DEEM: A Novel Approach to Semi-Supervised and Unsupervised Image Clustering under Uncertainty using Belief Functions and Convolutional Neural Networks

Loïc Guiziou[a], Emmanuel Ramasso[a], Sébastien Thibaud[a], Sébastien Denneulin[b]

[a] *Université Marie et Louis Pasteur, SUPMICROTECH, CNRS, institut FEMTO-ST (UMR 6174), F-25000 Besançon, France*
[b] *SAFRAN CERAMICS, rue de Touban, 33185, Le Haillan, France*

Abstract

DEEM (Deep Evidential Encoding of iMages) is a clustering algorithm that combines belief functions with convolutional neural networks in a Siamese-like framework for unsupervised and semi-supervised image clustering. In DEEM, images are mapped to Dempster-Shafer mass functions to quantify uncertainty in cluster membership. Various forms of prior information, including must-link and cannot-link constraints, supervised dissimilarities, and Distance Metric Learning, are incorporated to guide training and improve generalisation. By processing image pairs through shared network weights, DEEM aligns pairwise dissimilarities with the conflict between mass functions, thereby mitigating errors in noisy or incomplete distance matrices. Experiments on MNIST demonstrate that DEEM generalises effectively to unseen data while managing different types of prior knowledge, making it a promising approach for clustering and semi-supervised learning from image data under uncertainty.

Keywords: Image clustering, Uncertainty, Deep learning, Unsupervised learning, Semi-supervised learning, Constrained clustering, Siamese architecture.

1. Introduction

Clustering is a fundamental task in data analysis across diverse fields such as medicine, social science, marketing, and physics [1, 2]. By grouping similar objects based on their inherent characteristics, clustering enables a deeper understanding of complex phenomena. Over the decades, clustering algorithms have evolved significantly, employing hierarchical, density-based, and centroid-based approaches. However, no single method emerges as the best, reflecting the complexity involved in data segmentation and pattern recognition. The challenges of clustering arise from factors such as high dimensionality of features, varied data structures (shapes, sizes, densities), and the presence of noise and outliers that can distort results. Real-world data often contain overlapping clusters and incomplete information, posing significant hurdles for accurate clustering. Consequently, managing uncertainty has become essential in modern clustering techniques.

To enhance robustness and accuracy, several formalisms have been developed to represent uncertainty about cluster membership. Methods like fuzzy clustering, probabilistic clustering, and possibilistic clustering – including Fuzzy C-Means (FCM) [3], the Gustafson-Kessel algorithm [4], mixture models [5], and Possibilistic C-Means (PCM) [6] – allow data points to belong to multiple clusters through varying degrees of membership or probability assignments. Bayesian clustering provides another approach by incorporating prior knowledge and probabilistic reasoning to manage uncertainties in parameter estimation and cluster assignments. For instance, Dirichlet Process Mixtures (DPM) allow for an infinite number of potential clusters, automatically adjusting model complexity based on the data [7]. Ensemble clustering techniques like Consensus Clustering [8] enhance robustness by aggregating results from different algorithms or multiple runs of the same algorithm, leading to insights unobtainable from single methods [9, 10].

Recently, the Dempster-Shafer theory [11] has been leveraged to handle uncertainties and incomplete information in clustering by associating basic belief assignments with clusters. Since belief functions can represent both probabilities and possibilities, extensions of clustering methods to this formalism have been proposed [12, 13], such as Evidential C-Means [14], the Evidential Mixture Model [15, 16], and Evidential Hidden Markov Models [17]. Other notable methods include Belief-Peaks Evidential Clustering [18] and Model-Based Evidential Clustering [19].

In a toolbox [20] proposed by T. Denœux, many clustering methods based on belief functions have been implemented and compared on benchmark datasets. These methods include extensions of the EVCLUS algorithm [21], which performs relational clustering based on belief function theory. Improvements to the EVCLUS algorithm have incorporated new constraints, as in CEVCLUS [22], or enhanced scalability for large datasets, as in k-CEVCLUS [23]. Another extension, Neural-Network-based Evidential Clustering (NN-EVCLUS) [24], demonstrated superior performance compared to existing approaches on several datasets. NN-EVCLUS distinguishes itself by integrating pairwise dissimilarities with the flexible and powerful framework of neural networks.

Neural network-based clustering methods have gained popularity due to their ability to learn complex, non-linear representations of data. These methods include various architectures. For example, autoencoders [25] and their variational extension [26] learn compact representations that can be used in clustering and classification [27, 28]. Siamese Networks [29] form another class of neural network architectures designed to learn similarity metrics by processing two identical subnetworks with shared weights. By minimising a contrastive loss function, Siamese Networks effectively distinguish between similar and dissimilar pairs. This type of network has proved effective for tasks such as image verification, face and signature verification, where dissimilarity between two inputs is critical [30].

NN-EVCLUS bridges the gap between distance-based and neural network approaches by integrating the strengths of both methodologies. Similarly to a Siamese Network, it relies on two identical subnetworks with shared weights. However, there are two main differences: (1) the loss function is designed to enable learning without labels, and (2) it incorporates prior knowledge such as class labels, uncertain priors, or pairwise constraints (must-link and cannot-link) [23]. These features ensure rigorous theoretical foundations, enabling NN-EVCLUS to deliver valid clustering results and providing a strong basis for extending the method to image clustering.

Despite its advanced features, the original NN-EVCLUS architecture is not well-suited for processing images without preliminary hand-crafted feature extraction. Extending NN-EVCLUS to handle images would automate this process. From an image processing perspective, this extension could facilitate clustering or semi-supervised learning by encoding uncertainties in images using belief functions.

Neural network approaches make it possible to capture intricate relationships within data, making them suitable for high-dimensional and complex datasets like images. In recent years, significant advancements in image clustering have emerged due to progress in deep learning techniques. Many state-of-the-art methods leverage autoencoders for this task. One pioneering method, Deep Embedded Clustering (DEC), employs a feature extraction technique that minimises the Kullback-Leibler (KL) divergence for clustering, with initialisation facilitated by an autoencoder [31]. This method has been further enhanced by incorporating convolutional layers to better preserve local structures in images [32].

Dimensionality reduction algorithms are essential for transforming images into meaningful features or embeddings, often serving as a pre-processing step in image clustering based on neural networks [33]. Methods such as t-SNE [34] and UMAP [35] surpass Principal Component Analysis (PCA) by utilising non-linear transformations to better capture complex data structures. An innovative approach, N2D ("Not Too Deep Clustering"), integrates manifold learning techniques with autoencoders, showcasing the effectiveness of t-SNE and UMAP in image clustering tasks [36].

Recent approaches like Spectral Deep Clustering (SDC) combine spectral clustering with deep learning, achieving improved clustering results [37, 38]. Self-supervised learning techniques have also gained traction, enabling models to learn from unlabelled image datasets, as demonstrated by methods like SimCLR and BYOL [39, 40].

Contributions – These advancements highlight the potential of integrating neural networks directly with clustering tasks, particularly for image data. Inspired by these developments, we introduce DEEM (Deep Evidential Encoding of iMages) [41], a novel unsupervised clustering algorithm that integrates belief functions with distance-based learning and neural networks. The belief functions framework enables DEEM to effectively represent uncertainty in clusters and incorporate prior knowledge, whether in the form of labels on images or from constraints (must-link and cannot-link) on pairs of images during training. Coupled with a DML algorithm [42], DEEM is capable of efficiently processing large image datasets while generating distances between pairs of images.

Enhancing NN-EVCLUS, DEEM utilises convolutional neural networks (CNNs) to directly handle image data, thereby eliminating the need for explicit feature extraction. By embedding belief functions within a CNN-based clustering framework, DEEM achieves robust clustering performance, even in the presence of uncertainty and complex data structures. To our knowledge, this is the first neural network-based clustering method able to generate belief functions from images. By fully exploiting the benefits offered by customisable layers, efficient optimisation algorithms, and other state-of-the-art techniques, DEEM allows for image clustering under uncertainty and can be used as a mass function generator.

First, the extended algorithm NN-EVCLUS is presented along with belief function theory in Section 2. The methodology of our proposed approach is then detailed in Section 3, while Section 4 presents the experimental results.

2. NN-EVCLUS

In this section, we outline the operational principle of the NN-EVCLUS algorithm as described in [24]. To do so, we first recall some key concepts regarding belief functions and evidential partitions.

2.1. Theory of belief functions background

The theory of belief functions, also known as the Dempster-Shafer theory, provides a mathematical framework for representing and managing uncertainty [11]. In the context of clustering, it allows masses to be assigned to subsets of a frame of discernment, enabling the quantification of uncertainty about cluster memberships.

The frame of discernment, $\Omega = \{\omega_1, \ldots, \omega_c\}$, is a finite set containing, for example, the possible clusters for a given problem. The mass function is a mapping from the power set of Ω to $[0, 1]$:

$$\begin{array}{lll} m: & 2^\Omega & \mapsto [0,1] \\ & A & \to m^\Omega(A) \text{ such that } \sum_A m^\Omega(A) = 1. \end{array}$$

Subset A is called a focal set if $m^\Omega(A) > 0$. From masses, other functions can be computed to represent various properties [43, 44]. Of practical interest, the plausibility $Pl(A)$ measures the extent to which one fails to believe in $\overline{A}$ [45], or the part of belief that could potentially be allocated to A. It is defined as

$$Pl(A) = \sum_{B \cap A \neq \emptyset} m(B), \forall A \subseteq \Omega. \tag{1}$$

The function $pl : \Omega \mapsto [0,1]$ that maps each element $\omega \in \Omega$ to its plausibility, $pl(\omega) = Pl(\{\omega\})$, is called the *contour function* associated with m and is often used for decision-making.

The unnormalised Dempster's rule allows two independent masses to be combined conjunctively if they come from different sources:

$$(m_1 \cap m_2)(C) = \sum_{A \cap B = C} m_1(A) \cdot m_2(B), \forall C \subseteq \Omega.$$

If the intersection between two subsets A and B is empty, as with $\{\omega_1\}$ and $\{\omega_2, \omega_3\}$, for instance, a conflict arises and is quantified as follows:

$$\kappa = (m_1 \cap m_2)(\emptyset) = \sum_{A \cap B = \emptyset} m_1(A) \cdot m_2(B). \tag{2}$$

This conflict has been used in several algorithms based on belief functions, such as target association [46] or the evidential hidden Markov model [47]. In NN-EVCLUS, conflict is key as it quantifies the dissimilarity between two objects. For example, consider a clustering problem with two possible clusters, ω_1 and ω_2, and two objects, o_1 and o_2, with the evidential partition specified in Table 1.

