Published in Image Processing On Line on 2022–12–20.
Submitted on 2022–03–24, accepted on 2022–11–30.
ISSN 2105–1232 © 2022 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2022.398

CAEclust: A Consensus of Autoencoders Representations for Clustering

Séverine Affeldt, Lazhar Labiod, Mohamed Nadif

Centre Borelli UMR 9010, Université Paris Cité, Paris, France
({severine.affeldt, lazhar.labiod, mohamed.nadif}@u-paris.fr)

Communicated by José Lezama *Demo edited by* Jérémy Anger

Abstract

The `CAEclust` Python package implements an original deep spectral clustering in an ensemble framework. Recently, strategies combining classical clustering approaches and deep autoencoders have been proposed, but their effectiveness is impeded by deep network hyperparameters settings. We alleviate this issue with a consensus solution that hinges on the fusion of multiple deep autoencoder representations and spectral clustering. `CAEclust` offers an efficient merging of encodings by using the *landmarks* strategy and demonstrates its effectiveness on benchmark data. `CAEclust` enables to reproduce our experiments and explore novel datasets.

Keywords: spectral clustering; ensemble learning; deep autoencoder

1 Introduction and Related Theory

In the context of unsupervised learning, recent studies improved classical clustering approaches by relying on deep networks for dimensionality reduction, as such networks can automatically learn important features from data [2, 11, 20, 26]. In particular, deep autoencoders (DAE) propose a new data representation or *encoding* using an unsupervised learning. Yet, the effectiveness and ease-to-use of existing DAE clustering strategies are impeded by their sensitivity to hyperparameters. First, network weights initialization adds randomness to the results. Pretraining helps to mitigate this issue, but is computationally intensive. Finding the correct deep architecture is another hurdle. In almost all recent papers on deep clustering, architectures are fine-tuned for each benchmark dataset and lack of technical rationales. Specifically, the architecture is usually derived by cross-validation on a validation set for a grid of hyperparameters [26, 2, 10]. Such determination is infeasible in practice for unsupervised clustering. Other studies reuse architectures that were historically proposed for the considered dataset, without any technical explanations. As an example, the DAE architecture used in [24, 7, 25, 5], which contains an encoder having 3 fully connected layers with 500-500-2000 units and an encoding size of 10, is taken from [21], which is inspired by [16] and [8].

The `CAEclust` package relies on our recent work [1] and practically addresses the above mentioned issues with a consensus optimization that reaches better predictive performance and more relevant

clusterings than single data representation strategies. `CAEclust` embeds the landmarks strategy that strongly diminishes the computational complexity and requires no pretraining. It also fills the gap between ensemble deep autoencoders and spectral clustering in order to propose an effective approach that takes simultaneously advantage of several deep models with various hyperparameters settings. Specifically, we apply spectral clustering on an ensemble of *fused* encodings obtained from m different deep autoencoders.

1.1 Spectral Clustering

A number of spectral clustering algorithms [6, 23], that propose different ways to use the eigenvectors of the normalized graph Laplacian matrix [17] or of the walk's transition matrix [14], can be found in the literature. Their objective function typically favors low similarity between clusters and high similarity within clusters to partition n datapoints of $\mathbf{X} \in \mathbb{R}^{n \times d}$ into k disjoint clusters. The spectral clustering algorithm, in its normalized version, uses the top k eigenvectors of the normalized graph Laplacian matrix. They are the relaxations of the indicator vectors which provide assignments of each datapoint to a cluster. Hence, 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 $\mathbf{S} = \mathbf{D}^{-1/2}\mathbf{K}\mathbf{D}^{-1/2} \in \mathbb{R}^{n \times n}$ 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.

An advantage of the spectral clustering is that it performs well on arbitrary shaped clusters, by contrast with several other clustering algorithms (e.g. *k-means*). However, this approach has also difficulties to handle large-scale datasets. This is the consequence of the high complexity of the graph Laplacian construction and the eigen-decomposition.

Recently, the *Landmark-based Spectral Clustering* (*LSC*) [4] (or *AnchorGraph* [12]) strategy offered a scalable spectral clustering. In particular, *LSC* allows to efficiently construct the graph Laplacian and compute the eigen-decomposition. 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. This, in turn, ensures a more efficient eigen-decomposition as compared to the above mentioned eigen-decomposition of $\mathbf{S}$ (Equation (1)).

1.2 Deep Autoencoders

An autoencoder [9] 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.

We can decompose an autoencoder in two parts, namely an *encoder* f_θ, followed by a *decoder* g_ψ. The first part provides the *encoding* $\mathbf{Y}$ of the input dataset by computing a feature vector $\mathbf{y}_i = f_\theta(\mathbf{x}_i)$ for each input training sample. Then, the encoding is transformed back to its original representation by the *decoder* part, following $\hat{\mathbf{x}}_i = g_\psi(\mathbf{y}_i)$.

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 $\mathcal{J}$ given by

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

where $\mathcal{L}$ is a cost function for measuring the divergence between the input training sample and the reconstructed data. 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.

2 `CAEclust` Algorithm

2.1 Algorithm Overview

We first obtain a set of m encodings $\{\mathbf{Y}_\ell\}_{\ell \in [1,m]}$, given an $n \times d$ data matrix $\mathbf{X}$, using m DAE trained with different hyperparameters settings. Then, for each embedding $\mathbf{Y}_\ell$, we construct a graph matrix $\mathbf{S}_\ell$. We *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, the spectral clustering applied to $\mathbf{S}$ benefits from the common subspace shared by the m deep embeddings. Hence, our proposal hinges on three challenges, namely *(i)* generating m deep embeddings, *(ii)* integrating the clustering in an ensemble learning framework and *(iii)* solving the clustering task in a highly efficient way. The steps of `CAEclust` are summarized in Algorithm 1 and illustrated by Figure 1.

