{R}^{n\times p}$ is then utilized to obtain a low-rank graph matrix,

$$\mathbf{S}_\ell \in \mathbb{R}^{n\times n}, \quad \mathbf{S}_\ell = \mathbf{Z}_\ell \Sigma^{-1} \mathbf{Z}_\ell^\top \text{ where } \Sigma = diag(\mathbf{Z}_\ell^\top \mathbb{1}).$$

As the Σ^{-1} normalizes the constructed matrix, $\mathbf{S}_\ell$ is bi-stochastic, *i.e.* the summation of each column and row equal to one, and the graph Laplacian becomes,

$$\mathbf{S}_\ell = \hat{\mathbf{Z}}_\ell \hat{\mathbf{Z}}_\ell^\top \quad \text{where} \quad \hat{\mathbf{Z}}_\ell = \mathbf{Z}_\ell \Sigma^{-1/2}. \tag{5}$$

4.4. Ensemble of affinity matrices

Given a set of m encodings $\{\mathbf{Y}_\ell\}_{\ell\in[1,m]}$ obtained using m DAE trained with different hyperparameters setting ℓ, the goal is to merge the m graph similarity matrices $\mathbf{S}_\ell$ in an ensemble similarity matrix which contains information provided by the m embeddings. To aggregate the different similarity matrices, we use an Ensemble Clustering idea analogous to that proposed in [30, 23] where a *co-association* matrix is first built as the summation of all basic similarity matrices, and where each basic partition matrix can be represented as a block diagonal matrix. Thus, the `SC-EDAE` ensemble affinity matrix is built as the summation of the m basic similarity matrices using the following formula,

$$\bar{\mathbf{S}} = \frac{1}{m} \sum_{\ell=1}^{m} \mathbf{S}_\ell. \tag{6}$$

Note that the obtained matrix, $\bar{\mathbf{S}}$ is still bi-stochastic. For many natural problems, $\bar{\mathbf{S}}$ is approximately block stochastic matrix, and hence the first k eigenvectors of $\bar{\mathbf{S}}$ are approximately piecewise constant over the k almost invariant rows subsets [31].

In the sequel, we aim to compute, at lower cost, $\mathbf{B}$ that is shared by the m graph matrices $\mathbf{S}_\ell$, and obtained by optimizing the following trace maximization problem

$$\max_{\mathbf{B}} Tr(\mathbf{B}^\top \bar{\mathbf{S}} \mathbf{B}) \quad s.t. \quad \mathbf{B}^\top \mathbf{B} = \mathbf{I}. \tag{7}$$

4.5. Proposed optimization and algorithm

The solution of Eq. 7 is to set the matrix $\mathbf{B}$ equal to the k eigenvectors corresponding to the largest k eigenvalues of $\bar{\mathbf{S}}$. However, as the computation of the eigen decomposition of $\bar{\mathbf{S}}$ of size $(n \times n)$ is $O(n^3)$, relying on proposition 4.1, we propose instead to compute the k left singular vectors of the concatenated matrix,

$$\bar{\mathbf{Z}} = \frac{1}{\sqrt{m}}[\hat{\mathbf{Z}}_1 | \dots | \hat{\mathbf{Z}}_j | \dots | \hat{\mathbf{Z}}_m]. \tag{8}$$

Using the sparse matrix $\bar{\mathbf{Z}} \in \mathbb{R}^{n \times \sum_{j=1}^m \ell_j}$ with $\sum_{j=1}^m \ell_j \ll n$, instead of $\bar{\mathbf{S}}$, which has a larger dimension, naturally induces an improvement in the computational cost of $\mathbf{B}$ (Fig. 1, (b)).

Proposition 4.1. *Given a set of m similarity matrices $\mathbf{S}_\ell$, such that each matrix $\mathbf{S}_\ell$ can be expressed as $\mathbf{Z}_\ell \mathbf{Z}_\ell^\top$. Let $\bar{\mathbf{Z}} \in \mathbb{R}^{n \times \sum_{j=1}^m \ell_j}$, where $\sum_{j=1}^m \ell_j \ll n$, denoted as $\frac{1}{\sqrt{m}}[\mathbf{Z}_1 | \dots | \mathbf{Z}_j | \dots | \mathbf{Z}_m]$, be the concatenation of the $\mathbf{Z}_\ell$'s, $\ell = 1, \dots, m$. We first have,*

$$\max_{\mathbf{B}^\top \mathbf{B} = \mathbf{I}} Tr(\mathbf{B}^\top \bar{\mathbf{S}} \mathbf{B}) \Leftrightarrow \min_{\mathbf{B}^\top \mathbf{B} = \mathbf{I}, \mathbf{M}} ||\bar{\mathbf{Z}} - \mathbf{B}\mathbf{M}^\top||_F^2. \tag{9}$$

Then, given $\texttt{SVD}(\bar{\mathbf{Z}})$, $\bar{\mathbf{Z}} = \mathbf{U}\Sigma\mathbf{V}^\top$ and the optimal solution $\mathbf{B}^*$ is equal to $\mathbf{U}$.

Proof. From the second term of Eq. 9, one can easily show that $\mathbf{M}^* = \bar{\mathbf{Z}}^\top \mathbf{B}$. Plugging now the expression of $\mathbf{M}^*$ in Eq. 9, the following equivalences hold

$$\begin{aligned}
\min_{\mathbf{B}^\top \mathbf{B} = \mathbf{I}, \mathbf{M}} ||\bar{\mathbf{Z}} - \mathbf{B}\mathbf{M}^\top||_F^2 \quad &\Leftrightarrow \quad \min_{\mathbf{B}^\top \mathbf{B} = \mathbf{I}} ||\bar{\mathbf{Z}} - \mathbf{B}\mathbf{B}^\top \bar{\mathbf{Z}}||_F^2 \\
&\Leftrightarrow \quad \max_{\mathbf{B}^\top \mathbf{B} = \mathbf{I}} Tr(\mathbf{B}^\top \bar{\mathbf{Z}} \bar{\mathbf{Z}}^\top \mathbf{B}) \\
&\Leftrightarrow \quad \max_{\mathbf{B}^\top \mathbf{B} = \mathbf{I}} Tr(\mathbf{B}^\top \bar{\mathbf{S}} \mathbf{B}).
\end{aligned}$$

On the other hand, $\texttt{SVD}(\bar{\mathbf{Z}})$ leads to $\bar{\mathbf{Z}} = \mathbf{U}\Sigma\mathbf{V}^\top$ (with $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$, $\mathbf{V}^\top\mathbf{V} = \mathbf{I}$) and therefore to the eigen-decomposition of $\bar{\mathbf{S}}$ as follows:

$$\begin{aligned}\bar{\mathbf{S}} = \bar{\mathbf{Z}}\bar{\mathbf{Z}}^\top &= (\mathbf{U}\Sigma\mathbf{V}^\top)(\mathbf{U}\Sigma\mathbf{V}^\top)^\top \\ &= \mathbf{U}\Sigma(\mathbf{V}^\top\mathbf{V})\Sigma\mathbf{U}^\top \\ &= \mathbf{U}\Sigma^2\mathbf{U}^\top.\end{aligned}$$

Thereby the left singular vectors of $\bar{\mathbf{Z}}$ are the same as the eigenvectors of $\bar{\mathbf{S}}$. □

The steps of our `SC-EDAE` algorithm are summarized in Algorithm 1 and illustrated by Figure 1. The `SC-EDAE` approach proposes a unique way to combine DAE encodings with clustering. It also directly benefits from the low complexity of the *anchors* strategy for both the graph affinity matrix construction and the eigen-decomposition.