Table 1: Example of BBA for two objects.

	$m(\emptyset)$	$m(\{\omega_1\})$	$m(\{\omega_2\})$	$m(\{\omega_1, \omega_2\})$
o_1	0.05	0.2	0.6	0.15
o_2	0.04	0.5	0.3	0.16

The contour function associated with object o_1 is

$$\begin{aligned} Pl_1(\{\omega_1\}) &= m_1(\{\omega_1\}) + m_1(\{\omega_1, \omega_2\}) = 0.2 + 0.15 = 0.35, \\ Pl_1(\{\omega_2\}) &= m_1(\{\omega_2\}) + m_1(\{\omega_1, \omega_2\}) = 0.6 + 0.15 = 0.75. \end{aligned}$$

Thus, this object can be assigned to cluster ω_2 in this example.

The dissimilarity between the two objects can be quantified using Equation (2), which represents the conflict κ_{12}:

$$\begin{aligned} \kappa_{12} &= m_1(\emptyset) + m_2(\emptyset) + m_1(\{\omega_1\}) \cdot m_2(\{\omega_2\}) + m_1(\{\omega_2\}) \cdot m_2(\{\omega_1\}) \\ \kappa_{12} &= 0.05 + 0.04 + 0.2 \cdot 0.3 + 0.6 \cdot 0.5 \\ \kappa_{12} &= 0.45, \end{aligned}$$

which is the complement of $Pl_{12}(\Omega)$, representing the plausibility that objects o_1 and o_2 belong to the same class, as $\kappa_{12} = 1 - Pl_{12}(\Omega)$.

2.2. Evidential clustering

Following the formalism of T. Denœux [24], consider a set of n objects, such as images, denoted as $\mathcal{O} = \{o_1, \ldots, o_n\}$. Each object is assumed to belong to at most one cluster in the set $\Omega = \{\omega_1, \ldots, \omega_c\}$, and the partial knowledge about the cluster membership of an object o_i is represented by a mass function m_i on Ω. The n-tuple $M = (m_1, \ldots, m_n)$ is referred to as an *evidential (or credal) partition* of $\mathcal{O}$ and encompasses a broad range of clustering structures.

Evidential clustering entails grouping objects (such as images) to form an evidential partition that satisfies an optimality criterion. Within the framework of belief functions, the foundational work of Denœux and Masson introduced an algorithm known as EVCLUS [21]. EVCLUS assigns belief masses to clusters using a dissimilarity matrix $D = (\delta_{ij})$, where δ_{ij} quantifies the dissimilarity between objects o_i and o_j without the requirement to satisfy strict distance properties.

The core principle of EVCLUS is that *similar objects should have higher plausibility of belonging to the same cluster*. This is accomplished by minimising the degree of conflict κ_{ij} between their mass functions, with plausibility defined as $pl_{ij} = 1 - \kappa_{ij}$. Consequently, the evidential partition $M = (m_1, \ldots, m_n)$ is optimised to align the degrees of conflict with the dissimilarities through a monotonic transformation, drawing parallels with MDS [48].

Since the introduction of EVCLUS, several related algorithms have been developed, as detailed in [24] and [49]. Examples include CEVCLUS [22], which incorporates pairwise constraints, and a scalable version designed for large datasets [50, 23].

2.3. NN-EVCLUS for feature vector clustering

NN-EVCLUS is a neural network-based algorithm specifically designed for evidential clustering. It learns to map input feature vectors to mass functions defined over a frame Ω of clusters. Similarly to EVCLUS, the algorithm ensures that similar inputs are associated with mass functions exhibiting lower levels of conflict.

The algorithm operates on two primary inputs: a set of feature vectors and a dissimilarity matrix. If additional information, such as pairwise constraints or labelled data, is available, it can be incorporated into specific loss functions to refine the clustering process.

The neural network used in NN-EVCLUS has a multilayer architecture made of fully connected layers and ReLU activation functions (although other functions could be used), combined with a custom loss function. This loss function *minimises the discrepancy between dissimilarities and the degrees of conflict for object pairs.* A key distinction from EVCLUS is that NN-EVCLUS encodes data representations in its connection weights, enabling the model to generalise to new data and produce evidential partitions without requiring retraining. The computation of quantities such as conflict has been formalised in matrix form, allowing for highly efficient implementation. *While the generalisation of the network to new data was not clearly demonstrated in the original work, this is one of the primary focuses of the present study.*

Similar to a Siamese network, training involves processing pairs of inputs. A forward pass is applied to each element of a pair using the same network (shared weights). Backpropagation is performed after computing the conflict between the two generated mass functions. In the original implementation, details on gradient computation were provided to manually perform backpropagation. In this work, automatic differentiation will be utilised.

The network parameters, θ, are obtained by minimising the following loss function:

$$\mathcal{L}_{\mathrm{D}} = \sum_{i=1}^{N} \sum_{j>i} \left(\delta_{ij} - \kappa_{ij}(\theta)\right)^2 . \tag{3}$$

This function depends on two key quantities: the conflict $\kappa_{ij}(\theta)$ (defined in Equation (2)) between the masses m_i and m_j predicted for inputs x_i and x_j (both in $\Re^d$), and the dissimilarity δ_{ij} between these feature vectors.

Various types of distance measures can be used, such as Euclidean distance or cosine distance applied to feature vectors. Since the conflict is bounded within $[0, 1]$, the distance d_{ij} must be rescaled to this range and interpreted as a dissimilarity measure δ_{ij}. A commonly used transformation is:

$$\delta_{ij} = 1 - \exp\left(-\gamma d_{ij}\right),$$

where the scaling factor γ is defined as proposed in [24]:

$$\gamma = -\frac{\log(0.05)}{d_0},$$

with d_0 being a normalising factor set in NN-EVCLUS as:

$$d_0 = \text{Percentile}\,90\left(d_{ij} \mid (i, j) \text{ are pairs of data points}\right),$$

which represents the value below which 90% of the distances between all considered pairs of data points fall.

The objective of minimising the cost function is to find the masses that produce a conflict matching the known distance between two points. Therefore, the method for computing distances must be chosen carefully to align with the expected output classes. This consideration becomes especially critical when dealing with high-dimensional data, such as images.

3. DEEM for image clustering

In this section, we present the DEEM algorithm. We begin by describing the advantages and general principles of this algorithm compared to NN-EVCLUS. Then, we emphasise the importance of dissimilarity generation, a key element for successful convergence.

3.1. Advantages of DEEM

DEEM is an extension of the NN-EVCLUS algorithm that handles image inputs. After training, a basic belief assignment can be generated from a given image. The main features of this extension include:

- DEEM provides access to the entire realm of deep learning, leveraging its advancements. It fully exploits the benefits offered by customisable layers, sophisticated optimisation algorithms, and other state-of-the-art techniques.

- The network weights are optimised using automatic differentiation (AD) [51], enabling users to configure the network according to specific application needs.

- The scalability of the implementation, using GPU and minibatches, is demonstrated on the MNIST dataset ($60k$ images for training and $10k$ for testing generalisation). This is achieved with a self-designed network detailed below, as well as with a ResNet18 model [52] ($11m$ parameters), demonstrating transfer learning capabilities on deep networks.

- Our implementation includes the option to incorporate priors on labels or constraints on pairs, similar to NN-EVCLUS.

We particularly focus on efficiently computing distances. Since images are high-dimensional, direct computation of pairwise distances is often ineffective. To address this, we propose several procedures, including a highly efficient approach based on DML.

3.2. DEEM algorithm

The learning process in DEEM is illustrated in Figure 1. The network requires two inputs: image data and pairwise dissimilarities. Dissimilarities can either be pre-computed and stored or computed online using a DML approach. For simplicity, a specific section is dedicated to this latter method.

For unsupervised tasks, straightforward pairwise distance calculations or other image-specific dissimilarity measures can be employed.

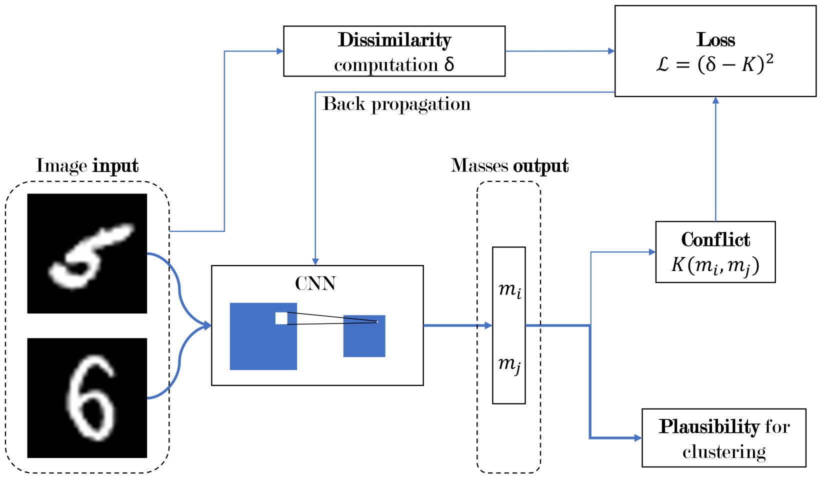

Dissimilarity computation δ
Loss $\mathcal{L} = (\delta - K)^2$
Back propagation
Image input
Masses output
CNN
m_i
m_j
Conflict $K(m_i, m_j)$
Plausibility for clustering

Figure 1: Unsupervised learning process of the DEEM method.

During training, a minibatch is sampled (e.g., 128 pairs of images). Images in a pair (x_i, x_j) are presented sequentially to the network. Each image passes through the network layers, which can be configured by the user. The network outputs a basic belief assignment m_i and m_j for each input image. The focal sets considered include the empty set, singletons, full ignorance, and pairs.

For each pair of masses corresponding to a pair of images, the conflict $\kappa_{ij}(\theta)$ is computed using the current network parameters. This conflict is then incorporated into the loss function (Eq. (3)) with the corresponding dissimilarity δ_{ij}:

$$\mathcal{L}_{ij} = (\delta_{ij} - \kappa_{ij}(\theta))^2.$$

If the true class of an image is known ($y_{ik} = 1$ for class ω_k for i-th image), an additional term can be added to the loss:

$$\mathcal{L}_i^{\mathrm{LAB}} = \sum_{\omega_k} (pl_{ik} - y_{ik})^2, \tag{4}$$

where $y_{ik} = I(y_i = \omega_k)$, $pl_{ik} = pl_i(\omega_k)$ with pl_i the contour function corresponding to the mass function m_i generated by the network for image i.