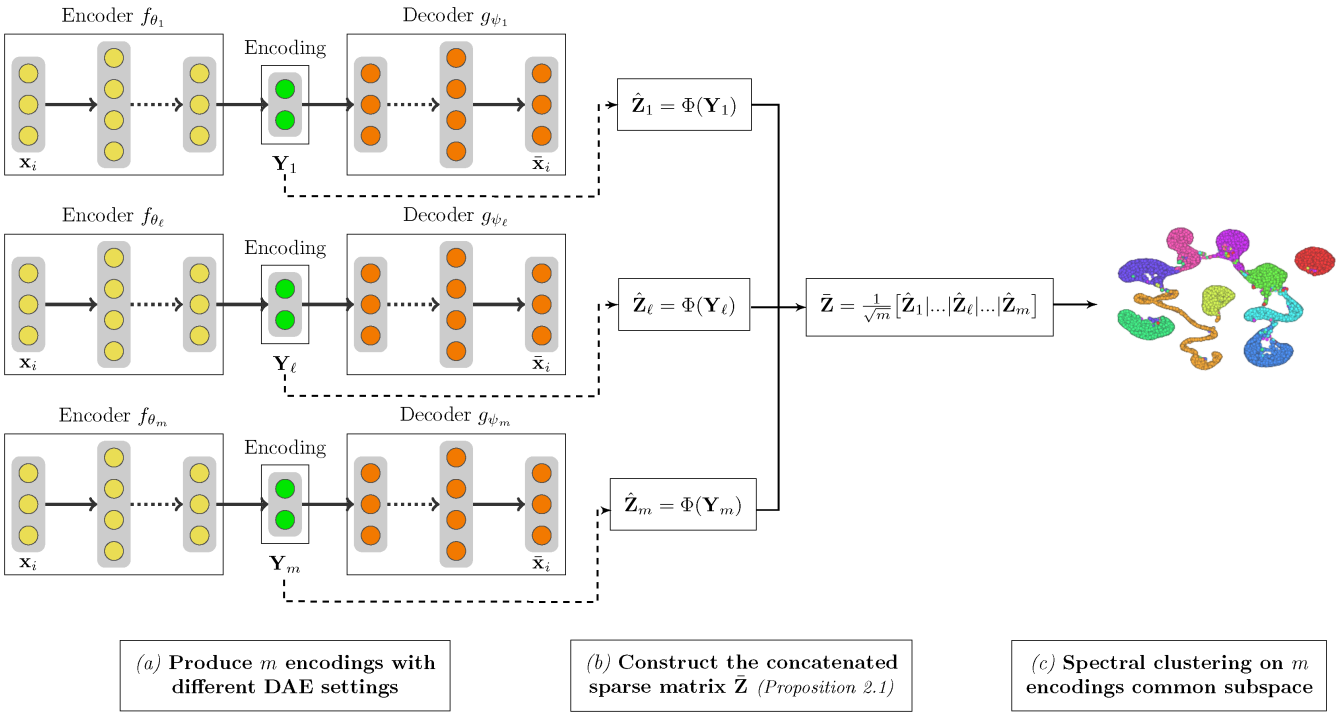


Figure 1: **CAEclust core algorithm overview.** `CAEclust` algorithm first computes m encodings from DAE with different hyperparameters settings (a), then generates m sparse affinity matrices, $\{\hat{\mathbf{Z}}_\ell\}_{\ell \in [1,m]}$, that are concatenated in $\bar{\mathbf{Z}}$ (b), and finally performs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

2.2 Constructing Graph Matrices from Embeddings

An autoencoder is composed of an encoder f_θ and a decoder g_ψ that can have multiple layers of different widths. Its cost function, given by Equation (3), measures the error between the input $\mathbf{x} \in \mathbb{R}^{d\times 1}$ and its reconstruction at the output $\hat{\mathbf{x}} \in \mathbb{R}^{d\times 1}$. We can learn m encodings $\{\mathbf{Y}_\ell\}_{\ell\in[1,m]}$ by training DAE with different hyperparameter settings, g_{ψ_ℓ} and f_{θ_ℓ}, and $\mathbf{Y}_\ell = f_{\theta_\ell}(\mathbf{X})$ (Figure 1, (a)).

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

The *Landmark Spectral Clustering* [4] and the *Anchor-Graphs* [12] strategies compute a smaller and sparser representation matrix $\mathbf{Z}_\ell \in \mathbb{R}^{n\times p}$ that approximates a full $n \times n$ affinity matrix. This matrix is built between the landmarks $\{\mathbf{u}_j^\ell\}_{j\in[1,p]}$ and the encoded points $\{\mathbf{y}_i^\ell\}_{i\in[1,n]}$ (Figure 1, (a)). `CAEclust` proposes to construct the graph matrix $\mathbf{S}_\ell$ following this idea. First, we obtain a set of p points ($p \ll n$), that are the landmarks which approximate the neighborhood structure, through k-means applied on the embedding matrix $\mathbf{Y}_\ell$. 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 [4], 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,

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

2.3 Algorithmic Optimization on the Ensemble of Affinity

`CAEclust` merges m graph similarity matrices, $\mathbf{S}_\ell$, in an ensemble similarity matrix, $\bar{\mathbf{S}}$. This matrix contains the information provided by the encodings $\{\mathbf{Y}_\ell\}_{\ell\in[1,m]}$ obtained using m DAE trained with different hyperparameters setting ℓ. Hence, the `CAEclust` ensemble affinity matrix is built as the summation of the m basic similarity matrices as given by Equation (6). Such aggregation follows an *Ensemble Clustering* idea analogous to that proposed in [19, 22] 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,

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

The $\bar{\mathbf{S}}$ matrix is approximately block stochastic for many natural problems. Hence, the first k eigenvectors of $\bar{\mathbf{S}}$ are approximately piecewise constant over the k almost invariant rows subsets [15]. In the following, we show how `CAEclust` computes, at lower cost, $\mathbf{B}$ that is shared by the m graph matrices $\mathbf{S}_\ell$. The $\mathbf{B}$ matrix is obtained by optimizing the trace maximization problem given by

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

We can solve Equation (7) by setting $\mathbf{B}$ equal to the k eigenvectors corresponding to the largest k eigenvalues of $\bar{\mathbf{S}}$. However, the computation of the eigen-decomposition of $\bar{\mathbf{S}}$ of size $(n \times n)$ is $O(n^3)$. By contrast, `CAEclust` computes instead the k left singular vectors of the concatenated matrix,

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

We naturally obtain an improvement in the computational cost of $\mathbf{B}$ by 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 (Figure 1, (b)).