Specifically, the computational cost for the construction of each $\mathbf{Z}_\ell$ affinity matrix amounts to $\mathcal{O}(np_\ell e(t+1))$ (Alg. 1, step (b)) , where n is the number of datapoints, p_ℓ is the number of landmarks for the ℓ^{th} DAE ($p_\ell \ll n$), e is the size of the DAE encoding $\mathbf{Y}_\ell$ ($e \ll n$) and t is the number of iterations for the *k-means* that is used to select the landmarks. It is worth noting that the computation of the $\mathbf{Z}_\ell$ matrices can be easily parallelized over multiple cores, thus limiting the computation time of the ensemble affinity matrix $\bar{\mathbf{Z}}$ to the most time consuming $\mathbf{Z}_\ell$. Furthermore, the eigen-decomposition of the sparse ensemble affinity matrix $\bar{\mathbf{Z}}$, which leads to the $\mathbf{B}$ embeddings (Alg. 1, step (c)), induces a computational complexity of $\mathcal{O}(p'^3 + p'2n)$, where p' is the sum of all landmarks numbers for the concatenated $\mathbf{Z}_\ell$ matrices, *i.e.* $p' = \sum_{j=1}^{m} \ell_j \ll n$. Finally, we need additional $\mathcal{O}(nctk)$ for the last *k-means* on $\mathbf{B} \in \mathbb{R}^{n\times k}$ (Alg. 1, output), where c is the number of centroïds, usually equal to k the number of eigenvectors, leading to $\mathcal{O}(ntk^2)$. The originality and efficiency of our ensemble method hinges on the replacement of a costly eigen-decomposition on $\bar{\mathbf{S}} \in \mathbb{R}^{n\times n}$ by an eigen-decomposition on a low-dimensional and sparse matrix $\bar{\mathbf{Z}} \in \mathbb{R}^{n\times\sum_{j=1}^{m}\ell_j}$, with $\sum_{j=1}^{m}\ell_j \ll n$ (Alg. 1, step (c)). In particular, the sparsity of $\bar{\mathbf{Z}}$ enables the use of fast iterative and partial eigenvalue decomposition.

Algorithm 1 : `SC-EDAE` algorithm

Input: data matrix $\mathbf{X}$;

Initialize: m DAE with different hyperparameters setting;

Do:

(a) Generate m deep embedding $\{\mathbf{Y}_\ell\}_{l\in[1,m]}$ *(Eq. 3)*

(b) Construct the ensemble sparse affinity matrix $\bar{\mathbf{Z}} \in \mathbb{R}^{n\times\sum_{j=1}^{m}\ell_j}$ *(Eq. 4, 8)*

(c) Compute $\mathbf{B}^* \in \mathbb{R}^{\mathbf{n}\times\mathbf{k}}$ by performing *sparse* `SVD` on $\bar{\mathbf{Z}}$ *(Eq. 9)*

Output: Run k-means on $\mathbf{B}^*$ to get the final clustering

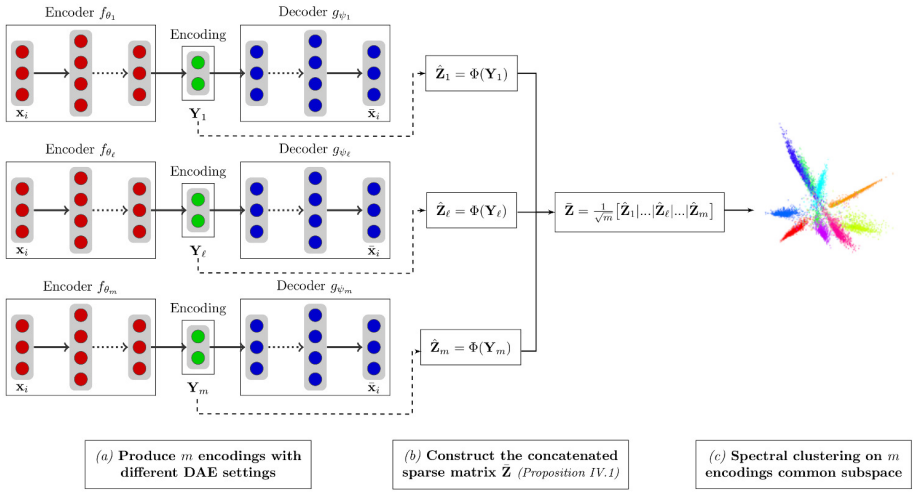


Figure 1: **Scheme of SC-EDAE.** The `SC-EDAE` algorithm computes first m encodings from DAE with different hyperparameters settings (a), then generates m sparse affinity matrix, $\{\hat{\mathbf{Z}}_\ell\}_{\ell\in[1,m]}$, that are concatenated in $\bar{\mathbf{Z}}$ (b), and finally performs a `SVD` on the ensemble graph affinity matrix $\bar{\mathbf{Z}}$ (c).

5. Experiments

5.1. Deep autoencoders settings

For our experiments, we trained fully connected autoencoders with an encoder f_θ of three hidden layers of size 50, 75 or 100 for synthetic datasets (`Tetra`, `Chainlink` and `Lsun`; Section 5.3), and three hidden layers of size 500, 750 or

1000 for real datasets (`MNIST`, `PenDigits` and `USPS`; Section 5.4), as suggested by Bengio *et al.* [6], in all possible orders. The decoder part g_ψ mirrors the encoder stage f_θ. For each DAE architecture (e.g., $\{750-500-1000\}$, $\{100-50-75\}$), 5 encodings were generated with 50, 100, 150, 200 and 250 epochs for real datasets and 200 epochs for synthetic datasets. The weights initialisation follows the Glorot's approach and all encoder/decoder pairs used rectified linears units (ReLUs), except for the output layer which requires a sigmoid function. The autoencoder data are systematically L_2 normalized. We configure the autoencoders using the `Keras tensorflow` Python package, and compile the neural network with binary cross-entropy loss and Adam optimizer [32] with the default `Keras` parameters.

5.2. SC-EDAE ensemble strategy

The ensemble strategy of `SC-EDAE` exploits the encodings $\{\mathbf{Y}_\ell\}_{\ell\in[1,m]}$vwhich are generated with either *(i)* m different DAE initialisations or m different DAE epochs number in association with one DAE structure (e.g. d–500–1000–750–e, with d and e the input and encoding layers width resp.), or *(ii)* m DAE with different structures for the same number of landmarks and epochs. In both cases, the `SC-EDAE` strategy enables to compute the m different sparse affinity matrices $\{\hat{\mathbf{Z}}_\ell\}_{\ell\in[1,m]}$ (Eq. 4) and, following Proposition 4.1, generate the ensemble affinity matrix $\bar{\mathbf{Z}}$ (Eq. 8).

5.3. Synthetic datasets

As a first step, we focus on synthetic datasets to illustrate the `SC-EDAE` algorithm and show the class-separability information embedded in the left singular vectors matrix of $\bar{\mathbf{Z}}$, noted as $\mathbf{B}^\star$ (Prop. 4.1 and Alg.1). We used generated synthetic data sets selected from the Fundamental Clustering Problem Suite (FCPS)[1]. FCPS yields some hard clustering problems, a short description of

[1]The suite can be downloaded from the website of the author: http://www.uni-marburg.de/fb12/datenbionik/data

`Tetra`, `Chainlink` and `Lsun` FCPS data sets and the inherent problems related to clustering are given in Table 1. Following the experiments on synthetic data proposed by Yang *et al.* [11], we transformed the low-dimensional FCPS data, $\mathbf{h}_i \in \mathbb{R}^2$ or $\mathbb{R}^3$, in high-dimensional datapoints, $\boldsymbol{x_i} \in \mathbb{R}^{100}$. Specifically, the $\mathbf{x}_i$ are transformed based on the following equation,

$$\mathbf{x}_i = \sigma(\mathbf{U}\sigma(\mathbf{W}\mathbf{h}_i)) \tag{10}$$

where the entries of matrices $\mathbf{W} \in \mathbb{R}^{10\times 2}$ and $\mathbf{U} \in \mathbb{R}^{100\times 10}$ follow the zero-mean unit-variance i.i.d. Gaussian distribution, and the sigmoid function $\sigma(.)$ introduces nonlinearity.

Table 1: Description of the used FCPS data sets.

Data sets	Characteristics			
	Samples	Features	Clusters	Main Problem
Tetra	400	3	4	inner vs inter cluster distances
Chainlink	1000	3	2	not linear separable
Lsun	400	2	3	different variances

5.4. Real datasets

Our `SC-EDAE` algorithm (Alg.1) is fully evaluated on three image datasets, namely `MNIST` (Modified National Institute of Standards and Technology), `PenDigits` (Pen-Based Recognition of Handwritten Digits) and `USPS` (U.S. Postal Service) and their DAE encodings (see Section 5.1 for details).