If prior information about the pair is available (e.g., whether the images are similar or dissimilar), specific loss terms can be defined, as initially proposed in CEVCLUS [53] and NN-EVCLUS [24]:

$$\mathcal{P}_{\mathrm{ML}} = \sum_{(i,j)\in \mathrm{ML}} \left(Pl_{ij}(\overline{S}_{ij}) + 1 - Pl_{ij}(S_{ij}) \right),$$

$$\mathcal{P}_{\mathrm{CL}} = \sum_{(i,j)\in \mathrm{CL}} \left(Pl_{ij}(S_{ij}) + 1 - Pl_{ij}(\overline{S}_{ij}) \right),$$

where ML and CL denote must-link (same classes) and cannot-link (different classes) constraints. Expressions for $Pl_{ij}(\overline{S}_{ij})$ and $Pl_{ij}(S_{ij})$ are derived by Antoine et al. [53]:

$$Pl_{ij}(S_{ij}) = 1 - \kappa_{ij}(\theta),$$

and

$$Pl_{ij}(\overline{S}_{ij}) = 1 - m_i(\emptyset) - m_j(\emptyset) + m_i(\emptyset)m_j(\emptyset) - \sum_{k=1}^{K} m_i(\{\omega_k\})m_j(\{\omega_k\}),$$

where K represents the number of subsets retained among the 2^Ω possible subsets. The composite loss is then defined as:

$$\mathcal{L}_{\mathrm{C}} = \frac{1}{|\mathrm{ML}|}\mathcal{P}_{\mathrm{ML}} + \frac{1}{|\mathrm{CL}|}\mathcal{P}_{\mathrm{CL}}, \tag{5}$$

where $|\mathrm{ML}|$ and $|\mathrm{CL}|$ denote the number of must-link and cannot-link constraints, respectively.

The final loss function combines these terms:

$$\mathcal{L}_{\mathrm{NET}} = \mathcal{L}_{\mathrm{D}} + \lambda \mathcal{L}_{\mathrm{LAB}} + \xi \mathcal{L}_{\mathrm{C}}, \tag{6}$$

where λ and ξ are hyperparameters that control the influence of supervised terms. Setting these coefficients to zero results in fully unsupervised training. Regularisation terms on the parameters can also be included (typically implemented in Python). Adjusting these hyperparameters enables the model to handle incomplete or partially supervised scenarios, which adapt to real-world cases.

The network parameters are optimised using automatic differentiation. Various optimisers, such as Adam [54], Stochastic Gradient Descent with Momentum (SGDM), or RMSProp, can be employed. Learning rate schedulers are also available to adapt the training process dynamically.

3.3. How to generate relevant pairwise distances?

In the DEEM algorithm, the cost function relies on dissimilarities computed from the input data, aiming to align the predicted conflicts with the actual distances. The efficiency of training is strongly influenced by the quality of the dissimilarity matrix D_{ij}. Poorly chosen distances can hinder convergence, highlighting the importance of finding a matrix that closely approximates the known labels. Both unsupervised and supervised approaches for determining dissimilarities are discussed below.

3.3.1. Unsupervised approaches for determining dissimilarities

Figure 2 illustrates three different approaches. The first one consists in vectorising images and applying a pairwise distance. This is the simplest and less effective approach since a pixel-wise comparison between two images often fails to capture geometric subtleties and is sensitive to variations like rotation, translation, and lighting.

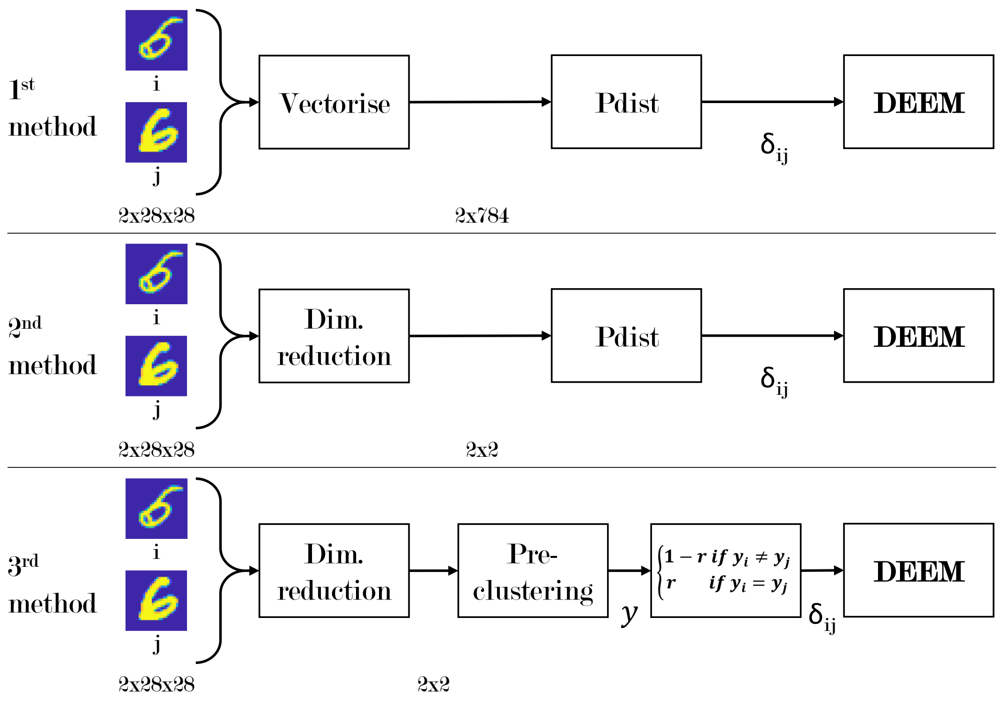


Figure 2: Presentation of three unsupervised methods for D_{ij} computation. Digit images from MNIST are used for illustration. “pdist” stands for pairwise distance.

The second approach relies on applying a dimensionality reduction method on the vectorised form, and then applying a pairwise distance. It partly overcomes the previous challenges because images are represented using a reduced set of features with some statistical properties. The images can also be pre-processed using for example Structural Similarity Index Measure (SSIM) [55], Scale-Invariant Feature Transform (SIFT) [56], Speeded-Up Robust Features (SURF) [57], and Oriented FAST and Rotated BRIEF (ORB) [58]. Dimensionality reduction techniques include Principal Component Analysis (PCA), t-Distributed Stochastic Neighbor Embedding (t-SNE) [34], and Uniform Manifold Approximation and Projection (UMAP) [35]. While PCA captures linear relationships, t-SNE and UMAP excel at capturing non-linear structures, making them effective for complex data. In the following tests, PCA, t-SNE, and UMAP are compared.

The third approach is mentioned as pre-clustering in the figure. It relies on the idea that a clustering method can be applied after dimensionality reduction in order to generate pseudo-labels. These labels can then be used to compute the dissimilarity matrix. Common methods include K-means, Gaussian Mixture Models (GMM), and hierarchical clustering which are compared in the experiment section. The dissimilarity matrix is generated by comparing pseudo-labels as follows:

$$\delta_{ij} = \begin{cases} 1 - r & \text{if } y_i \neq y_j, \\ r & \text{if } y_i = y_j, \end{cases} \tag{7}$$

where y_i represents the pseudo-label of image i, and r is a random noise value drawn from a uniform distribution in $[0, 1]$.

3.3.2. Supervised approaches for determining dissimilarities

A simple way to incorporate supervision in D_{ij} computation is to apply Equation (7) to real labels, identifying must-link (ML) and cannot-link (CL) cases. The loss function is designed to minimise conflict κ_{ij} in ML cases and approach 1 in CL cases.

Another approach involves training an intermediate algorithm to generate dissimilarities, thereby improving generalisability and reducing the reliance on labels. This can be achieved using a *Distance Metric Learning (DML)* procedure with triplet loss [59]. In the context of triplet-based learning, an anchor (an image in our context) is a reference input from the dataset. It serves as the basis for comparing two other inputs: the positive and the negative. The positive input is a sample that is similar to the anchor. Specifically for this work, it belongs to the same class. The negative input, in contrast, is a sample that differs significantly from the anchor, from a different class in our case. The goal of the triplet loss function is to learn an embedding function f such that the distance between the anchor (a) and positive (p) embeddings is minimised, while the distance between the anchor and negative (n) embeddings is maximised, with a margin (α) ensuring sufficient separation:

$$\mathcal{L}_t(a, p, n) = \max(0, ||f(a) - f(p)||^2 - ||f(a) - f(n)||^2 + \alpha),$$

By training on anchor-positive-negative triplets, the model learns an embedding space suitable for computing dissimilarities (likewise to the "code" generated by an autoencoder), which can then be used as input to the DEEM algorithm. The pairwise distance is therefore applied on the output of the

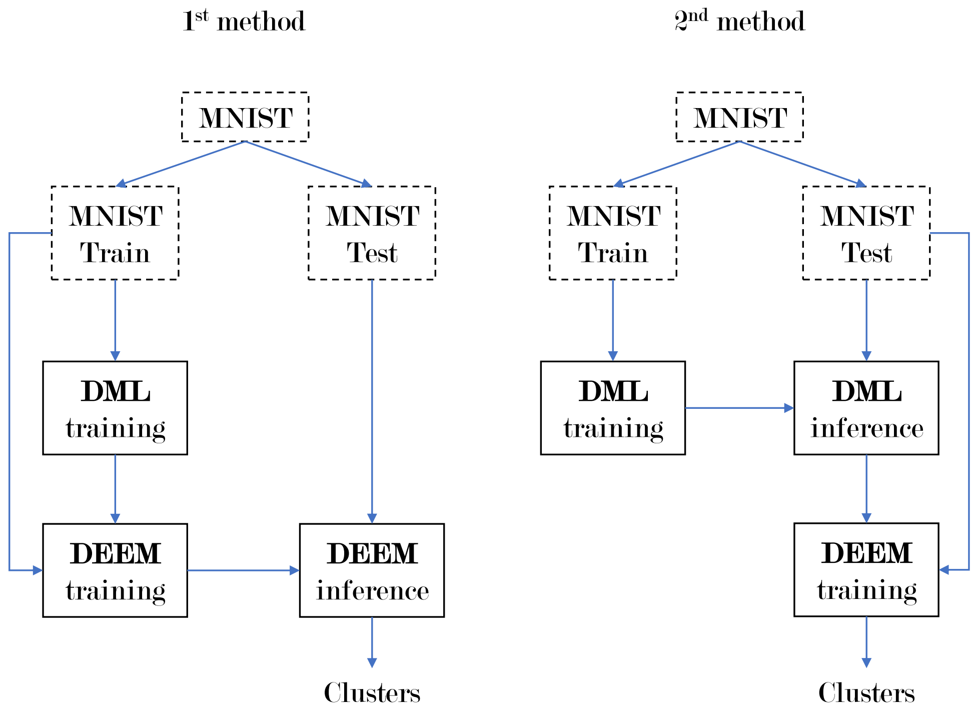


Figure 3: Two different usages of the DML model.