Proposition 2.1. *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 $\frac{1}{\sqrt{m}}[\mathbf{Z}_1|\ldots|\mathbf{Z}_j|\ldots|\mathbf{Z}_m]$, be the concatenation of the $\mathbf{Z}_\ell$'s, $\ell = 1, \ldots, m$. 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 `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 Equation (9), one can easily show that $\mathbf{M}^* = \bar{\mathbf{Z}}^\top\mathbf{B}$. Plugging now the expression of $\mathbf{M}^*$ in Equation (9), the following equivalences hold

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

On the other hand, `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 $\bar{\mathbf{S}}$. □

With `CAEclust`, we propose a unique way to combine DAE encodings for clustering. Our method also 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 $\mathcal{O}(np_\ell e(t+1))$ (Algorithm 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* algorithm that is used to select the landmarks. It is worth noting that `CAEclust` proposes a parallelized computation of the $\mathbf{Z}_\ell$ matrices over multiple cores. This limits the computation time of the ensemble affinity matrix $\bar{\mathbf{Z}}$ to the most time consuming $\mathbf{Z}_\ell$. Furthermore, the eigen-decomposition of $\bar{\mathbf{Z}}$, which leads to the $\mathbf{B}$ embeddings (Algorithm 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. $p' = \sum_{j=1}^{m} \ell_j \ll n$. The last *k-means* on $\mathbf{B} \in \mathbb{R}^{n \times k}$ (Algorithm 1) requires additional $\mathcal{O}(nctk)$, where c is the number of centroids, usually equal to k the number of eigenvectors, leading to $\mathcal{O}(ntk^2)$.

The originality and efficiency of the `CAEclust` 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 $\sum_{j=1}^{m} \ell_j \ll n$ (Algorithm 1, step (c)). In particular, the sparsity of $\bar{\mathbf{Z}}$ enables the use of fast iterative and partial eigenvalue decomposition.

3 CAEclust Package: Implementation and Evaluation

3.1 Software Architecture and Functionalities

We summarize the package usage in Figure 2 and detail below CAEclust functionalities.

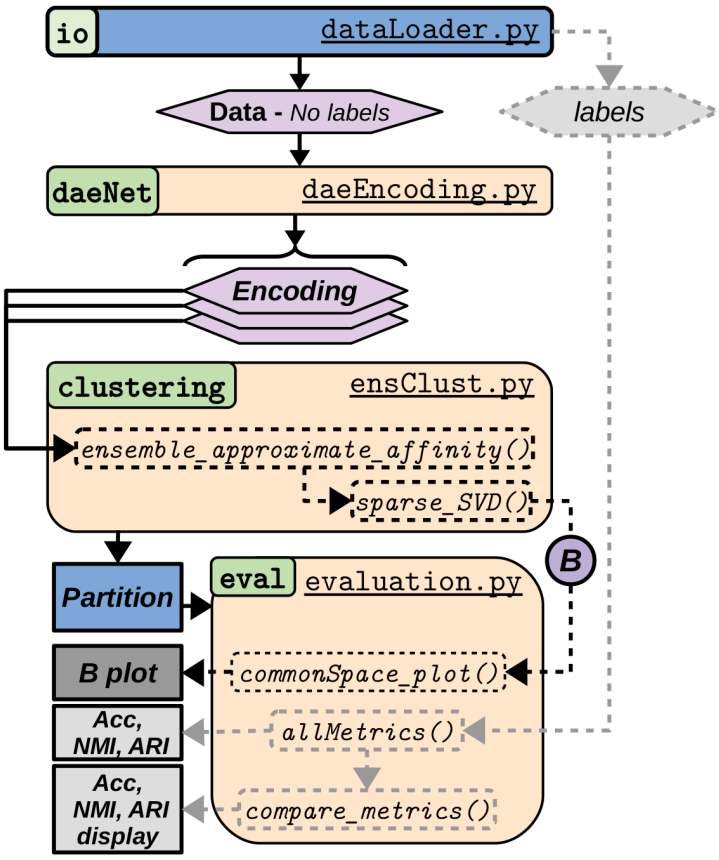


Figure 2: CAEclust package.

[io] `dataLoader.py` loads a dataset. `CAEclust` proposes by default two well-known benchmark images datasets, namely USPS and MNIST. `dataLoader.py` is needed at line 1 of Algorithm 1.

[daeNet] `daeEncoding.py` sets a DAE with general parameters (e.g. optimizer function, number of epochs, batch size, encoding dimension), and any layer number and width. The method `deep_ae()` generates the encoded data. The package proposes to generate the encodings either in serial (`serial_encodings.py`) or in parallel (`parallel_encodings.py`). `daeEncoding.py` is needed at lines 2 and 4 of Algorithm 1.

[clustering] `ensClust.py` computes the ensemble sparse affinity matrix (Equation (4)) through the method `ensemble_approximate_affinity()` (Algorithm 1, line 5). The deep consensus clustering is then obtained from the shared space $\mathbf{B}$ (Equation (8), Equation (7) & *Proposition 2.1*) using the `sparse_SVD()` method (Algorithm 1, line 6).

[evaluation] `evaluation.py` provides 2D and 3D visualizations of the shared space $\mathbf{B}$ with `commonSpace_plot()`. These visualizations are also proposed with the UMAP [13] transformation. When studying benchmark datasets with ground truth labels, `allMetrics()` function can compute three informative metrics (accuracy, NMI [19], and ARI [18]). Furthermore, `compare_metrics()` provides an *html* summary table to easily compare the `CAEclust` results with other clustering algorithms.

3.2 Relating Functionalities and Algorithm

Algorithm 1 provides the `CAEclust` pseudo-code. Each step indicates the associated Python script. The corresponding functions are detailed in Section 3.1.