MNIST The database is loaded from the `Keras` Python package. The training and testing sets contain respectively $60,000$ and $10,000$ images of size 28×28 of the integers in range $0-9$. The images are of grayscale levels rescaled within $[0, 1]$ by dividing by 255.

PenDigits The training and testing sets contain respectively $7,494$ and $3,498$ images of size 16×16 of the integers in range $0-9$. The images with 16 numeric attributes rescaled within $[0, 1]$ by dividing by 100.

USPS The database is prepared as proposed in [16] and contains $9,298$ images of size 16×16 pixels of the 10- digits (integers in range $0-9$) rescaled within $[0, 1]$.

The classes distribution for each dataset is given in Table 2. `MNIST` and `PenDigits` are balanced-class datasets while `USPS` has an imbalanced distribution.

Table 2: Class distribution for `MNIST`, `PenDigits` and `USPS` datasets.

	0	1	2	3	4	5	6	7	8	9
MNIST	5923	6742	5958	6131	5842	5421	5918	6265	5851	5949
PenDigits	780	779	780	719	780	720	720	778	719	719
USPS	1194	1005	731	658	652	556	664	645	542	644

5.5. Experiment results

5.5.1. Evaluation on synthetic data

Synthetic data enable us to easily explore the separability capacity of the embeddings matrix $\mathbf{B}$. For the experiments related to synthetic data, `SC-EDAE` is used in its ensemble structure version, with $m = 6$ encodings from different structures, and the number of landmarks is set to 100. Applying `SC-EDAE` on the data sets `Tetra`, `Chainlink` and `Lsun`, we note that the 2D representations of the obtained clusters reflect the real cluster structure (Fig. 2 a, b, c; projection on the two first components of the matrix $\mathbf{B}$ as computed in Alg.1, step c). The `SC-EDAE` accuracy is of 1.00 for `Tetra` and `Chainlink`, and 0.90 for `Lsun`. The colored labels correspond to the predicted clusters. Supplementary experiments with different transformation functions (Appendix A.1) and FCPS datasets (Appendix A.2) confirm this trend.

5.5.2. Baseline evaluations on real data

As baseline, we first evaluate *k-means* and *LSC* [21] on the three real datasets. The $kmeans_{++}$ approach corresponds to the `scikit-learn` Python package *k-means* implementation with the default parameters and $kmeans_{++}$ initialisation scheme [33]. We implemented the *LSC* method in Python, following the Matlab implementation proposed in [21], and kept the same default

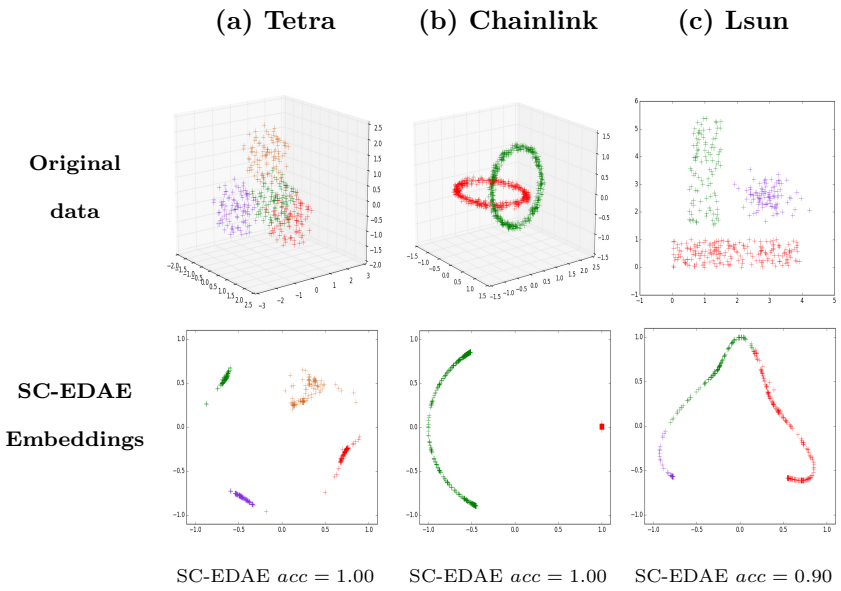


Figure 2: Visualization of the `SC-EDAE` embeddings on `Tetra`, `Chainlink` and `Lsun` datasets The two first components of $\mathbf{B}$ (Alg.1, step (c)) gives a visualization of the datapoints separability with the `SC-EDAE` method. Colors indicate the predicted labels.

parameters. The *LSC* landmarks initialisation is done with *k-means*, which has been shown to provide better accuracy results than the random initialisation [21, 9]. We consider landmarks number within 100 and 1000, by step of 100. The evaluations are done either on the original datasets (Table 3, columns *LSC* and $kmeans_{++}$) or on the encodings (Table 3, columns *DAE-LSC* and *DAE-*$kmeans_{++}$). The accuracy reported for *LSC* and $k\text{-}means_{++}$ corresponds to the mean over 10 clustering replicates on the original datasets, over all epoch and landmark numbers. The accuracy reported for *DAE-LSC* and *DAE-*$kmeans_{++}$ corresponds to an average over 50 replicates (10 replicates on each of the 5 encodings per DAE structure), over all epoch and landmark numbers (see annexes for complementary results per DAE structure, Section Appendix A.4).

As can be seen from Table 3 and already reported in [21], *LSC* outperforms $kmeans_{++}$ for the clustering task on the three datasets (bold values, columns *LSC* and $kmeans_{++}$), yet with larger standard deviations. The same trend is

observed when applying *LSC* and *kmeans*$_{++}$ on encodings, with standard deviations of similar magnitude for both clustering methods (bold values, columns *DAE-LSC* and *DAE-kmeans*$_{++}$).

Table 3: **Mean clustering accuracy for *LSC* and *k-means* on original real datasets and encodings:** Evaluations on MNIST, PenDigits, USPS data and their encodings. Bold values highlight the higher accuracy values.

Data	LSC	kmeans$_{++}$	DAE structure	DAE-LSC	DAE-kmeans$_{++}$
MNIST	**68.55** ±2.25	55.13 ±0.05	500–750–1000	87.06 ±8.27	76.33 ±7.69
			500–1000–750	90.48 ±5.20	79.22 ±5.93
			750–500–1000	88.31 ±5.46	77.71 ±6.03
			750–1000–500	90.30 ±4.89	79.45 ±5.81
			1000–500–750	**91.54** ±3.06	79.98 ±5.98
			1000–750–500	90.96 ±3.98	77.70 ±5.09
PenDigits	**80.17** ±3.76	73.89 ±3.97	500–750–1000	**85.59** ±2.34	73.64 ±4.00
			500–1000–750	85.11 ±3.15	74.67 ±3.43
			750–500–1000	85.36 ±2.91	73.47 ±3.89
			750–1000–500	85.27 ±2.92	74.64 ±4.01
			1000–500–750	85.02 ±2.72	74.20 ±3.84
			1000–750–500	84.39 ±3.04	73.78 ±3.55
USPS	**77.20** ±1.49	68.36 ±0.08	500–750–1000	81.78 ±8.08	72.85 ±3.52
			500–1000–750	**83.47** ±7.40	73.44 ±3.70
			750–500–1000	79.72 ±6.21	72.46 ±2.78
			750–1000–500	80.29 ±5.70	73.80 ±3.51
			1000–500–750	81.39 ±4.46	74.07 ±3.07
			1000–750–500	83.08 ±5.64	72.41 ±3.06

The results from Table 3 demonstrate that the simple combination of DAE and *LSC* or *k-means* already reaches higher accuracy and smaller standard deviations than without the autoencoder step. These results also show the advantage of associating the DAE encodings with the landmark-based representation over the *k-means* approach for the clustering task (columns *DAE-LSC* and *DAE-kmeans*$_{++}$). In particular, the average accuracy for the `MNIST` and `USPS` datasets varies within $[87.06; 91.54]$ and $[79.72; 83.47]$ respectively for *DAE-LSC* and within $[77.70; 79.98]$ and $[72.41; 74.07]$ respectively for *DAE-kmeans*$_{++}$.