DML model. A small amount of prior is generally sufficient to generate very efficient DML models.

Figure 3 illustrates two methods for utilising DEEM with a DML model. In both approaches, DML is trained on a given training dataset to generate embeddings by inference. The distinction lies in how DEEM is used:

- First method: DEEM is trained on the outputs of DML derived from the training set. During testing, DEEM infers clusters directly from images of the testing set, enabling an evaluation of generalisation capabilities. DML is not used on the testing set.
- Second Method: The DML model generates embeddings for the testing set, which are then used to compute pairwise distances. DEEM is subsequently trained on the testing set in an unsupervised manner. While this may seem questionable, it is essential to note that no labels or supervised information are used during this process. The test set is only used for parameter updates under purely unsupervised conditions,

with the assumption that the test data is available in its entirety or in sufficiently large batches to perform minibatch updates.

Figure 4 summarises the entire process of coupling DML and DEEM using the first method, with two approaches for integrating prior knowledge. Positive and negative pairs are assigned based on anchor classes, with an adjustable percentage of labelled data. This percentage is varied in subsequent tests to evaluate performance under different amounts of prior information, considering either class labels or pairwise similarity.

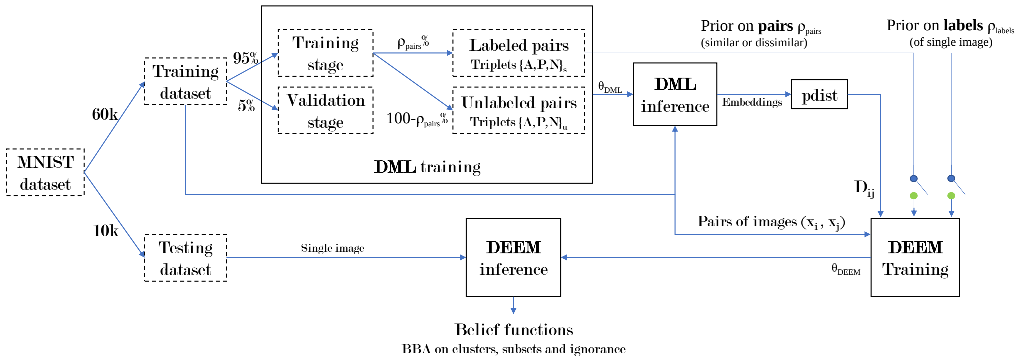

MNIST dataset
60k
10k
Training dataset
95%
5%
Testing dataset
Training stage
Validation stage
ρ_pairs %
100-ρ_pairs %
Labeled pairs Triplets {A,P,N}_s
Unlabeled pairs Triplets {A,P,N}_u
DML training
θ_DML
DML inference
Embeddings
pdist
Prior on pairs ρ_pairs (similar or dissimilar)
Prior on labels ρ_labels (of single image)
D_ij
Pairs of images (x_i, x_j)
DEEM Training
θ_DEEM
Single image
DEEM inference
Belief functions
BBA on clusters, subsets and ignorance

Figure 4: Schematic explaining how DEEM and DML are coupled and where prior knowledge plays a role.

4. Experiments

In this section, three main aspects emerge regarding the application of DEEM. The first part is exploratory, aiming to justify the relevance of the method. Here, we observe the behaviour of the mass partition to extract useful insights. Furthermore, exploiting masses on pairs – a distinctive feature of the method – allows for a significant performance improvement.

The second part focuses on the utilisation of the algorithm in an unsupervised mode. Within this, we develop an optimisation approach for the algorithm, relying on pre-existing methods to compute dissimilarities, and a benchmark is proposed across various datasets.

Lastly, the third part presents the use of the algorithm in supervised mode. In this final section, we address several ways to provide the algorithm with side information and analyse the impact of the amount of prior knowledge used.

Before delving into these three aspects, we begin with a configuration section to establish the necessary foundations for the discussion.

4.1. Experimental settings

Performance computation – To monitor the network's performance in real time, the comparison between the predicted clusters and the actual classes was computed at each iteration based on Adjusted Rand Index (ARI), Accuracy (ACC), and Normalised Mutual Information (NMI) criteria. These three criteria are defined in $[0, 1]$, where 1 corresponds to the best achievable performance. For improved readability, the performance metrics in the following experiments are presented as [ACC/NMI/ARI]. Unless specified in some specific tests, training was stopped after 7,000 iterations. The model retained corresponds to the one achieving the highest performance on the evaluation set. The test set performance is then computed.

Main dataset – All experiments were conducted using the MNIST dataset[1]. MNIST is a handwritten digit dataset of $60k$ training images and $10k$ testing images. Additionally, to perform a benchmark on other datasets, we also tested our algorithm in Section 4.3.3 on USPS[2], a digit image dataset, and COIL-20 (processed)[3], an image dataset of 20 classes of everyday-life objects. Table 2 summarises the characteristics of these datasets.

Table 2: Dataset characteristics.

	MNIST	**USPS**	**COIL-20**
Type	Digit images	Digit images	Object images
Number of classes	10	10	20
Image size	28x28	16x16	128x128
Size of testing	$10k$	2,007	-
Size of training	$60k$	7,291	-
Total images	$70k$	9,298	1,440

General configuration – Our method was implemented in MATLAB R2024a for all experiments, except for the test with the DML in Section 4.4.3

[1]Different shuffles might exist, so we specify we used *mnist_784* from *fetch_openml*, found under `https://www.openml.org/d/554`.

[2]Available on `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets`.

[3]Available on `https://cave.cs.columbia.edu/repository/COIL-20`.

or when using ResNet18 [52] where a Python version was developed. A two-layer convolutional network (CNN) was used as shown in Figure 5. For 10 classes (like MNIST), this configuration results in a network with 57 outputs and 8.7k learnable parameters. DEEM also requires hyperparameters, summarised in Table 3.

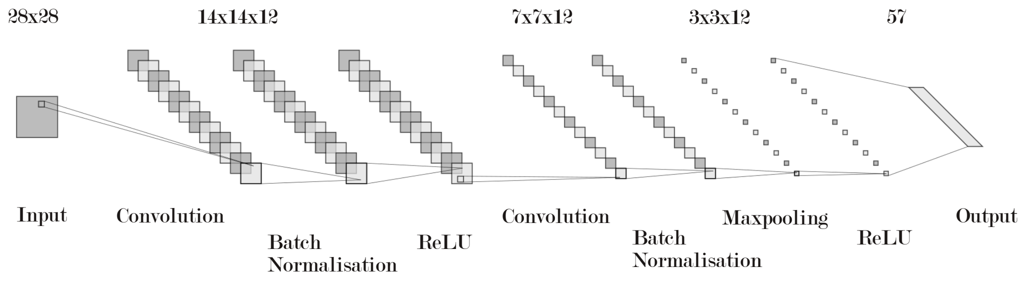


Figure 5: Neural network layers configuration for the tests carried with DEEM (image made thanks to https://alexlenail.me/NN-SVG).

Table 3: DEEM hyperparameters summary.

Minibatch size	128
Weight initialisation	He
Optimiser	Adam
Learning rate	0.01
Supervision coefficients λ and ξ	0.4

Specific configurations – During the unsupervised usage of DEEM, we used two methods for similarity calculation: t-SNE from MATLAB and UMAP 0.5.5 from Python. Both methods were used with default setups to avoid any supervision by selecting their best configuration for a given dataset. Default parameters are summarised in Appendix B, Table B.15 for t-SNE and Table B.16 for UMAP.

Concerning DML, the CNN was composed of two convolutional layers: the first one with 64 filters and the second one with 128 filters, both using 3×3 kernels with stride 1 and padding 1. Each convolutional layer was followed by a ReLU activation and a max-pooling operation with a 2×2 kernel. The resulting feature maps are flattened and passed through two fully connected layers, with 256 and 128 neurons, respectively. This configuration enables the network to process input images and produce embeddings in a 128-dimensional feature space, suitable for downstream tasks such as clustering or classification. This DML model was trained with $ADAM$, a learning rate of 0.001, a margin $\alpha = 8$, a minibatch size of 256 and stopped after 20 epochs, where one epoch corresponds to pass all single images from the training set.

4.2. Preliminary results

The advantage of DEEM method is that it generates masses from images. In this section, we illustrate on MNIST dataset the interest of representing uncertainty using belief functions.

For that, we first observed the evolution of the evidential partition containing all mass vectors. Presented in video format[4], from which Figure 6 is extracted, this evolution shows a random distribution of masses in the first iteration of the network, converging towards an ordered assignment of the classes. Here, each column of pixels in the figure represents the mass vector output by the network. The masses are ordered *a posteriori* by the true classes for the sake of interpretation. We can observe that they converge towards singletons. As a reminder, the organisation of the masses is carried out similarly to Table 1. In the last iteration, we observe that class 5 (corresponding to digit 4) has not been assigned to a singleton but the classification remains effective using the plausibility contour function. It can be explained by Equation (1) and observing that the mass is mainly distributed on subsets containing class 4, in particular ω_{04}, ω_{14}, ω_{34}, ω_{46}, ω_{47}, and ω_{48} (where $\omega_{kl} = \{\omega_k, \omega_l\}$). By observing the adjacent classes, we notice a similar result, for example, for digit 6, the mass on ω_{26} is the largest, and for digit 7, the mass is distributed on ω_{17} and ω_{67}. Therefore, the network, once trained, seems to hesitate between classes prone to doubt, such as 1 and 7. The complexity of the architecture ($8.7k$ learnable parameters) can make it sensitive to other parameters that we, as humans, do not necessarily distinguish. Therefore, the doubt between 2 and 6, and between 6 and 7, seems more difficult to interpret.