Algorithm 1 : `CAEclust` core algorithm

1: **Input:** data matrix $\mathbf{X}$ $\rightsquigarrow$ *dataLoader.py*
2: **Initialize:** m DAE with different hyperparameters settings
3: **Do:**
4: (a) Generate m deep embedding $\{\mathbf{Y}_\ell\}_{l\in[1,m]}$ (*Equation* (3))
$\rightsquigarrow$ *daeEncoding.py* [*serial_ encodings.py* | *parallel_ encodings.py*]
5: (b) Construct the ensemble sparse affinity matrix $\bar{\mathbf{Z}} \in \mathbb{R}^{n\times\sum_{j=1}^{m}\ell_j}$ (*Equation* (4), (8))
$\rightsquigarrow$ *ensemble_ approximate_ affinity()*
6: (c) Compute $\mathbf{B}^* \in \mathbb{R}^{\mathbf{n}\times\mathbf{k}}$ by performing *sparse* `SVD` on $\bar{\mathbf{Z}}$ (*Equation* (9))
$\rightsquigarrow$ *sparse_ SVD()*
7: **Output:** Run k-means on $\mathbf{B}^*$ to get the final clustering

`CAEclust` package metadata are summarized by Table 1.

Code metadata description	**metadata**
Current code version	V1.0.0
Legal Code License	GPL-3.0 License
Software code languages, tools, and services used	Python (= 3.8)
Compilation requirements, operating environments & dependencies	Python (= 3.8); packages: scikit-learn, matplotlib, pandas, ipykernel, jinja2, umap-learn, tensorflow (2.6 Windows, otherwise 2.7)
Support email for questions	severine.affeldt@u-paris.fr

Table 1: `CAEclust` package metadata.

3.3 Evaluations

The `CAEclust` ensemble package is evaluated on USPS and MNIST. The sets of encodings are made with different DAE initializations (*Init.*), epoch numbers (*Ep.*) or structures (*Struct.*). We trained fully connected autoencoders with an encoder f_θ of three hidden layers of size 500, 750 or 1000 for `MNIST` and `USPS`, as suggested by Bengio et al. [3], in all possible orders. The decoder part g_ψ mirrors the encoder stage f_θ. The encoding dimension e corresponds to the number of expected classes, which is 10. We consider landmarks numbers within 100 and $1,000$ by step of 100. The accuracy reported for `DAE-LSC` corresponds to an average over 50 replicates (10 replicates on each of the 5 encodings per DAE structure), over all epoch and landmark numbers.

The simple use of LSC on encoded data already improves the LSC performance. Indeed, we found with LSC an average accuracy of 68.55 ± 2.25 and 77.20 ± 1.49 on the original MNIST and USPS datasets respectively, while the use of a single encoding enables to reach an average accuracy between 87.06 ± 8.27 and 91.54 ± 3.06 for MNIST ([500–750–1000] and [1000–500–750] DAE hidden layers) and between 79.72 ± 6.21 and 83.47 ± 7.40 for USPS ([750–500–1000] and [500–1000–750] DAE hidden layers). Yet, finding a priori the most appropriate DAE structure remains a challenging task. The accuracy may also vary for different landmark and epoch numbers. The ensemble strategy provided by `CAEclust` offers a straightforward way to alleviate these issues and avoid the fine tuning

Data	structure	DAE-LSC	CAEclust *no pretraining* Init.	CAEclust *no pretraining* Ep.	CAEclust *no pretraining* Struct.	Deep k-means Variants *no pret.* DCN	*no pret.* IDEC	*no pret.* DKM	*pret.* DCN	*pret.* IDEC	*pret.* DKM
MNIST	500–750–1000	87.06 ±8.27	89.19 ±0.41	85.54 ±4.30	93.23 ± 2.84	34.8 ± 3.0	61.8 ± 3.0	82.3 ± 3.2	81.1 ± 1.9	85.7 ± 2.4	84.0 ± 2.2
	500–1000–750	90.48 ±5.20	**95.33** ±0.07	94.34 ±2.68							
	750–500–1000	88.31 ±5.46	92.15 ±0.25	92.03 ±3.87							
	750–1000–500	90.30 ±4.89	92.65 ±0.13	92.26 ±3.71							
	1000–500–750	**91.54** ±3.06	94.28 ±0.20	94.57 ±1.48							
	1000–750–500	90.96 ±3.98	93.87 ±0.38	**95.25** ±0.59							
USPS	500–750–1000	81.78 ±8.08	80.07 ±1.95	81.36 ±5.09	81.78 ± 3.61	36.4 ± 3.5	53.9 ± 5.1	75.5 ± 6.8	73.0 ± 0.8	75.2 ± 0.5	75.7 ± 1.3
	500–1000–750	**83.47** ±7.40	80.54 ±0.77	82.06 ±3.54							
	750–500–1000	79.72 ±6.21	79.49 ±1.19	81.10 ±3.86							
	750–1000–500	80.29 ±5.70	79.29 ±1.05	79.88 ±2.69							
	1000–500–750	81.39 ±4.46	84.12 ±1.80	81.89 ±3.21							
	1000–750–500	83.08 ±5.64	**85.22** ±2.14	**84.96** ±3.29							

Table 2: Mean clustering accuracy comparisons (*pret. denotes pretraining*).

of the DAE hyperparameters. As can be seen from Table 2, CAEclust provides higher clustering accuracy as compared to the use of a single encoding. In particular, the mean accuracy values obtained with the initialization ensemble strategy (*Init.*, $m = 6$) can reach, 95.33 ± 0.07 for MNIST and 85.22 ± 2.14 for USPS. Results for the epoch ensemble strategy (*Ep.*, $m = 5$, epochs within $[50; 250]$) reach 95.25 ± 0.59 for MNIST and 84.96 ± 3.29 for USPS.

The CAEclust ensemble approach on the DAE structures (*Struct.*, $m = 6$, *all studied structures*) enables also to reach high accuracy as compared to the single combination. For MNIST, it outperforms all the single encoding combination (93.23 ± 0.28 vs. 91.54 ± 3.06). For USPS, it reaches the third best single encoding combination with a better standard deviation (81.78 ± 3.61 vs. 81.78 ± 8.08). These good results are obtained with the added benefit of avoiding the training of a particular DAE structure, which is not possible in an unsupervised context. Figure 3 gives an overview of the common shared space $\mathbf{B}$ for diverse ensemble of encodings and demonstrates the good capacity of $\mathbf{B}$ to separate the classes.