Although the encodings generated by the deep autoencoder improve the clustering accuracy, finding *a priori* the most appropriate DAE structure remains a challenging task. The accuracy may also vary for different landmark and epoch numbers (Table 5, annexes Tables A.8 & A.9). As will be seen in the following

sections, the ensemble strategy of SC-EDAE provides a straightforward way to alleviate these issues and avoid the fine tuning of the DAE hyperparameters.

5.5.3. SC-EDAE ensemble evaluations

The Table 4 summarizes the performance of our *LSC*-based ensemble approach in the two cases detailed in section 5.2. Specifically, the columns *Ens.Init.* and *Ens.Ep.* indicate the clustering accuracy for the case *(i)* with an ensemble approach on the DAE weights initialisation (*Ens.Init.*, $m = 5$) and the DAE training epoch numbers (*Ens.Ep.*, $m = 5$). The clustering accuracy values for the ensemble approach on various DAE structures, *i.e.* case (*ii*), is provided in the column *Ens.Struct.* ($m = 6$).

The SC-EDAE ensemble strategy provides higher clustering accuracy as compare to the baseline evaluations (Table 3). In particular, the mean accuracy values obtained with the ensemble strategy for MNIST, PenDigits and USPS can reach, 95.33 ± 0.07, 87.28 ± 0.48 and 85.22 ± 2.14 respectively, *vs.* 91.54 ± 3.06, 85.59 ± 2.34 and 83.47 ± 7.40 (Table 3).

The SC-EDAE ensemble approach on the DAE structures (*Ens.Struct.*) enables also to reach higher accuracy as compare to the baseline evaluations for MNIST (93.23 ± 0.28 *vs.* 91.54 ± 3.06) and PenDigits (86.44 ± 1.42 *vs.* 85.59 ± 2.34), but with the added benefit of avoiding the fine tuning of a particular DAE structure, which is not possible in an unsupervised context. The SC-EDAE results for USPS with an ensemble on several structures are lower than our reference evaluations (81.78 ± 3.61 *vs.* 83.47 ± 7.40), yet the accuracy value remains fairly high with lower standard deviation.

While the SC-EDAE method aims at providing an ensemble strategy for the deep architecture settings (*Ens.Init.*, *Ens.Ep.* and *Ens.Struct.*, Table 4), it relies also on the *LSC* idea which depends on the number of landmarks. We studied the possibility of an ensemble on the number of landmarks ($m = 5$). As can be seen from Table 5, which provides mean accuracy on 10 replicates, the ensemble strategy enables again to reach high accuracy values as compared to our baseline evaluations. The results still remain dependent from the DAE structure type,

Table 4: **Mean clustering accuracy for SC-EDAE, ensemble on initialisations, epochs number and structures:** Bold values highlight the higher accuracy values.

Dataset	DAE structure	Ens.Init.	Ens.Ep.	Ens.Struct.
MNIST	500–750–1000	89.19 ±0.41	85.54 ±4.30	93.23 ±2.84
	500–1000–750	**95.33** ±0.07	94.34 ±2.68	
	750–500–1000	92.15 ±0.25	92.03 ±3.87	
	750–1000–500	92.65 ±0.13	92.26 ±3.71	
	1000–500–750	94.28 ±0.20	94.57 ±1.48	
	1000–750–500	93.87 ±0.38	**95.25** ±0.59	
PenDigits	500-750-1000	**86.80** ±0.74	87.08 ±1.10	86.44 ±1.42
	500–1000–750	85.95 ±0.73	86.69 ±1.33	
	750–500–1000	86.69 ±0.87	87.27 ±0.60	
	750–1000–500	86.48 ±1.09	86.91 ±1.01	
	1000–500–750	86.75 ±0.64	86.96 ±0.81	
	1000–750–500	86.66 ±0.95	**87.28** ±0.48	
USPS	500–750–1000	80.07 ±1.95	81.36 ±5.09	81.78 ±3.61
	500–1000–750	80.54 ±0.77	82.06 ±3.54	
	750–500–1000	79.49 ±1.19	81.10 ±3.86	
	750–1000–500	79.29 ±1.05	79.88 ±2.69	
	1000–500–750	84.12 ±1.80	81.89 ±3.21	
	1000–750–500	**85.22** ±2.14	**84.96** ±3.29	

in particular for MNIST and USPS, and we would therefore recommend to use SC-EDAE in its ensemble structure version (*ie.*, *Ens.Struct.*).

Table 5: **Mean clustering accuracy for SC-EDAE, ensemble on landmarks:** Bold values highlight the higher accuracy values.

DAE structure	MNIST	PenDigits	USPS
500–750–1000	88.84 ±1.22	**87.31** ±1.13	82.17 ±3.79
500–1000–750	**95.35** ±0.20	87.21 ±0.36	81.96 ±2.74
750–500–1000	92.48 ±1.27	87.16 ±0.99	80.61 ±3.46
750–1000–500	92.53 ±0.76	87.09 ±0.95	80.30 ±1.26
1000–500–750	93.76±1.14	86.67 ±1.40	86.35 ±2.62
1000–750–500	95.08 ±0.17	87.13 ±1.26	**87.32** ±4.85

5.6. Evaluation in terms of NMI and ARI

Evaluating clustering results is not a trivial task. The clustering accuracy[2] is not always a reliable measure when the clusters are not balanced and the number

[2]accuracy is defined as $\left\{\max_{p\in P}\frac{1}{n}\sum_{i=1}^{n}1(y_i=\hat{y}_i)\right\}$, with P the set of all permutations in $[1;K]$, K the number of true clusters, $\{y_i\}$ and $\{\hat{y}_i\}$ the ground truth and predicted label.

of clusters is high. To better appreciate the quality of our approach, in the sequel we retain two widely used measures to assess the quality of clustering, namely the Normalized Mutual Information [30] and the Adjusted Rand Index [34]. Intuitively, NMI quantifies how much the estimated clustering is informative about the true clustering, while the ARI measures the degree of agreement between an estimated clustering and a reference clustering. Higher NMI/ARI is better.

We report in Figure 3 the ARI and NMI values for the three real datasets (`MNIST`, `PenDigits` and `USPS`). The ARI and NMI values are given for the baseline evaluations (*DAE-kmeans*$_{++}$ and *DAE-LSC*; average results over 10 runs), and the various ensemble versions of `SC-EDAE` (*Ens.Init*, *Ens.Ep.* and *Ens.Struct.*; average results over 10 runs for each of the 5 different encodings). The ensemble paradigm of `SC-EDAE` ensures high ARI and NMI results with low standard deviations for all real datasets, even for `USPS` which is an imbalanced-class dataset (Fig. 3, green boxplots). We also detail the ARI and NMI evaluations per DAE structure in annexes, Tables A.8 & A.9. These supplementary results highlight the strong influence of a particular DAE structure on the ARI and NMI values. As an example, the ARI minimal and maximal values for *DAE-LSC* are 73.66 and 77.75 respectively for `USPS`, a difference of 4.09 (Table A.8). Another striking example can be found for the `SC-EDAE` in its ensemble initialisation version (*Ens.Init.*) applied to `MNIST`, where the ARI values fluctuate within a $[81.87; 90.17]$ (Table A.9). Based on these evaluations, and as already mentioned (Section 5.2), we would recommend to use `SC-EDAE` in its ensemble structure version (i.e., *Ens.Struct.*) to alleviate the issue of the DAE structure choice.