Uncertainty management by belief functions in deep learning can boost the network's performance [24, 49]. As an illustration, we used the same algorithm architecture in two situations: first by considering pairs of classes, and second without pairs. Two $10k$ MNIST datasets drawn randomly were

[4]`https://drive.google.com/file/d/1UoEB4tMhQPdm2JCccV3BvYoRuGn6u7S3`

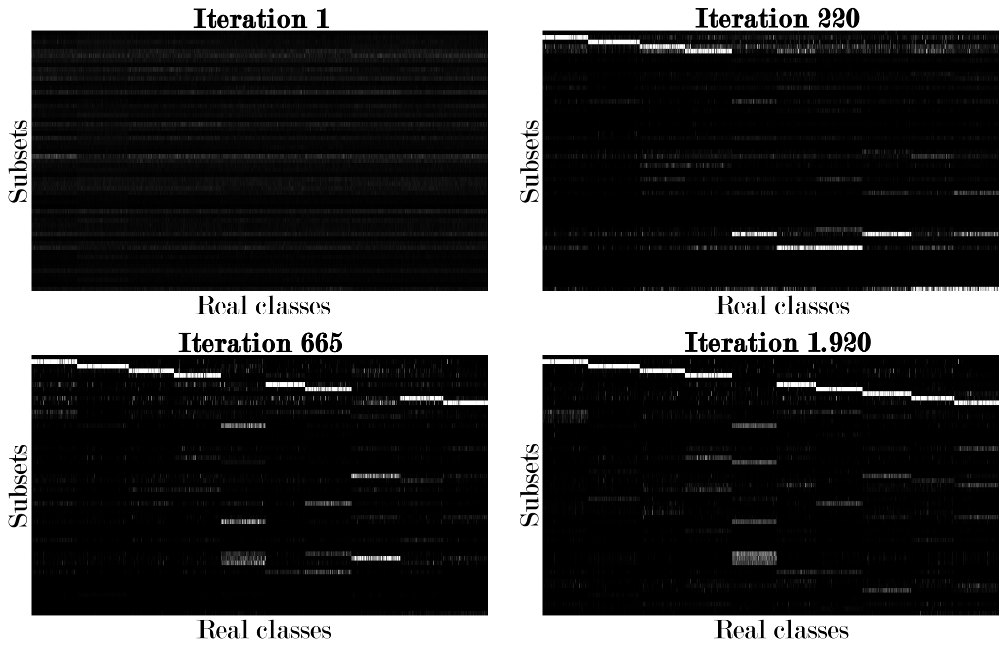


Figure 6: Evolution of masses subsets for the clustering of MNIST dataset (57 subsets and $1k$ images ordered in 10 classes).

tested for this purpose, with one case containing all images from three classes and another case with 10 classes. The results reported in Table 4 show, in a fully unsupervised setting, a clear improvement of the performance when doubt between digits is considered.

Table 4: Performance with and without doubt (expressed by pairs) on digits. Examples with 3 classes (digits 5, 6 & 8) and 10 classes from MNIST dataset.

	3 classes		**10 classes**	
	with pairs	**w/o pairs**	**with pairs**	**w/o pairs**
ACC	0.98	0.89	0.90	0.81
NMI	0.89	0.70	0.93	0.87
ARI	0.93	0.71	0.88	0.77

The objective of the algorithm is not to reproduce the dissimilarity matrix identically. Minimising the cost function reduces the gap between conflict and distances, but it is still crucial that the network avoids overfitting and remains able to effectively classify similar data with an incorrectly calculated distance. To illustrate this point, we generated two distinct datasets and intentionally introduce a percentage of errors in the dissimilarity matrix, which in this case is directly determined by the labels. Two classes from the MNIST dataset were used, namely the 1s and 6s, which exhibit notable dissimilarity. The labels were subsequently altered for 1, 10, 100, and 500 points to create a dissimilarity matrix with errors. This matrix was generated by imposing random numbers close to zero for identical classes and numbers close to one for different classes. Figure 7 shows the two clusters with the alteration of 500 labels. The results of the study, summarised in Table 5, demonstrate the network's ability to ignore the errors in the matrix and converge to an error-free score.

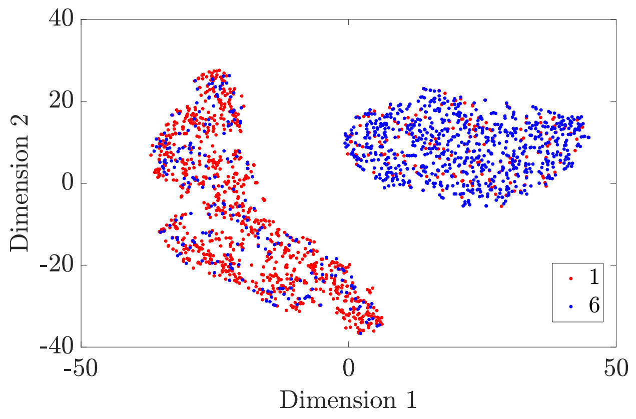


Figure 7: t-SNE representation of clusters with 500 label modifications for perfect 1 and 6 digit clusters.

Table 5: Results in convergence for two classes, 1 and 6 digits, with different rate of errors in the dissimilarity matrix.

Number of affected labels	0	1	10	100	500
Corresponding ratio	0%	0.05%	0.5%	5%	25%
1st iteration to 0 errors	38	385	609	895	N/A
Final number of errors	0	0	0	0	1

4.3. Unsupervised case

As previously explained, our algorithm operates in both unsupervised and supervised modes. In this section, we focus on the former. Particular attention must be paid to the definition of dissimilarities, as they are crucial for successful convergence. Poorly defined dissimilarities tend to destabilise the training process, leading to suboptimal performance. While we have demonstrated earlier that DEEM can mitigate errors in dissimilarities, this capability diminishes as the complexity of the data (for example the image content or the number of classes) increases. For this reason, we explore strategies in this section to optimise the performance of unsupervised clustering by carefully selecting the method for calculating dissimilarities.

4.3.1. Pairwise dissimilarity

On distances – To evaluate the effectiveness of distance calculation methods, we measure the overlap area A between the within-class (H_1) and between-class (H_2) histograms:

$$A = \frac{\sum \min(H_1, H_2)}{\sum H_1}.$$

As shown in Table 6, the overlap between the distances for dissimilar classes and those for identical classes is significant. This high overlap explains the challenges that DEEM faces in achieving effective convergence. The table also includes results after applying dimensionality reduction techniques. These methods improve the spatial representation of the data, resulting in better separation between clusters.

Table 6: Histogram overlapping measure for several distances.

	Raw images	**UMAP**	**t-SNE**
Euclidean	92%	37%	62%
Squared Euclidean	92%	37%	62%
Cityblock	92%	39%	64%
Minkowski	92%	37%	62%
Chebychev	99%	37%	63%
Cosine	88%	92%	83%
Correlation	88%	100%	100%
Hamming	94%	100%	100%
Jaccard	100%	100%	100%
Spearman	87%	100%	100%

For comparison, Table 7 summarises the results of both the K-Means and the NN-EVCLUS algorithms applied to the MNIST dataset. The third part of the table concerns DEEM (using images as input and embeddings for distances). A random batch of $5k$ images was selected[5]. The input data was processed using several methods. The first one involves vectorising the images before submitting them to the clustering algorithms. Other methods apply transformations directly to the images, employing PCA, UMAP, and t-SNE (Section 3.3.1) to generate a new representation with a limited number of dimensions (2 or 3 in this study).

Table 7: Performance of K-means, NN-EVCLUS and DEEM in the unsupervised mode using $5k$ images. For K-means and NN-EVCLUS, the inputs were feature vectors computed using different methods. For DEEM, embeddings from UMAP and t-SNE were used to compute distances while handling images as inputs.

	K-means			**NN-EVCLUS**			**DEEM**		
	ACC	**NMI**	**ARI**	**ACC**	**NMI**	**ARI**	**ACC**	**NMI**	**ARI**
Vectorised	0.47	0.49	0.31	0.47	0.43	0.30	-	-	-
PCA	0.50	0.50	0.29	0.48	0.42	0.30	-	-	-
UMAP 2D	0.82	0.85	0.75	0.40	0.63	0.35	0.73	0.79	0.65
UMAP 3D	0.81	0.85	0.75	0.48	0.73	0.41	0.70	0.78	0.63
t-SNE 2D	0.84	0.81	0.75	0.61	0.66	0.51	0.70	0.70	0.73
t-SNE 3D	0.85	0.82	0.75	0.57	0.68	0.51	0.81	0.78	0.71

The results highlight the challenges of processing MNIST images directly in vectorised form, even after applying PCA. However, dimensionality reduction techniques like UMAP and t-SNE appear to facilitate the grouping of similar data. Notably, the errors produced by the K-Means and NN-EVCLUS clustering algorithms are more attributable to their final predictions than to the grouping achieved by the reduction methods. For instance, both clustering methods tend to split clusters into subgroups, resulting in erroneous classifications. Two dimensions seem sufficient for the algorithms to achieve their best results. While t-SNE delivers slightly better results than UMAP with NN-EVCLUS, the overlap with t-SNE was twice as high as that obtained with UMAP.

DEEM outperforms NN-EVCLUS in all cases, highlighting the advantages of extending the approach to images. However, K-means achieves better overall performance on this reduced dataset ($5k$ images). This can be largely

[5] Available at `https://github.com/emmanuelramasso/DEEM_Deep_Evidential_Clustering_Images/blob/main/Matlab/data/MNIST_5k_data.mat`.

attributed to the method used to compute pairwise distances. As previously discussed, constructing the dissimilarity matrix based on these distances may not be optimal, as overlap in the data tends to penalise DEEM. The next section investigates alternative methods for constructing the dissimilarity matrix to enhance its performance. We also explore the generalisation capabilities of the methods.