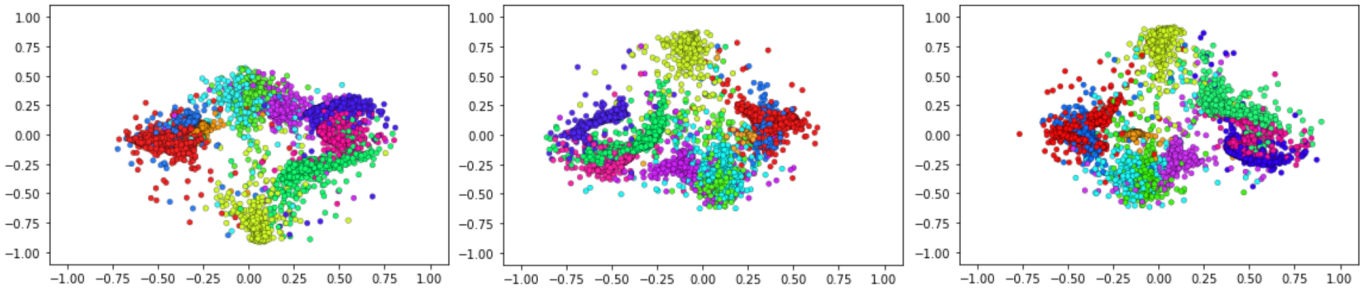


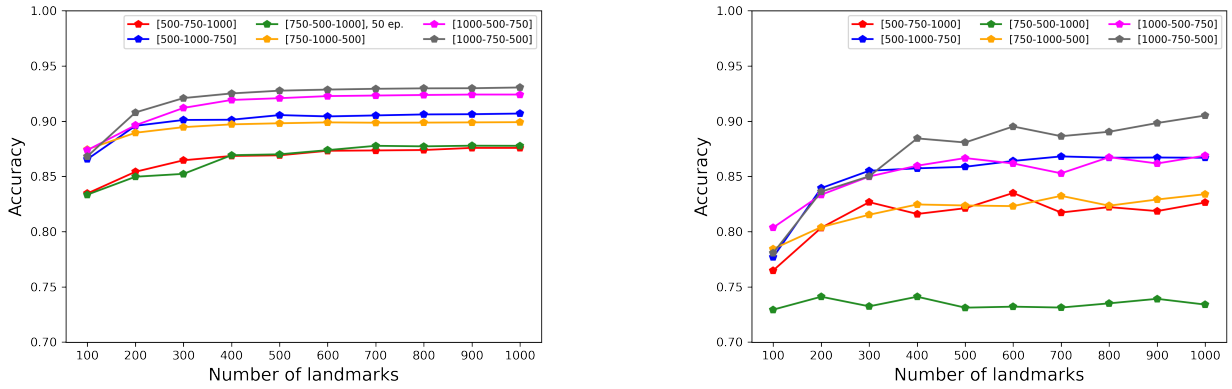
Figure 3: 2D Visualization of the embeddings $\mathbf{B}$ from CAEclust on USPS for an ensemble of encodings based on diverse initializations (*Ens. Init.*), epochs (*Ens. Epoch*) and structures (*Ens. Struct.*). Colors indicate the true USPS digits labels.

Among the existing deep clustering methods, two approaches can now be considered as state-of-the-art methods, namely IDEC (Improved Deep Embedded Clustering) [7] and DCN (Deep Clustering Network) [25]. Recently, the DKM (Deep k-means) algorithm, which applies a k-means in an AE embedding space, outperformed these approaches [5]. While our CAEclust approach does not require

any pretraining for the initialization of the DAE weights, it outperforms the DCN, IDEC and DKM methods in their pretrained version (Table 2).

In particular, DCN treats DAE adjacent layers as restricted Boltzmann machines (RBM) and trains RBMs bottom-up to obtain good initial weights. Then, the network weights are fine-tuned. IDEC uses a greedy layer-wise pre-training with dropout before fine-tuning the entire network without dropout while DKM also relies on a pretraining step of 50 epochs before applying its main algorithm. By contrast, CAEclust only trains the DAEs without any weight initialization that could be obtained from a pretraining strategy. As can be seen from Table 2, none of these challenging deep clustering approaches outperform CAEclust in their *no pretraining* version (see Table 2, *Deep k-means variants / no pret. column*). Besides, even when injecting pretraining in DCN, IDEC or DKM, CAEclust still remains really competitive, as its lowest accuracy mainly outperforms the accuracy of the other approaches. Most importantly, the CAEclust ensemble based on the merging of encodings produced by various structures (Table 2, *Struct.* column) provides a straightforward way to combine embeddings from various DAEs without having to choose a specific architecture (Table 2, *Init.* & *Ep.* columns). With the *Struct.* ensemble, CAEclust also offers a better accuracy than other approaches.

Beyond the DAEs hyperparameters, CAEclust also depends on the number of landmarks that are used for the LSC clustering. As can be seen in [12, 2], the anchor graph strategy reduces the error rate as the number of landmarks increases. As an example, the experiments from [12] demonstrate that the error rate of such an approach reaches a plateau above 800 landmarks on the USPS dataset. In [2], experimental results show that the higher the number of landmarks, the better results for MNIST. Our experiments tend to confirm these behaviors, as can be seen in Figure 4, which provides the accuracy of CAEclust on MNIST and USPS, for different DAE architectures trained during 50 epochs.



MNIST, 50 epochs

USPS, 50 epochs

Figure 4: Mean clustering accuracy per number of landmarks for different DAE architectures.

As complementary experiments, we investigate an ensemble strategy of CAEclust on the number of landmarks. More specifically, for a chosen DAE architecture, we construct an ensemble sparse affinity matrix, $\bar{\mathbf{Z}}$, where each $\hat{\mathbf{Z}}_i$ is based on a different number of anchors (Algorithm 1, step (b); Equation (8)). As an example, we randomly choose $m = 5$ number of landmarks and evaluate the CAEclust accuracy at several numbers of epochs. As can be seen from Table 3, this *landmark variant* of CAEclust provides again better accuracy than the other approaches.