5.7. Comparison to deep k-means variants

Several strategies that use deep learning algorithm and *k-means* approaches, sequentially or jointly, have demonstrated accuracy improvement on the clustering task. Among these methods, two approaches can now be considered as state-of-the-art methods, namely IDEC (Improved Deep Embedded Cluster-

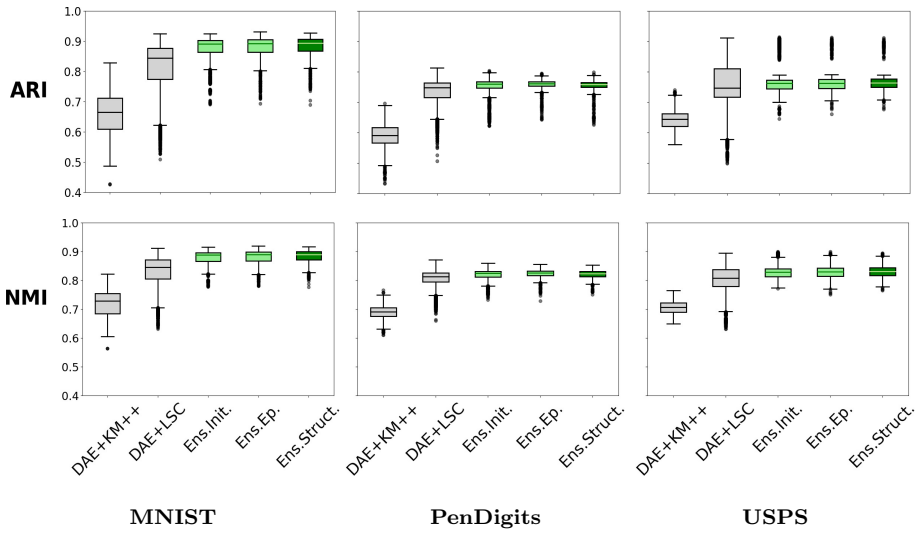


Figure 3: Comparison of Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI) for our `SC-EDAE` approach (ensemble on initialisation, epochs and structures; 10 runs) and baseline methods (combination of deep autoencoders and *k-means* or *LSC*; 10 runs for each of the 5 encodings).

ing) [16] and DCN (Deep Clustering Network) [11]. Very recently, the DKM (Deep *k-means*) algorithm, which applies a *k-means* in an AE embedding space, outperformed these approaches [35].

We compare `SC-EDAE` to these three methods and summarise these evaluations in Table 6. The last six rows of Table 6 are directly extracted from the DKM authors study [35]. The accuracy and NMI values of these six rows are an average over 10 runs. The other values correspond to our evaluations. Specifically, baseline results are given in the first four rows, and correspond to the clustering task via $k\text{-}means_{++}$ or *LSC* (average results over 10 runs), and via a combination of DAE and *k-means* or *LSC* (average results over 10 runs for each of the 5 different encodings). The `SC-EDAE` rows gives the accuracy and NMI results for our ensemble method, with an ensemble over several initialisations (*SC-EDAE Ens.Init.*), epoch numbers (*SC-EDAE Ens.Ep.*) and DAE

Table 6: **Mean clustering accuracy and NMI comparison with deep k-means variants:** Mean accuracy and NMI for MNIST and USPS over 10 replicates with `SC-EDAE` and comparison to baselines and state-of-the-art approaches. Bold values highlight the higher accuracy values.

Model	MNIST		USPS	
	ACC	NMI	ACC	NMI
baselines				
kmeans_{++}	55.13 ±0.05	52.89 ±0.02	68.36 ±0.08	65.67 ±0.10
LSC	68.55 ±2.25	70.54 ±0.83	77.20 ±1.49	79.48 ±0.90
DAE+kmeans_{++}	78.40 ±6.09	71.97 ±4.13	73.17 ±3.27	70.48 ±1.84
DAE+LSC	89.78 ±5.14	83.06 ±4.38	81.62 ±6.25	80.44 ±3.39
no pretraining required				
SC-EDAE Ens.Init.	92.91 ±0.24	87.65 ±0.18	81.46 ±1.48	82.88 ±0.59
SC-EDAE Ens.Ep.	92.33 ±2.77	87.72 ±2.42	**81.88** ±3.62	83.03 ±1.88
SC-EDAE Ens.Struct.	**93.23** ±2.84	**87.93** ±2.27	81.78 ±3.61	**83.17** ±1.96
Deep clustering approaches without pretraining *(Fard et al. 2018) [35]*				
DCN^{np}	34.8 ±3.0	18.1 ±1.0	36.4 ±3.5	16.9 ±1.3
IDEC^{np}	61.8 ±3.0	62.2 ±1.6	53.9 ±5.1	50.0 ±3.8
DKM^{a}	82.3 ±3.2	78.0 ±1.9	75.5 ±6.8	73.0 ±2.3
Deep clustering approaches with pretraining *(Fard et al. 2018) [35]*				
DCN^{p}	81.1 ±1.9	75.7 ±1.1	73.0 ±0.8	71.9 ±1.2
IDEC^{p}	85.7 ±2.4	86.4 ±1.0	75.2 ±0.5	74.9 ±0.6
DKM^{p}	84.0 ±2.2	79.6 ±0.9	75.7 ±1.3	77.6 ±1.1

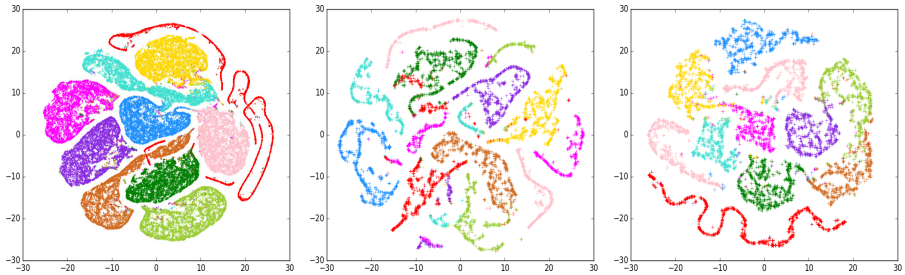
architectures (*SC-EDAE Ens.Struct.*).

As can be seen from Table 6, while our `SC-EDAE` approach does not require any pretraining, it outperforms the DCN and IDEC methods in their pretrained version (Table 6, DCN^{p} and IDEC^{p} results). The DKM method performs well with and without pretraining. Yet, our `SC-EDAE` approach reaches higher accuracy and NMI results than the DKM approach with and without pretraining.

5.8. Visualization of latent space

We investigate the quality of the representation learned with `SC-EDAE` and in particular the positive influence of the left singular vectors matrix of $\bar{\mathbf{Z}}$, $\mathbf{B}$ (Alg.1, step c), on the clustering task. Specifically, we visualize the datapoints nearest-neighbor from the $\mathbf{B}$ matrix using the t-SNE visualization tool [36] that can project embeddings into two components (`TSNE` Python version from the `sklearn` package). The results are given in Figure 4. The t-SNE hyperparameters *perplexity*, *learning rate* and *number of iterations* are set to 40, 200 and

500 for MNIST, and 25, 100 and 400 for PenDigits and USPS, following the recommendations and experimental setup of Maaten *et al.* [36]. For each dataset,



MNIST **PenDigits** **USPS**

Figure 4: t-SNE Vizualization of the embeddings **B** from the SC-EDAE approach on MNIST, PenDigits and USPS datasets. The t-SNE approach provides clustering visualization of the datapoints from the **B** embeddings.

we can observe clearly separated clusters. The ground truth labels nicely match the t-SNE datapoints gathering, highlighting the ability of SC-EDAE to separate data according to the underlying classes. As already noticed in [36], the t-SNE results obtained from the SC-EDAE ensemble affinity matrix reflects the local structure of the data, such as the orientation of the ones, by showing elongated clusters (e.g., Fig. 4, red cluster).