On pre-clustering and generalisation capabilities – Trained neural networks possess adaptability properties that can surpass conventional clustering methods in certain scenarios. In our case, data reduced by UMAP and t-SNE already exhibit significant potential for effective labelling by conventional algorithms [60]. Given the limitations of distance-based methods, we propose using an initial unsupervised clustering step to derive a dissimilarity matrix. Standard clustering algorithms such as K-Means, Gaussian Mixture Models (GMM), and hierarchical clustering are applied to t-SNE and UMAP dimensionality reductions. The influence of the size of the dataset is studied considering $5k$ and $70k$ MNIST images. Table 8 summarises the performances of each clustering method on the different reductions.

The best clustering performance was achieved using hierarchical clustering on UMAP for the $5k$ image subset. However, this method demonstrates poorer performance when applied to the largest dataset, highlighting its limitations in scalability.

Table 8: Conventional clustering performances on dimensionality reduced MNIST with UMAP and t-SNE for $5k$ and $70k$ images.

		K-means			**GMM**			**Hierarchical**		
		ACC	**NMI**	**ARI**	**ACC**	**NMI**	**ARI**	**ACC**	**NMI**	**ARI**
UMAP	**5*k***	0.82	0.85	0.78	0.83	0.86	0.78	0.97	0.93	0.94
	70*k*	0.85	0.85	0.78	0.85	0.88	0.82	0.82	0.89	0.81
t-SNE	**5*k***	0.84	0.81	0.75	0.77	0.66	0.75	0.86	0.82	0.78
	70*k*	0.92	0.85	0.83	0.68	0.72	0.56	0.89	0.84	0.79

Using dissimilarities generated from labels provided by hierarchical clustering on $5k$ images, DEEM achieves comparable performance to the original clustering: [0.97/0.93/0.94]. When the trained model is applied to the full dataset ($70k$ MNIST images), DEEM generalises effectively, reaching [0.96/0.89/0.91]. In contrast, the hierarchical clustering method exhibits a significant loss of performance on the largest dataset, achieving only [0.82/0.89/0.81] despite being "trained" on the full dataset.

Furthermore, the generalisation from $5k$ to $70k$ images highlights a notable drop in performance for the hierarchical clustering method, as shown in Table 9. In contrast, DEEM appears to leverage the prior information generated by hierarchical clustering efficiently. By building relevant features through its CNN layers, DEEM demonstrates superior generalisation capabilities.

Table 9: Generalisation on $70k$ MNIST images after training on $5k$ images.

	ACC	**NMI**	**ARI**
K-means	0.65	0.63	0.48
GMM	0.49	0.53	0.36
Hierarchical	-	-	-
DEEM	0.96	0.89	0.91

4.3.2. Performance of unsupervised methods

The methodology previously described is applied to the MNIST dataset in this section. Table 10 summarises the obtained performances. In the first lines, dissimilarities was computed from pairwise distances (pdist) using $5k$ randomly drawn images from MNIST leading to poor performance ([0.48/0.43/0.32]). The next three rows in the table highlight the best performances of the K-Means, NN-EVCLUS, and DEEM methods when using dissimilarities derived from t-SNE dimensionality reduction. Among them, DEEM demonstrated the highest performance, reaching [0.70/0.70/0.73].

The subsequent tests involve pre-clustering after dimensionality reduction. Using hierarchical pre-clustering, DEEM achieved [0.97/0.93/0.94]. While DEEM does not surpass the performance of pre-clustering alone, it remains highly effective in terms of generalisation as shown in the final rows which illustrate the performance of DEEM on the full MNIST dataset, both with and without generalisation. DEEM achieved [0.96/0.89/0.91] on the entire $70k$ batch, outperforming other methods that showed significant declines in effectiveness for the full dataset.

Table 10: Results for different clustering methods over DML embedding on MNIST dataset.

Generalisation	**Train**	**Test**	**Dij calculation**	**Evaluated algorithm**	**ACC**	**NMI**	**ARI**
No	$5k$	$5k$	Distance	DEEM	0.48	0.43	0.32
No	$5k$	$5k$	t-SNE → Distance	K-Means	0.79	0.79	0.72
No	$5k$	$5k$	t-SNE → Distance	NN-EVCLUS	0.61	0.66	0.51
No	$5k$	$5k$	t-SNE → Distance	DEEM	0.70	0.70	0.73
No	$5k$	$5k$	t-SNE → Clustering	K-means	0.84	0.81	0.75
No	$5k$	$5k$	t-SNE → Clustering	GMM	0.77	0.66	0.75
No	$5k$	$5k$	t-SNE → Clustering	Hierarchical	0.97	0.93	0.94
No	$5k$	$5k$	t-SNE → Clustering (Hierarchical)	DEEM	0.89	0.82	0.78
No	$5k$	$5k$	UMAP → Clustering (Hierarchical)	DEEM	0.97	0.93	0.94
No	$70k$	$70k$	t-SNE → Clustering	K-means	0.92	0.85	0.83
No	$70k$	$70k$	t-SNE → Clustering	GMM	0.68	0.72	0.56
No	$70k$	$70k$	t-SNE → Clustering	Hierarchical	0.89	0.84	0.79
Yes	$5k$	$70k$	t-SNE → Clustering	K-means	0.65	0.63	0.48
Yes	$5k$	$70k$	t-SNE → Clustering	GMM	0.49	0.53	0.36
Yes	$5k$	$70k$	t-SNE → Clustering (Hierarchical)	DEEM	0.96	0.89	0.91

4.3.3. Benchmark

To evaluate the performance of our algorithm on different datasets, we applied clustering on USPS (a digit dataset) and COIL-20 (a collection of everyday objects). Table 11 summarises the results of our method in comparison to other approaches reported in the literature for these datasets.

In both cases, DEEM demonstrates high performance. For the datasets concerning digits, the algorithm's performance is constrained by a few classes that are particularly challenging to classify. In the COIL-20 dataset, three specific classes could be considered as subclasses since they correspond to car-toy images that are visually similar to one another. This close resemblance poses a significant challenge for our algorithm, which remains an area for future improvement.

Table 11: Benchmark comparison of DEEM with existing methods on image clustering tasks using USPS and Coil-20 datasets.

	MNIST		**USPS**		**Coil-20**	
	ACC	**NMI**	**ACC**	**NMI**	**ACC**	**NMI**
DEC [31]	0.863	0.834	0.762	0.767	0.712	0.836
SCAE-KMS [61]	0.823	0.738	0.681	0.660	0.723	0.787
FCAE-KMS [62]	0.794	0.698	0.667	0.645	0.843	0.882
DBC [62]	0.964	0.917	0.743	0.724	0.793	0.895
DEEM	0.957	0.890	**0.969**	**0.924**	**0.847**	**0.898**

4.4. Supervised case

Incorporating side information about the true classes of the inputs introduces an element of supervision into the training process. In this context, the generalisation performance of the network is consistently evaluated using a test dataset that remains unseen during training. To enhance generalisation, it is crucial to carefully consider both the type of side information provided and the way it is utilised during the training process.

This section on supervised learning is organised into four parts. First, we compare four methods for leveraging prior knowledge. Next, we focus on DML, a method designed to generate accurate embeddings for dissimilarity computation. Then, we examine the impact of using varying percentages of prior knowledge. Finally, we analyse the robustness of DML when confronted with different levels of prior knowledge.

4.4.1. Four supervision methods

We explore the application of supervision in our algorithm through various approaches to evaluate generalisability. Specifically, we present results for four methods that make use of prior knowledge:

- by incorporating it into the loss function in a "conventional" manner (Eq. (4));
- by embedding it in the dissimilarity matrix referred to as "D_{ij} prior";
- by leveraging ML (must-link) and CL (cannot-link) constraints within the loss function (Eq. (5));
- or by applying DML beforehand for dissimilarity computation.

About conventional prior – Conventional prior knowledge can be integrated as an additional term in the loss function, which compares the predicted class to the true class. When a difference occurs, the loss function is penalised by this additional term. As expressed in Eq. (6), the learning process is influenced by the coefficient λ. If λ is set too low, training will progress slowly and primarily depend on $\mathcal{L}_{\text{D}}$ rather than $\mathcal{L}_{\text{LAB}}$. Conversely, if λ is set too high, it may lead to overfitting. The default value considered in 4.1 results in [0.98/0.94/0.95].

On Perfect dissimilarities – In an ideal scenario, distances can be directly derived from the true class labels: a distance of 0 is assigned if two images belong to the same class, and a distance of 1 if they belong to different classes. Supervised dissimilarities leverage this prior information by incorporating it into $\mathcal{L}_{\text{D}}$ through δij, as shown in Eq. (3). This approach introduces must-link and cannot-link constraints between pairs of images, differing from traditional supervised learning where class labels are directly utilised in a specific loss function ($\mathcal{L}_{\text{LAB}}$, Eq. (4)).

To evaluate the robustness of DEEM with supervised dissimilarities, increasing levels of randomness were introduced into the perfect distance matrix. Figure 8 displays the ARI scores across 10 trials, where noise is progressively added to the dissimilarities derived from the true classes. The dissimilarities stored in the matrix D_{ij} shift from being perfect to a distribution with increasing width, $\Delta_{D_{ij}}$.

Perfect distances, defined as 0 and 1 (left-hand side), show limited results due to numerical issues arising from gradient calculations. Starting from the

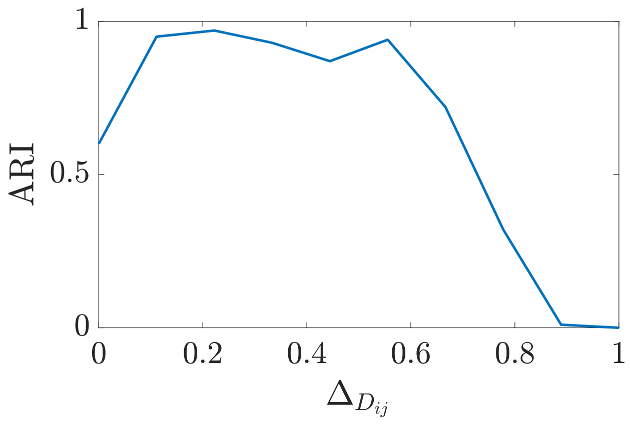


Figure 8: ARI performance for dissimilarity variations.

7th level of noise, ARI scores begin to decline as cluster separability diminishes, driven by increasing overlap between the distributions of distances for pairs of similar and dissimilar images. As shown in Figure 9, at the 7th noise level, the overlap constitutes approximately one-third of the total distribution. Despite this, the ARI score remains relatively high (0.7), demonstrating the algorithm's robustness in handling such noise.