Data	structure	50 epochs	100 epochs	150 epochs	200 epochs	250 epochs
MNIST	500–750–1000	85.44 ±1.07	95.14 ±0.35	85.43 ±3.82	95.37 ±0.24	82.81 ±0.62
	500–1000–750	**94.52** ±0.18	**96.04** ±0.26	**95.62** ±0.19	95.33 ±0.23	95.24 ±0.14
	750–500–1000	92.80 ±2.07	85.74 ±2.88	94.90 ±0.21	94.86 ±0.93	94.09 ±0.26
	750–1000–500	89.19 ±1.23	88.90 ±0.65	94.77 ±0.07	93.80 ±1.61	**95.99** ±0.25
	1000–500–750	89.31 ±2.76	93.12 ±2.50	95.54 ±0.14	95.17 ±0.17	95.65 ±0.15
	1000–750–500	94.09 ±0.22	95.64 ±0.13	94.90 ±0.28	**95.57** ±0.10	95.19 ±0.11
USPS	500–750–1000	79.57 ±4.14	78.35 ±0.78	78.32 ±0.31	90.17 ±6.62	84.42 ±7.10
	500–1000–750	79.18 ±0.31	**85.07** ±5.34	78.92 ±0.53	86.89 ±7.24	79.72 ±0.27
	750–500–1000	77.50 ±0.32	81.27 ±4.11	82.11 ±4.57	80.82 ±4.18	81.36 ±4.13
	750–1000–500	82.40 ±5.00	80.25 ±0.31	80.08 ±0.34	80.01 ±0.23	78.78 ±0.40
	1000–500–750	**94.34** ±0.35	79.60 ±0.18	80.16 ±0.16	**91.39** ±5.75	86.28 ±6.67
	1000–750–500	92.59 ±4.27	80.09 ±0.17	**86.65** ±7.01	89.52 6.39±	**87.73** ±6.42

Table 3: Mean clustering accuracy comparisons (*landmarks ensemble*).

4 Illustrative Examples

`CAEclust` includes two tutorial notebooks. The first one, *Baseline_ evaluations*, shows the influence of the DAE structure on the clustering for a non ensemble approach. The second one, *Ensemble_ evaluations*, demonstrates the effectiveness `CAEclust` consensus deep clustering.

4.1 Baseline Evaluations

The notebook *Baseline_ evaluations* first loads the dataset with `dataLoader.py`. Then, it performs k-means and LSC clusterings on the original data. For LSC, the required `CAEclust` functions are `single_approximate_affinity()` followed by `sparse_SVD()`. We then set several DAEs with different structures to encode data using `daeEncoding.py`. Finally, we perform k-means and LSC on each encoding, evaluate the results with `allMetrics()` and display them using `compare_metrics()`.

Figure 5[1] provides the evaluation metrics for k-means and `LSC` clusterings on single deep representations obtained with various autoencoders structures. As can be seen, the use of a single encoding usually improves the clustering results, both for k-means (Figure 5, *a* & *b*) and LSC (Figure 5, *c* & *d*). However, the results strongly depend on the DAE structure, as exemplified by Figure 6 which provides the UMAP representations for the most efficient deep encoding (left) and the least efficient deep encoding (right) for `MNIST`. In the second notebook, `CAEclust` proposes several ensemble solutions to alleviate such critical hyperparameters influence.

4.2 Ensemble Evaluations

The notebook *Ensemble_ evaluations* first loads the dataset with `dataLoader.py`. Then, it encodes the data with various hyperparameters using `daeEncoding.py` (either in serial or in parallel). To compute the ensemble sparse affinity matrix $\bar{\mathbf{Z}}$, we use `ensemble_approximate_affinity()`. The clustering is obtained from the common shared space using `sparse_SVD()`. We can compare the different ensemble strategies based on the metrics of `allMetrics()` and display them using `compare_metrics()`.

Figure 7[2] provides the evaluations of different `CAEclust` ensemble strategies on MNIST and USPS. As can be seen, all the ensemble strategies (initialization, epoch and structure) enable to reach high accuracy, NMI and ARI.

[1]Tables are images extracted from the outputs of the `CAEclust` *Baseline_ evaluations* tutorial notebooks.
[2]Tables are images extracted from the outputs of the `CAEclust` *Ensemble_ evaluations* tutorial notebooks.

	ACC	NMI	ARI	mean
DAE [500,1000,750] + k-means	0.8932	0.7946	0.7882	*0.8253*
DAE [1000,750,500] + k-means	0.8957	0.7929	0.7857	*0.8248*
DAE [750,500,1000] + k-means	0.8229	0.7336	0.6876	*0.7480*
DAE [1000,500,750] + k-means	0.7776	0.7103	0.6480	*0.7120*
DAE [750,1000,500] + k-means	0.7665	0.6869	0.6219	*0.6918*
DAE [500,750,1000] + k-means	0.6278	0.6215	0.4903	*0.5799*
k-means	0.5129	0.4908	0.3606	*0.4548*

(a) `MNIST` – DAE+k-means

	ACC	NMI	ARI	mean
DAE [1000,750,500] + k-means	0.7409	0.7397	0.6694	*0.7167*
DAE [1000,500,750] + k-means	0.7652	0.7157	0.6610	*0.7140*
DAE [500,1000,750] + k-means	0.7475	0.7229	0.6647	*0.7117*
DAE [500,750,1000] + k-means	0.7306	0.7301	0.6619	*0.7075*
DAE [750,500,1000] + k-means	0.7594	0.6943	0.6383	*0.6973*
DAE [750,1000,500] + k-means	0.7519	0.6661	0.6046	*0.6742*
k-means	0.6765	0.6379	0.5585	*0.6243*

(b) `USPS` – DAE+k-means

	ACC	NMI	ARI	mean
DAE [1000,750,500] + LSC	0.9571	0.8956	0.9080	*0.9202*
DAE [500,1000,750] + LSC	0.9522	0.8873	0.8986	*0.9127*
DAE [1000,500,750] + LSC	0.9378	0.8624	0.8695	*0.8899*
DAE [750,500,1000] + LSC	0.9338	0.8592	0.8622	*0.8851*
DAE [750,1000,500] + LSC	0.8285	0.7628	0.7237	*0.7717*
DAE [500,750,1000] + LSC	0.7719	0.7096	0.6185	*0.7000*
LSC	0.7098	0.7055	0.5937	*0.6697*

(c) `MNIST` – DAE+LSC

	ACC	NMI	ARI	mean
DAE [1000,750,500] + LSC	0.9364	0.8705	0.8874	*0.8981*
DAE [500,1000,750] + LSC	0.9328	0.8631	0.8800	*0.8920*
DAE [750,1000,500] + LSC	0.8959	0.8092	0.8162	*0.8404*
DAE [500,750,1000] + LSC	0.8133	0.8208	0.7639	*0.7993*
LSC	0.7807	0.8127	0.7351	*0.7762*
DAE [1000,500,750] + LSC	0.7918	0.7985	0.7349	*0.7751*
DAE [750,500,1000] + LSC	0.7851	0.7968	0.7352	*0.7724*

(d) `USPS` – DAE+LSC

Figure 5: Clustering on a single deep representation with `CAEclust` package.