6. Conclusion

We report in this paper a novel clustering method that combines the advantages of deep learning, spectral clustering and ensemble strategy. Several studies have proposed to associate, either sequentially or jointly, deep architecture and classical clustering methods to improve the partitioning of large datasets. However, these methods are usually confronted to important issues related to well known challenges with neural networks, such as weight initialisation or structure settings. Our SC-EDAE approach alleviates these issues by exploiting an ensemble procedure to combine several deep models before applying a spectral clustering; it is quite simple and can be framed in three steps: *(i)* generating m deep embeddings from the original data, *(ii)* constructing a sparse and

low-dimensional ensemble affinity matrix based on anchors strategy, and *(iii)* applying spectral clustering to obtain the common space shared by the m encoding. Specifically, the proposed method enables the use of spectral clustering for large scale datasets by using anchors (or landmarks) that provide a sparse and low-dimensional ensemble affinity matrix. Although the spectral clustering is usually limited in practice due to its computational complexity, the use of landmarks lessen this issue in `SC-EDAE` and efficiently bridges the gap between large-scale spectral clustering and deep learning in an ensemble approach. It is also worth noting that the `SC-EDAE` procedure could easily benefits from the parallelization of the m encodings computation in the first step of the algorithm.

The experiments on real and synthetic datasets demonstrate the robustness and high performance of `SC-EDAE` on image datasets. `SC-EDAE` can be used in different versions with an ensemble on weights initialisations, epoch numbers or deep architectures. These variants provide higher accuracy, ARI and NMI results than state-of-the art methods. Most importantly, the high performance of `SC-EDAE` is obtained *without* any deep models pretraining. Our experiments also show that the encoding ensemble strategy improves on average the evaluation metrics as compare to the deep non-ensemble approach. `SC-EDAE` takes advantage of the diversity from multiple DAE trained with various hyperparameters. By constrast, a model selection procedure would possibly ignore valuable information contained in unselected models. `SC-EDAE` improves the clustering by *fusing* these DAE encodings arising from various settings. Hence, the good performance of the `SC-EDAE` ensemble hinges on the diversity of DAE encodings, in combination with the good individual performance of each encoding.

We expect that an ensemble of all the `SC-EDAE` variations – over initialisations, epochs and structures – would perform better than the *splitted* ensembles. Such `SC-EDAE` *meta-model* would indeed benefit from an increased diversity of individually good encodings and is an interesting continuation of the proposed work. Another perspective is to enable clusters number selection within `SC-EDAE`, as our approach is not primarily meant for this task. Selecting the number of clusters remains an open challenge and the literature is abundant

with approaches that can be applied to the embedding matrix $\mathbf{B}$, as exemplify by the R package `NbClust` which provides 30 indices [37]. The Bayesian information criterion (BIC) was also successfully used with the mixture approach to the problem of determining the components number in a model [38]. We detailed in Appendix A.3 promising results for the clusters number selection within `SC-EDAE`. Finally, future works will also focus on the joint optimization of the feature extraction and clustering tasks, for which benefits have been reported for clustering [11] or anomaly detection [39]. A joint optimization within the ensemble framework of `SC-EDAE` will most likely lead to further improvements.

Appendix A. Appendix

Appendix A.1. Supplementary data transformations on synthetic data

As proposed in [11], we provide two complementary examples of clustering with `SC-EDAE` that demonstrate the ability of the $\mathbf{B}$ embeddings to correctly recover the underlying classes of a given dataset. We first consider the following two transformations, $\mathbf{x}_i = \sigma(\sigma(\mathbf{W}\mathbf{h}_i))^2$ and $\mathbf{x}_i = \tan(\sigma(\mathbf{W}\mathbf{h}_i))$. The Figure A.5 shows the two first embeddings of $\mathbf{B}$ obtained with the transformed data. This representation highlights the separability power of `SC-EDAE`. The corresponding accuracy is 1.00 for `Tetra`, `Chainlink` and `Lsun`. For both supplementary transformation, we can observe patterns that are similar to clusters presented in the main text (Fig. 2).

Appendix A.2. Supplementary experiments on FCPS synthetic datasets

We highlight the good separability capacity of `SC-EDAE` on supplementary FCPS datasets having at least two clusters (Fig. A.6). Details on the associated clustering main problem are provided in Table A.7.

The Figure A.6 substantiates the good separability capacity of the embeddings matrix $\mathbf{B}$ on the FCPS problems, with a clustering accuracy between 0.94 and 1.0, except for the dataset `Target` for which the accuracy reaches 0.76. The `Target` dataset corresponds to a difficult *outliers* problem where 4 out of the 6

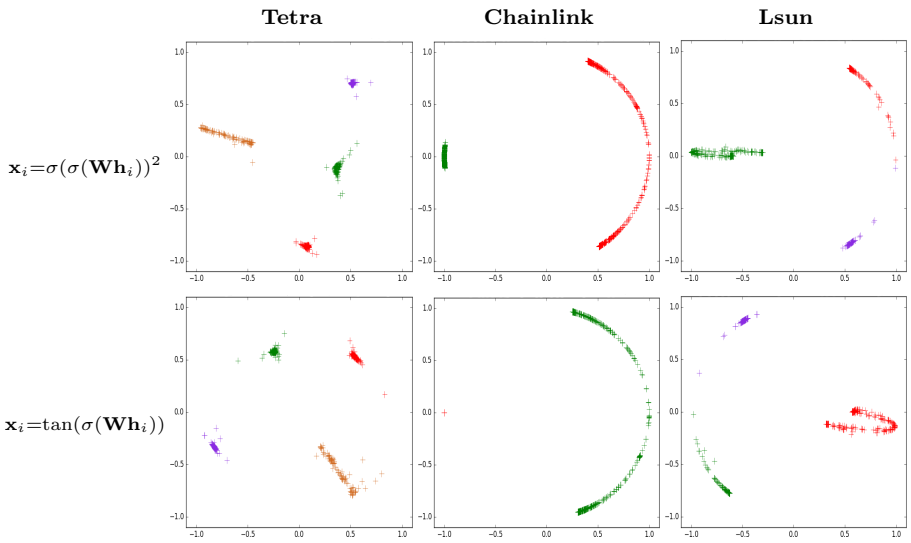


Figure A.5: **Embeddings B from `SC-EDAE` on `Tetra`, `Chainlink` and `Lsun` high-dimensional datasets:** Colors indicate the predicted labels.

Table A.7: Description of the supplementary FCPS data sets.

Data sets	Characteristics			
	Samples	Features	Clusters	Main Problem
Hepta	212	3	7	clearly defined clusters, different variances
TwoDiamonds	800	2	2	cluster borders defined by density
WingNut	1070	2	2	density vs. distance
Atom	800	3	2	different variances and linear not separable
EngyTime	4096	2	2	Gaussian mixture
Target	770	2	6	outliers

clusters contain only 3 datapoints, making the dataset classes strongly imbalanced. `SC-EDAE` correctly identifies all datapoints of the first class (395 out of 395) and about half datapoints of the second class (189 out of 375), but fails to correctly identify the outliers clusters.

Appendix A.3. Experiments on the selection of the number of clusters

As discussed in main text, various algorithms can be easily applied to the embedding matrix $\mathbf{B}$ to obtain the number of clusters within `SC-EDAE`. A possible approach is the `Mean Shift` clustering algorithm [40], which is a general nonparametric technique that can delineate arbitrarily shaped clusters in a com-

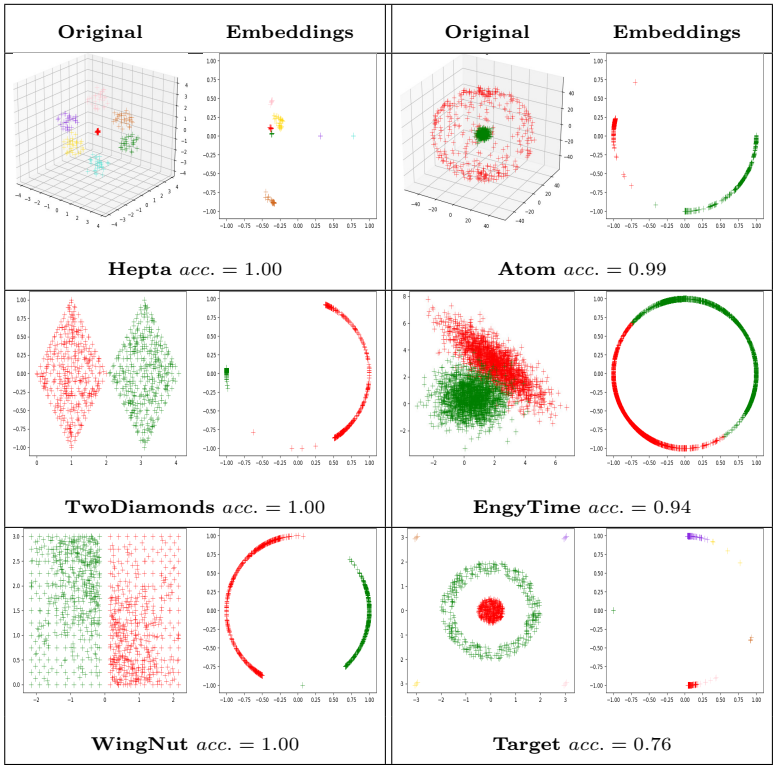


Figure A.6: Visualization of the `SC-EDAE` embeddings and accuracy on FCPS datasets. `SC-EDAE` is used in its ensemble structure version (see main Text, Section 5.3, for details). Colors indicate the predicted labels.

plex multimodal feature space. The use of a method such as `Mean Shift` within `SC-EDAE` makes it possible to automatically obtain the correct number of clusters for almost all supplementary FCPS datasets (see Fig. A.7).