Finally, the best result achieved by DEEM is [0.97/0.92/0.93], which is slightly below the performance of the first method.

On ML and CL constraints – Instead of applying constraints in the dissimilarity matrix, Antoine et al. [53] proposed to add an additional term in the loss function, $\mathcal{L}_{\mathrm{C}}$ from Eq. (5). With 100% of constraints, DEEM reached [0.59/0.76/0.62].

On DML – The final method for integrating prior knowledge into training is DML. It learns to generate dissimilarities using labelled data. With DML, DEEM achieves its best results: [0.994/0.980/0.986].

Overall comparison – In summary, Table 12 presents a comparison of the fully supervised learning methods and their generalisation performances. Among these, DML achieves the highest performance, followed by conventional supervision and dissimilarities derived directly from labels. In contrast, supervision based on must-link (ML) and cannot-link (CL) constraints shows the weakest performance.

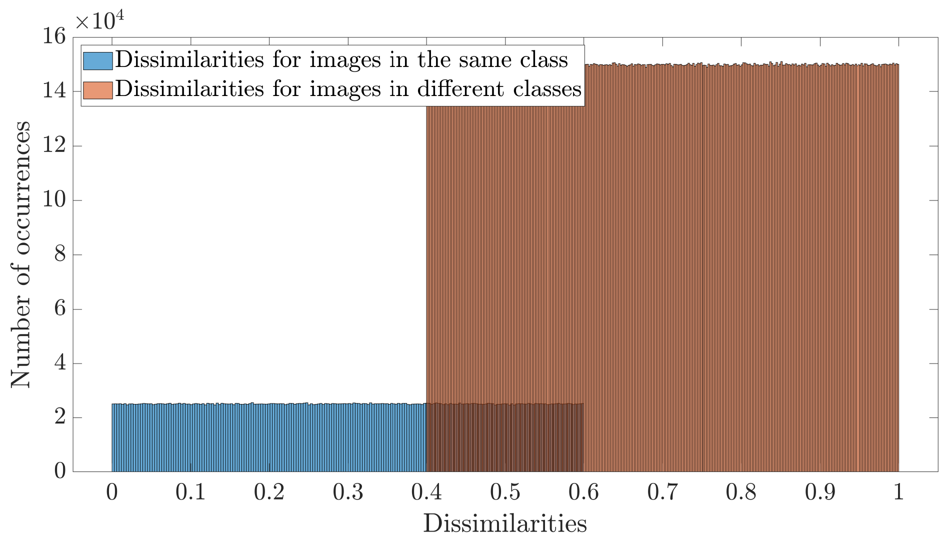


Figure 9: Distance histograms of MNIST classes with $\Delta_{D_{ij}} = 0.6$.

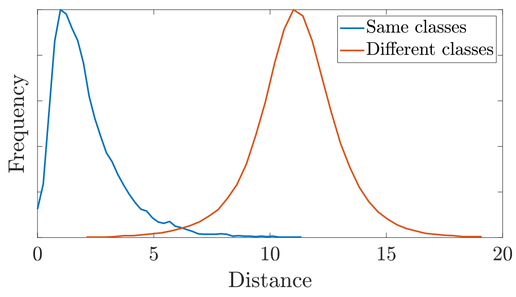
Table 12: Results for supervised clustering on MNIST

Train	**Test**	**Method**	**ACC**	**NMI**	**ARI**
$5k$	$60k$	Dij from labels	0.97	0.91	0.93
$5k$	$60k$	Supervised Loss	0.98	0.94	0.95
$5k$	$60k$	ML and CL	0.59	0.76	0.62
$48k$	$12k$	DML $\rightarrow$ K-means	**0.994**	**0.980**	**0.986**

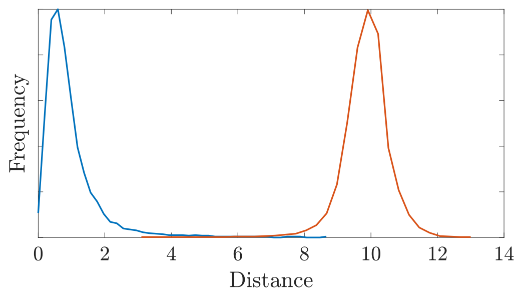
4.4.2. A focus on the DML method

DML was utilised to generate dissimilarities prior to clustering with DEEM. By incorporating 50% of the prior knowledge from the $60k$ MNIST training images, DML facilitated the generation of embeddings for any new input. These embeddings enabled the computation of distances between images based on learned features, as shown in Figure 10. The figure illustrates the separation between the distributions of within-class and between-class pairwise distances computed from the DML embeddings. From the initial iteration (left-hand side) to the final iteration (right-hand side), the separation between the between-class and within-class distributions improved by a factor of 10.

Table 13 (lines 1-2) highlights that K-means achieved excellent performance on the DML-generated embeddings, demonstrating the effectiveness



(a)



(b)

Figure 10: Histogram spreading between distances from same and different clusters on MNIST dataset: (a) 1st iteration and (b) 1,900th iteration of DML. Distributions were normalised independently for better readability.

of DML in producing meaningful representations. Applying t-SNE for dimensionality reduction (from 128 dimensions) prior to K-means or DEEM clustering (lines 2, 4 and 6) resulted in very close performances with a slight decrease. These results underscore the strength of DML in creating high-quality embeddings. Both DEEM and K-means delivered comparable performance in various configurations, competitive with the state-of-the-art N2D method [36].

Table 13: Results for supervised clustering on MNIST

Train	**Test**	**Method**	**Algo.**	**ACC**	**NMI**	**ARI**
$48k$	$12k$	DML	K-means	**0.994**	**0.980**	**0.986**
$48k$	$12k$	DML $\rightarrow$ t-SNE	K-means	0.993	0.980	0.985
$48k$	$12k$	DML	DEEM	0.992	0.975	0.982
$48k$	$12k$	DML $\rightarrow$ t-SNE	DEEM	0.985	0.961	0.967
$48k$	$12k$	DML $\rightarrow$ K-means	DEEM	**0.994**	**0.980**	**0.986**
$48k$	$12k$	DML $\rightarrow$ t-SNE $\rightarrow$ K-means	DEEM	0.993	0.980	0.985

4.4.3. Variation of the amount of prior information

One of the key advantages of our method is its ability to incorporate prior knowledge into the learning process without relying on the complete set of true class labels. In this section, we evaluate the performance of different configurations based on the amount of prior knowledge. Specifically, we consider the first three cases from the previous section for incorporating prior knowledge: "conventional" (Eq. (4)), dissimilarity matrix (referred to as "D_{ij} prior"), and leveraging ML and CL constraints (Eq. (5), referred to as "Constraints"). The DML case is addressed in the next section.

For each configuration, labelled images were randomly sampled from the training set based on a probability value in the set $\{0.005, 0.025, 0.05, 0.01, 0.1, 0.2, 0.3, \ldots, 0.9\}$. *The original training/testing MNIST datasets were reversed to simulate limited data availability and training on a smaller subset of data.* A total of $5k$ images were randomly drawn from the original testing set (the same images as in previous tests) and used for training, while the full training set of $60k$ images was utilised for performance evaluation using ARI, NMI, and ACC.

Dissimilarities were computed from distances obtained after dimensionality reduction with t-SNE. Although t-SNE underperforms compared to UMAP, it was deliberately chosen to establish a clear baseline, thereby highlighting the added value of incorporating prior knowledge. Notably, DML was not used in this study.

The results, shown in Figure 11, indicate that conventional supervision with the loss $\mathcal{L}_{\text{LAB}}$ consistently achieves the highest performance across all three evaluation metrics, regardless of the amount of prior knowledge. For the smallest amount of prior (0.005), the performance reached approximately [0.719/0.651/0.576], while for the largest amount of prior (0.9), it improved to [0.953/0.882/0.899]. Among the three methods compared, conventional supervision outperforms the others, particularly when the proportion of labelled data exceeds 10%. This can be attributed to the fact that when a single dissimilarity δij is supervised or constrained, conventional supervision effectively supervises two images for the same amount of prior knowledge, thereby doubling the weight of the loss.

However, the overall performance is lower compared to previous results achieved with DML. For example, as shown in Table 12, with 50% prior information, a performance drop of approximately 20% is observed.

One potential explanation for this discrepancy is that DEEM relies, in this case, on a relatively simple CNN (Figure 5). When DML is used, it leverages a second CNN, which better facilitates the separation of clusters. To validate this hypothesis, the next study explores a significantly more complex network architecture, ResNet18.

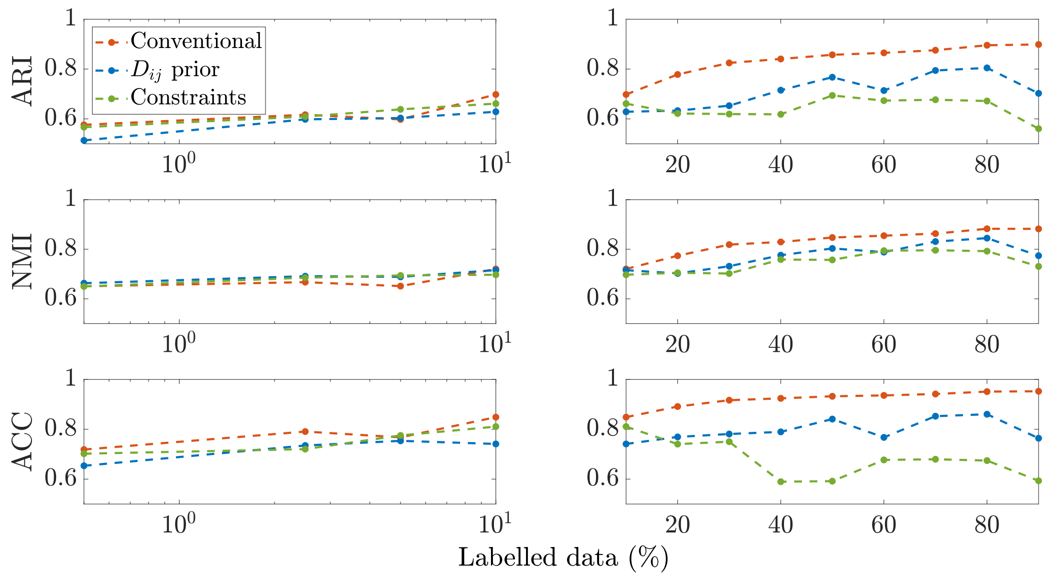


Figure 11: Performances of DEEM for different amount of prior on MNIST dataset using three supervision modes referred as Conventional (use of labels in $\mathcal{L}_{\text{LAB}}$), D_{ij} prior (δ_{ij} modified as 0 or 1 in $\mathcal{L}_{\text{D}}$) or Constraints (use of ML and CL constraints in $\mathcal{L}_{\text{C}}$). DEEM was trained on $5k$ images and tested on $60k$ images, without DML.