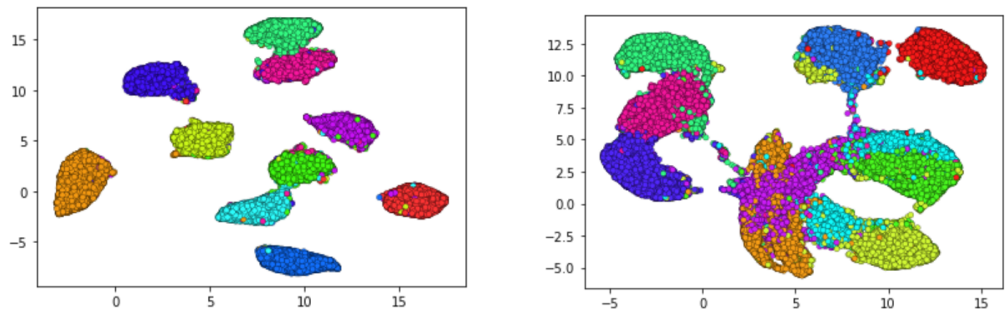


Figure 6: UMAP on *best* (left) and *worse* (right) deep *individual* representation for `MNIST`.

	ACC	NMI	ARI	mean
Ens. Meta	0.9563	0.9034	0.9084	*0.9227*
Ens. Init.	0.9557	0.8980	0.9061	*0.9199*
Ens. Epch.	0.9475	0.8928	0.8929	*0.9111*
Ens. Struct.	0.9471	0.8883	0.8893	*0.9082*

(a) `MNIST` – Consensus results

	ACC	NMI	ARI	mean
Ens. Meta	0.9442	0.8868	0.8996	*0.9102*
Ens. Init.	0.9432	0.8857	0.8987	*0.9092*
Ens. Struct.	0.9409	0.8805	0.8949	*0.9054*
Ens. Epch.	0.9395	0.8775	0.8905	*0.9025*

(b) `USPS` – Consensus results

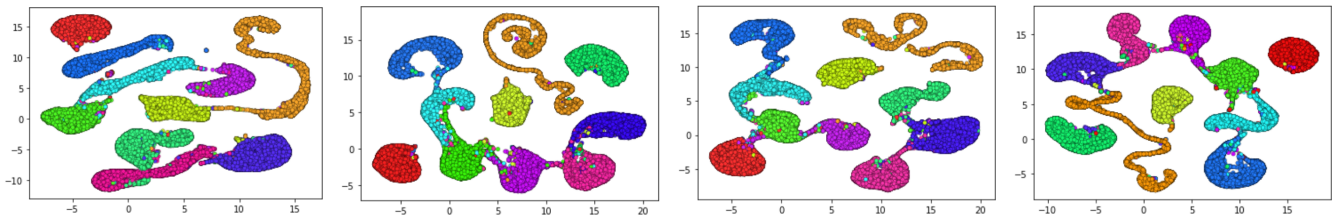
Figure 7: `CAEclust` ensemble evaluations on `MNIST` (left) and `USPS` (right).

In particular, the ensemble structure on MNIST (Figure 7, *a*; *Ens. Struct.*) reaches much better results than the results obtained with the least efficient structure (*Ens. Struct.*, 0.9082 vs. $DAE[500, 750, 1000] + LSC$, 0.70). Furthermore, the ensemble initialization results are comparable to the results obtained with the most efficient encoding (*Ens.Init.*, 0.9199 vs. $DAE[1000, 750, 750] + LSC$,

0.9202).

For USPS, all ensemble strategies outperform the results obtained with the individual DAE structure. Indeed, the worst ensemble results is 0.9025 (Figure 7, b) while the best individual encoding enable to reach a score of 0.8981 (Figure 5, d; $DAE[1000, 750, 500] + LSC$). It is worth noting that *Ens. Epch.* and *Ens. Init.* results are obtained from the DAE structure $[750, 500, 1000]$ which is the worse DAE structure in this example.

The *Ensemble_ evaluations* notebook also proposes a *meta* ensemble strategy, for which the encodings from the initialization, epoch and structure ensemble sets are integrated in the common shared space $\mathbf{B}$. For both `USPS` and `MNIST`, the `CAEclust` *meta* ensemble strategy outperforms the other ensemble strategies as well as their best encoding performance. Figure 8 provides the UMAP transformation of the common share space $\mathbf{B}$ and highlights the good capacity of $\mathbf{B}$ to separate the classes. In practice, one could generate various encodings following a predefined grid and take advantage of all the deep representations to obtain a efficient clustering with very few hyperparameters settings constraints.



`MNIST` *Ens. Struct.* `MNIST` *Ens. Epoch* `MNIST` *Ens. Init.* `MNIST` *Ens. Meta.*

Figure 8: 2D UMAP Visualization of the embeddings $\mathbf{B}$ from `CAEclust` on `MNIST` for an ensemble of encodings based on diverse initializations (*Ens. Init.*), epochs (*Ens. Epoch*) and structures (*Ens. Struct.*). Colors indicate the true digits labels.

5 Conclusions

We developed a Python package `CAEclust` for deep spectral clustering in an ensemble framework. `CAEclust` is capable of learning a consensus solution that hinges on the fusion of multiple deep autoencoder representations and spectral clustering. It is an effective and user friendly package that proposes parallelization to overcome the computation time issue. Future extensions of this work will be dedicated to the challenging text data, which is by nature unstructured, highly dimensional and sparse. All in all, `CAEclust` alleviates the hyperparameters settings issue that usually impedes the efficiency of deep network based clustering techniques, and offers an efficient merging of encodings by using the landmarks strategy.