Appendix A.4. Complementary experiments on real data

Appendix A.4.1. Baseline evaluations

The Table A.8 provides complementary results for the baseline evaluations on real datasets. Specifically, it gives the mean Adjusted Rand Index (ARI) and the Normalized Mutual Information (NMI) for *LSC* and $kmeans_{++}$. The mean is taken over 10 replicates on the original datasets, over all epoch and landmark numbers. The results for *DAE-LSC* and *DAE-*$kmeans_{++}$ are averaged over 50

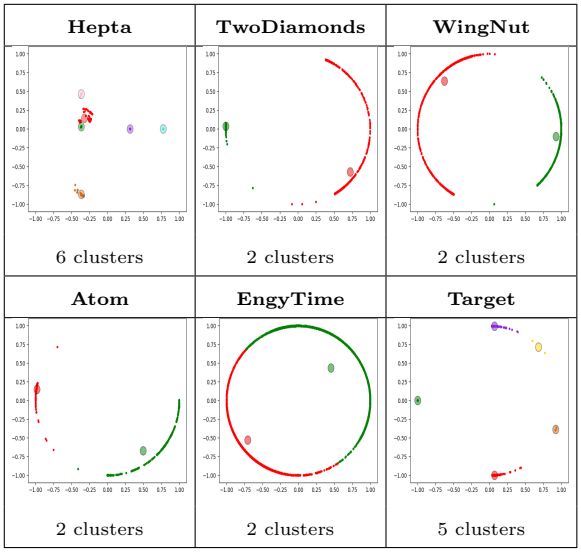


Figure A.7: **`Mean Shift` clustering on `SC-EDAE` embeddings and number of predicted clusters for supplementary FCPS datasets.** Colors indicate the `Mean Shift` predicted labels.

replicates (10 replicates on each of the 5 encodings per DAE structure type), over all epoch and landmark numbers. These results follow the same trend as the accuracy results detailed in main text.

Appendix A.4.2. SC-EDAE ensemble evaluations

The Table A.9 provides complementary results for the ensemble evaluations on real datasets. Specifically, it gives the mean Adjusted Rand Index (ARI) and the Normalized Mutual Information (NMI) for `SC-EDAE`. The mean is taken over 10 replicates on the encodings. The columns *Ens.Init.* and *Ens.Ep.* indicate the results for an ensemble approach on the DAE weight initialisations (*Ens.Init.*, $m = 5$) and the DAE training epoch numbers (*Ens.Ep.*, $m = 5$). The column *Ens.Struct.* provides the evaluations for an ensemble approach on various DAE structure types ($m = 6$).

Table A.8: **Mean clustering Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI) for `LSC` and `k-means` on original real datasets and encodings.** Evaluations on MNIST, PenDigits, USPS datasets. Bold values highlight the higher results

	ARI		**NMI**			**ARI**		**NMI**	
Data	*LSC*	$kmeans_{++}$	*LSC*	$kmeans_{++}$	DAE structure	*DAE-LSC*	*DAE-*$kmeans_{++}$	*DAE-LSC*	*DAE-*$kmeans_{++}$
MNIST	**54.86** ±1.69	39.98 ±0.03	**70.54** ±0.83	52.89 ±0.02	500–750–1000	78.16 ±10.26	63.58 ±8.14	80.88 ±6.58	70.29 ±5.38
					500–1000–750	82.84 ±6.65	67.66 ±6.36	84.04 ±4.25	73.21 ±4.02
					750–500–1000	79.20 ±8.42	65.32 ±7.36	81.57 ±5.55	71.50 ±4.64
					750–1000–500	82.23 ±6.33	66.75 ±6.48	83.52 ±4.13	72.32 ±4.10
					1000–500–750	**83.66** ±4.23	67.48 ±5.80	**84.29** ±2.80	72.80 ±3.52
					1000–750–500	83.15 ±4.81	65.28 ±4.78	84.07 ±2.99	71.69 ±3.09
PenDigits	**68.58** ±3.79	57.58 ±2.61	**79.78** ±1.42	69.72 ±0.58	500–750–1000	**74.12** ±2.53	59.62 ±3.79	**81.06** ±1.43	69.33 ±2.11
					500–1000–750	73.18 ±3.55	58.97 ±3.41	80.46 ±1.85	69.14 ±1.99
					750–500–1000	73.47 ±3.12	58.23 ±3.73	80.55 ±1.48	68.56 ±2.25
					750–1000–500	73.30 ±3.17	58.82 ±3.73	80.38 ±1.62	68.74 ±2.07
					1000–500–750	73.07 ±2.97	58.92 ±3.35	80.23 ±1.79	69.53 ±2.23
					1000–750–500	73.40 ±3.17	58.16 ±3.12	80.66 ±1.60	68.83 ±1.90
USPS	**72.09** ±1.52	57.70 ±0.12	**79.48** ±0.90	65.67 ±0.10	500–750–1000	76.12 ±8.45	63.62 ±3.02	80.32 ±4.89	70.35 ±2.25
					500–1000–750	77.34 ±7.71	64.22 ±3.34	80.69 ±4.30	70.37 ±2.16
					750–500–1000	73.66 ±6.38	63.34 ±2.67	78.77 ±3.81	70.11 ±1.77
					750–1000–500	75.17 ±5.23	64.87 ±2.66	80.13 ±3.11	70.94 ±2.03
					1000–500–750	76.15 ±4.29	64.63 ±2.02	80.98 ±2.12	70.80 ±1.36
					1000–750–500	**77.75** ±5.02	63.88 ±2.07	**81.74** ±2.12	70.33 ±1.45

References

References

[1] M. Yamamoto, H. Hwang, A general formulation of cluster analysis with dimension reduction and subspace separation, Behaviormetrika 41 (1) (2014) 115–129.

[2] K. Allab, L. Labiod, M. Nadif, A semi-NMF-PCA unified framework for data clustering, IEEE Transactions on Knowledge and Data Engineering 29 (1) (2016) 2–16.

[3] K. Allab, L. Labiod, M. Nadif, Simultaneous spectral data embedding and clustering, IEEE Trans. Neural Netw. Learning Syst. 29 (12) (2018) 6396–6401.

Table A.9: **Mean clustering Adjusted Rank Index (ARI) and Normalized Mutual Information (NMI) for the SC-EDAE algorithm.** The ensemble is done on initialisations, epochs number and structures. Bold values highlight the higher results.

		ARI			**NMI**		
Data	DAE structure	*Ens.Init.*	*Ens.Ep.*	*Ens.Struct.*	*Ens.Init.*	*Ens.Ep.*	*Ens.Struct.*
MNIST	500–750–1000	81.87 ±0.49	83.22 ±7.07	87.25 ±3.88	84.69 ±0.28	85.44 ±4.22	87.93 ±2.27
	500–1000–750	**90.17** ±0.14	88.84 ±3.93		**89.59** ±0.10	88.87 ±2.32	
	750–500–1000	84.66 ±1.71	85.29 ±5.18		86.86 ±0.21	86.68 ±3.06	
	750–1000–500	86.18 ±0.20	85.86 ±4.89		87.44 ±0.13	87.17 ±2.66	
	1000–500–750	88.47 ±0.27	88.86 ±2.49		88.53 ±0.17	88.71 ±1.45	
	1000–750–500	88.59 ±0.38	**90.02** ±1.13		88.81 ±0.18	**89.44** ±0.81	
PenDigits	500–750–1000	**75.67** ±0.80	76.15 ±1.23	74.88 ±1.57	**82.33** ±0.48	82.73 ±0.78	81.87 ±0.84
	500–1000–750	74.00 ±0.88	74.83 ±1.82		80.96 ±0.43	81.68 ±0.99	
	750–500–1000	75.13 ±0.95	75.71 ±0.81		81.89 ±0.48	**82.19** ±0.51	
	750–1000–500	74.98 ±1.20	75.38 ±1.18		81.88 ±0.61	81.99 ±0.73	
	1000–500–750	75.07 ±0.69	**75.63** ±0.89		81.86 ±0.39	82.14 ±0.70	
	1000–750–500	75.16 ±1.00	75.60 ±0.76		81.97 ±0.52	82.13 ±0.57	
USPS	500–750–1000	75.68 ±1.80	76.85 ±5.37	77.61 ±3.69	81.93 ±0.81	82.39 ±3.04	83.17 ±1.96
	500–1000–750	76.12 ±0.71	77.67 ±3.63		82.29 ±0.39	83.03 ±1.93	
	750–500–1000	74.32 ±1.02	76.53 ±3.74		81.69 ±0.48	82.07 ±1.87	
	750–1000–500	75.33 ±0.93	75.70 ±2.82		82.34 ±0.43	82.35 ±1.59	
	1000–500–750	79.93 ±1.64	77.73 ±3.00		84.28 ±0.65	83.58 ±1.37	
	1000–750–500	**80.96** ±1.99	**80.79** ±3.21		**84.75** ±0.78	**84.75** ±1.47	

[4] G. E. Hinton, R. Salakhutdinov, Reducing the Dimensionality of Data with Neural Networks, Science 313 (2006) 504–507.

[5] Y. Bengio, L. Yao, G. Alain, P. Vincent, Generalized denoising auto-encoders as generative models, in: Advances in neural information processing systems, 2013, pp. 899–907.

[6] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, in: Advances in neural information processing systems, 2007, pp. 153–160.

[7] F. Tian, B. Gao, Q. Cui, E. Chen, T. Liu, Learning deep representations for graph clustering, in: AAAI 2014, 2014, pp. 1293–1299.

[8] M. Leyli-Abadi, L. Labiod, M. Nadif, Denoising autoencoder as an effective dimensionality reduction and clustering of text data, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2017.

[9] E. Banijamali, A. Ghodsi, Fast spectral clustering using autoencoders and landmarks, in: International Conference Image Analysis and Recognition, Springer, 2017, pp. 380–388.

[10] J. Xie, R. B. Girshick, A. Farhadi, Unsupervised deep embedding for clustering analysis, in: ICML, 2016, pp. 478–487.

[11] B. Yang, X. Fu, N. D. Sidiropoulos, M. Hong, Towards k-means-friendly spaces: Simultaneous deep learning and clustering, in: Proceedings of the 34th ICML, 2017, pp. 3861–3870.

[12] K. Tian, S. Zhou, J. Guan, Deepcluster: A general clustering framework based on deep learning, in: Machine Learning and Knowledge Discovery in Databases, 2017.

[13] L. Yang, X. Cao, D. He, C. Wang, X. Wang, W. Zhang, Modularity based community detection with deep learning, in: Proceedings of the 25^{th} IJCAI, 2016.

[14] M. Seuret, M. Alberti, M. Liwicki, R. Ingold, PCA-initialized deep neural networks applied to document image analysis, in: 14th International Conference on Document Analysis and Recognition, 2017, pp. 877–882.

[15] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, S. Bengio, Why does unsupervised pre-training help deep learning?, J. Mach. Learn. Res 11 (2010) 625–660.

[16] X. Guo, L. Gao, X. Liu, J. Yin, Improved deep embedded clustering with local structure preservation, in: IJCAI, 2017, pp. 1753–1759.

[17] P. Ji, T. Zhang, H. Li, M. Salzmann, I. Reid, Deep subspace clustering networks, in: Advances in Neural Information Processing Systems, 2017, pp. 24–33.

[18] D. Verma, M. Meila, A comparison of spectral clustering algorithms, University of Washington Tech Rep UWCSE030501 1 (2003) 1–18.

[19] M. Filippone, F. Camastra, F. Masulli, S. Rovetta, A survey of kernel and spectral methods for clustering, Pattern recognition 41 (1) (2008) 176–190.

[20] G. Celeux, G. Govaert, A classification em algorithm for clustering and two stochastic versions, Computational statistics & Data analysis 14 (3) (1992) 315–332.

[21] X. Chen, D. Cai, Large scale spectral clustering with landmark-based representation, in: 25^{th} Conference on Artificial Intelligence (AAAI), 2011.

[22] W. Liu, J. He, S.-F. Chang, Large graph construction for scalable semi-supervised learning, in: Proceedings of the 27th ICML, 2010.

[23] S. Vega-Pons, J. Ruiz-Shulcloper, A survey of clustering ensemble algorithms, International Journal of Pattern Recognition and Artificial Intelligence 25 (03) (2011) 337–372.

[24] V. Berikov, I. Pestunov, Ensemble clustering based on weighted co-association matrices: Error bound and convergence properties, Pattern Recognition 63 (2017) 427–436.

[25] B. Hanczar, M. Nadif, Ensemble methods for biclustering tasks, Pattern Recognition 45 (11) (2012) 3938–3949.

[26] P. Baldi, Autoencoders, unsupervised learning, and deep architectures, in: Proceedings of ICML workshop on unsupervised and transfer learning, 2012, pp. 37–49.

[27] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Transactions on pattern analysis and machine intelligence 22 (8) (2000) 888–905.

[28] M. Meila, J. Shi, Learning segmentation by random walks, in: Advances in neural information processing systems, 2001, pp. 873–879.

[29] G. E. Hinton, R. S. Zemel, Autoencoders, minimum description length and Helmholtz free energy, in: Advances in neural information processing systems, 1994, pp. 3–10.

[30] A. Strehl, J. Ghosh, Cluster ensembles — a knowledge reuse framework for combining multiple partitions, J. Mach. Learn. Res. 3 (2003) 583–617.

[31] M. Meila, J. Shi, A random walks view of spectral segmentation, in: Proceedings of the 8^{th} International Workshop on Artificial Intelligence and Statistics, 2001.

[32] S. J. Reddi, S. Kale, S. Kumar, On the convergence of adam and beyond, in: International Conference on Learning Representations, 2018.

[33] D. Arthur, S. Vassilvitskii, k-means++: The advantages of careful seeding, in: Proceedings of the 18^{th} annual ACM-SIAM symposium on Discrete algorithms, 2007, pp. 1027–1035.

[34] D. Steinley, Properties of the Hubert-Arable Adjusted Rand Index., Psychological methods 9 (3) (2004) 386.

[35] M. M. Fard, T. Thonet, E. Gaussier, Deep k-means: Jointly clustering with k-means and learning representations, arXiv preprint arXiv:1806.10069.

[36] L. v. d. Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res 9 (2008) 2579–2605.

[37] M. Charrad, N. Ghazzali, V. Boiteau, A. Niknafs, M. M. Charrad, Package 'nbclust', Journal of statistical software 61 (2014) 1–36.

[38] C. Fraley, A. E. Raftery, How many clusters? which clustering method? answers via model-based cluster analysis, The computer journal 41 (8) (1998) 578–588.

[39] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, H. Chen, Deep autoencoding gaussian mixture model for unsupervised anomaly detection, in: 6th ICLR (Poster), 2018.

[40] D. Comaniciu, P. Meer, Mean shift: A robust approach toward feature space analysis, IEEE Transactions on Pattern Analysis & Machine Intelligence (5) (2002) 603–619.