Acknowledgments

We thank Amine Ferdjaoui (Université Paris Cité, Centre Borelli UMR9010, Paris, France) for his valuable technical assistance in the design and implementation of the CAEclust IPOL demo. This work was supported by a grant overseen by the French National Research Agency (ANR) (ANR-19-CE23-0002). It also received the labelling of *Cap Digital* and *EuroBiomed/Cancer-Bio-Santé* competitiveness clusters.

References

[1] S. Affeldt, L. Labiod, and M. Nadif, *Spectral clustering via ensemble deep autoencoder learning (SC-EDAE)*, Pattern Recognition, (2020), p. 107522. https://doi.org/10.1016/j.patcog.2020.107522.

[2] E. Banijamali and A. Ghodsi, *Fast spectral clustering using autoencoders and landmarks*, in International Conference Image Analysis and Recognition, Springer, 2017, pp. 380–388. http://dx.doi.org/10.1007/978-3-319-59876-5_42.

[3] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, *Greedy layer-wise training of deep networks*, in Advances in Neural Information Processing Systems, 2007, pp. 153–160.

[4] X. Chen and D. Cai, *Large scale spectral clustering with landmark-based representation*, in Conference on Artificial Intelligence (AAAI), 2011.

[5] M.M. Fard, T. Thonet, and E. Gaussier, *Deep k-means: Jointly clustering with k-means and learning representations*, Pattern Recognition Letters, 138 (2020), pp. 185–192. https://doi.org/10.1016/j.patrec.2020.07.028.

[6] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta, *A survey of kernel and spectral methods for clustering*, Pattern Recognition, 41 (2008), pp. 176–190. https://doi.org/10.1016/j.patcog.2007.05.018.

[7] X. Guo, L. Gao, X. Liu, and J. Yin, *Improved deep embedded clustering with local structure preservation*, in International Joint Conference on Artificial Intelligence (IJCAI), 2017, pp. 1753–1759. https://dl.acm.org/doi/10.5555/3172077.3172131.

[8] G.E. Hinton, S. Osindero, and Y-W. Teh, *A fast learning algorithm for deep belief nets*, Neural Computation, 18 (2006), pp. 1527–1554. https://doi.org/10.1162/neco.2006.18.7.1527.

[9] G.E. Hinton and R.S. Zemel, *Autoencoders, minimum description length and Helmholtz free energy*, in Advances in Neural Information Processing Systems, 1994, pp. 3–10.

[10] P. Ji, T. Zhang, H. Li, M. Salzmann, and I. Reid, *Deep subspace clustering networks*, in Advances in Neural Information Processing Systems, 2017, pp. 24–33. https://dl.acm.org/doi/abs/10.5555/3294771.3294774.

[11] M. Leyli-Abadi, L. Labiod, and 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, pp. 801–813. https://doi.org/10.1007/978-3-319-57529-2_62.

[12] W. Liu, J. He, and S-F. Chang, *Large graph construction for scalable semi-supervised learning*, in International Conference on Machine Learning (ICML), 2010. https://dl.acm.org/doi/10.5555/3104322.3104409.

[13] L. McInnes, J. Healy, and J. Melville, *UMAP: Uniform manifold approximation and projection for dimension reduction*, 2018. arXiv preprint https://doi.org/10.48550/arXiv.1802.03426.

[14] M. Meila and J. Shi, *Learning segmentation by random walks*, in Advances in Neural Information Processing Systems, 2001, pp. 873–879.

[15] M. Meila and J. Shi, *A random walks view of spectral segmentation*, in International Workshop on Artificial Intelligence and Statistics, 2001.

[16] R. Salakhutdinov and G. Hinton, *Learning a nonlinear embedding by preserving class neighbourhood structure*, in International Conference on Artificial Intelligence and Statistics, PMLR, 2007, pp. 412–419.

[17] J. Shi and J. Malik, *Normalized cuts and image segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 22 (2000), pp. 888–905. https://doi.org/10.1109/34.868688.

[18] D. Steinley, *Properties of the Hubert-Arable Adjusted Rand Index*, Psychological Methods, 9 (2004), p. 386. https://doi.org/10.1037/1082-989x.9.3.386.

[19] A. Strehl and J. Ghosh, *Cluster ensembles — a knowledge reuse framework for combining multiple partitions*, The Journal of Machine Learning Research, 3 (2003), pp. 583–617. https://dl.acm.org/doi/10.1162/153244303321897735.

[20] K. Tian, S. Zhou, and J. Guan, *Deepcluster: A general clustering framework based on deep learning*, in Machine Learning and Knowledge Discovery in Databases, 2017. https://doi.org/10.1007/978-3-319-71246-8_49.

[21] L. Van Der Maaten, *Learning a parametric embedding by preserving local structure*, in International Conference on Artificial Intelligence and Statistics, PMLR, 2009, pp. 384–391.

[22] S. Vega-Pons and J. Ruiz-Shulcloper, *A survey of clustering ensemble algorithms*, International Journal of Pattern Recognition and Artificial Intelligence, 25 (2011), pp. 337–372. http://dx.doi.org/10.1142/S0218001411008683.

[23] D. Verma and M. Meila, *A comparison of spectral clustering algorithms*, 2003. https://sites.stat.washington.edu/spectral/papers/UW-CSE-03-05-01.pdf.

[24] J. Xie, R.B. Girshick, and A. Farhadi, *Unsupervised deep embedding for clustering analysis*, in International Conference on Machine Learning (ICML), 2016, pp. 478–487. https://dl.acm.org/doi/10.5555/3045390.3045442.

[25] B. Yang, X. Fu, N.D. Sidiropoulos, and M. Hong, *Towards k-means-friendly spaces: Simultaneous deep learning and clustering*, in International Conference on Machine Learning (ICML), 2017, pp. 3861–3870. https://dl.acm.org/doi/10.5555/3305890.3306080.

[26] L. Yang, X. Cao, D. He, C. Wang, X. Wang, and W. Zhang, *Modularity based community detection with deep learning*, in International Joint Conference on Artificial Intelligence (IJCAI), 2016. https://dl.acm.org/doi/10.5555/3060832.3060936.