4.4.4. On using deep networks with transfer learning

A *ResNet18* model [52], consisting of approximately 11 million parameters and pre-trained on the ImageNet database, was used as the backbone for DEEM to explore the potential of deep transfer learning [49]. The same hyperparameters as those used in previous experiments with the custom network (initially consisting of approximately 8,700 parameters) were applied. The training set ($60k$ images) and testing set ($10k$ images) remained consistent with the original MNIST dataset.

Given the significant time required for convergence, which varies with the amount of prior knowledge, early stopping was implemented based on the gradient norm of the model parameters. Specifically, training was halted if the gradient norm fell below 0.05 for five consecutive iterations or if the number of iterations exceeded $500k$ with a learning rate set to 0.0001. Performance was evaluated based on the maximum and average ARI values on the test set ($10k$ images) over the last 30 iterations. As in the previous section, the amount of prior knowledge was varied across the set $\{0.005, 0.025, 0.05, 0.01, 0.1, 0.2, 0.3, \ldots, 0.9\}$.

Labels were randomly selected and used with DML to create true positive and true negative pairs. For each minibatch, an anchor image was drawn. If the image was labelled, a positive sample was selected from the same class, while a negative sample was drawn from a different class. For unlabelled images, the positive sample was the same as the anchor, while the negative sample was selected randomly. These labels were preserved to ensure consistent use of the same labelled images in DEEM.

In DEEM, these labels were employed in two distinct ways. The first approach, referred to as "constraints" used conventional supervision by directly applying class labels. The second approach, termed "D_{ij} prior" leveraged Must-Link (ML) and Cannot-Link (CL) constraints to determine whether two images belonged to the same or different classes.

For comparison, K-means clustering was applied to the DML-generated embeddings from the training data, with performance evaluated on the test set to assess generalisation. Additionally, COP-K-means was used to integrate pairwise constraints into K-means [63]. Our implementation diverged slightly from the original paper: we first initialised the algorithm using K-means++ without constraints. Then, during iterations, points that violated the constraints were reassigned to the nearest cluster in accordance with the standard K-means procedure.

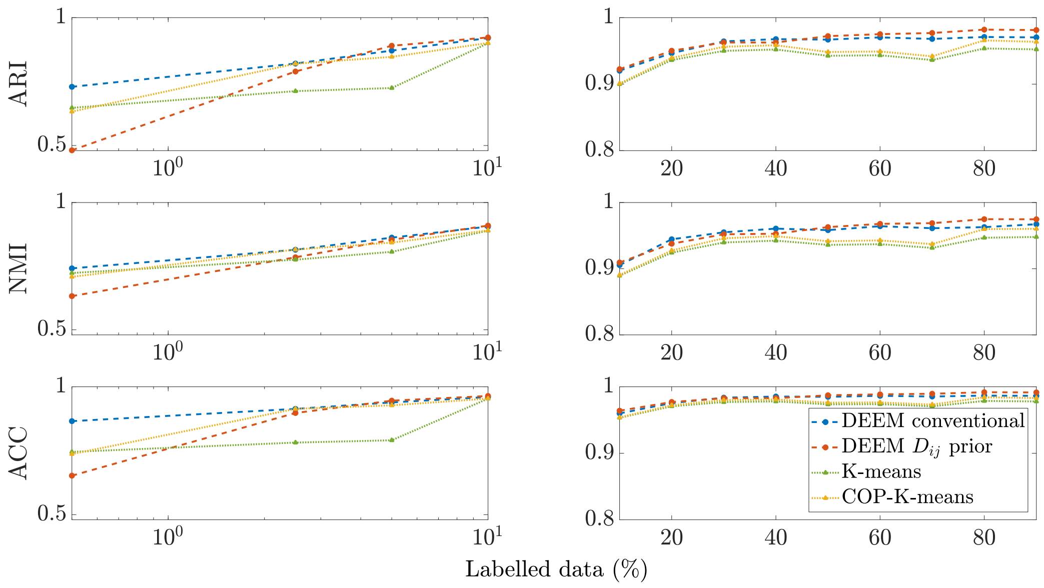


Figure 12: Variation of performances of DEEM, K-means and COP-K-means for different amount of prior (with DML) on MNIST dataset.

Figure 12 illustrates the results for the "constraints" and "D_{ij} prior" configurations, as well as for K-means with and without constraints. As expected, performance improved with an increasing proportion of labelled data. Notably, the "D_{ij} prior" configuration was slightly better than the "constraints" one when more than 10% of the training data was labelled (corresponding to $6k$ images in this example).

This result differs from the test made with the simpler network where the "constraints" configuration was better. Pairwise constraints can be easier to satisfy as they require only information about whether two images belong to the same or different classes, rather than relying on explicit class labels.

Table 14 presents the results obtained using ResNet18 with different types and amounts of prior knowledge. For comparison, we have also included some results from previous sections that were achieved with the simpler CNN. This table shows that complex configurations, including ResNet and DML, lead to better generalisation with a large margin.

Table 14: Performance of transfer learning from ResNet18.

Arch.	**Prior qty**	**DML**	**Prior type**	**ARI**	**NMI**	**ACC**
Simple	5%	No	ML & CL	0.812	0.697	0.661
Simple	5%	No	Conventional	0.849	0.720	0.698
ResNet18	5%	No	ML & CL	0.692	0.820	0.734
ResNet18	5%	No	Conventional	0.722	0.842	0.762
ResNet18	5%	Yes	ML & CL	**0.964**	**0.909**	**0.923**
ResNet18	5%	Yes	Conventional	0.960	0.906	0.920
Simple	50%	No	ML & CL	0.592	0.757	0.694
Simple	50%	No	Conventional	0.932	0.847	0.857
ResNet18	50%	No	ML & CL	0.733	0.773	0.815
ResNet18	50%	No	Conventional	0.838	0.922	0.842
ResNet18	50%	Yes	ML & CL	**0.987**	**0.963**	**0.972**
ResNet18	50%	Yes	Conventional	0.985	0.958	0.967
Simple	90%	No	ML & CL	0.593	0.730	0.561
Simple	90%	No	Conventional	0.953	0.882	0.899
ResNet18	90%	No	ML & CL	0.772	0.783	0.868
ResNet18	90%	No	Conventional	0.868	0.938	0.867
ResNet18	90%	Yes	ML & CL	**0.992**	**0.975**	**0.981**
ResNet18	90%	Yes	Conventional	0.986	0.967	0.970

In conclusion, the results demonstrate that in scenarios with limited supervision, DEEM is particularly effective and relevant using transfer learning based on ResNet, DML and pairwise constraints.

5. Conclusion

DEEM is a novel approach to image clustering through the integration of belief functions and convolutional neural networks. This combination enables efficient image processing and robust performance, particularly in the presence of uncertainty. By encoding uncertainty through belief functions, DEEM significantly enhances clustering tasks, making it well-suited for handling complex image data.

Our experiments, particularly on the MNIST dataset, demonstrate the effectiveness of DEEM, even in unsupervised settings, due to its innovative dissimilarity processing. The algorithm's flexibility to incorporate prior knowledge makes it a powerful tool for both supervised and unsupervised learning scenarios. Furthermore, the robustness of DEEM is evident in its ability to mitigate errors in the dissimilarity matrix, highlighting the importance of this component in achieving optimal performance.

The incorporation of Distance Metric Learning (DML) further strengthens DEEM's capabilities, particularly when working with partially labelled datasets. This feature is especially advantageous for real-world applications where labelled data is often scarce. The algorithm's adaptability to integrate various forms of prior knowledge, such as must-link and cannot-link constraints, underscores its versatility and potential for broader applications.

Looking ahead, DEEM holds promise for applications beyond image clustering, such as semi-supervised learning for acoustic emission scalograms or spectrograms. Future research could focus on refining the network architecture to reduce reliance on dissimilarity matrices and exploring alternative methods for incorporating soft supervision to enhance generalisation across diverse datasets.

In conclusion, DEEM offers a robust and versatile solution for clustering under uncertainty, particularly for image data. Its ability to handle large datasets, manage uncertainty, and integrate prior knowledge positions it as a valuable tool in domains where data labelling is limited or uncertain, paving the way for new advancements in clustering and semi-supervised learning.

Acknowledgement

This work has been supported by the EIPHI Graduate school (contract ANR-17-EURE-0002) and the Region Bourgogne-Franche-Comté, with the financial support of SAFRAN CERAMICS and Ministère des Armées – Agence de l'innovation de défense (AID).

Datasets and code availability

Datasets and codes are available at `https://github.com/emmanuelramasso/DEEM_Deep_Evidential_Clustering_Images`.

References

[1] C. X. Gao, D. Dwyer, Y. Zhu, C. L. Smith, L. Du, K. M. Filia, J. Bayer, J. M. Menssink, T. Wang, C. Bergmeir, S. Wood, S. M. Cotton, An overview of clustering methods with guidelines for application in mental health research, Psychiatry Research 327 (2023) 115265. `doi:10.1016/j.psychres.2023.115265`.

[2] A. Decelle, B. Seoane, L. Rosset, Unsupervised hierarchical clustering using the learning dynamics of restricted boltzmann machines, Phys. Rev. E 108 (2023) 014110. `doi:10.1103/PhysRevE.108.014110`.

[3] J. C. Bezdek, Pattern recognition with fuzzy objective function algorithms, Springer Science & Business Media, 2013. `doi:10.1007/978-1-4757-0450-1`.

[4] D. E. Gustafson, W. C. Kessel, Fuzzy clustering with a fuzzy covariance matrix, in: 1978 IEEE Conference on Decision and Control including the 17th Symposium on Adaptive Processes, 1978, pp. 761–766. `doi:10.1109/CDC.1978.268028`.

[5] G. J. McLachlan, S. X. Lee, S. I. Rathnayake, Finite mixture models, Annual review of statistics and its application 6 (1) (2019) 355–378. `doi:10.1146/annurev-statistics-031017-100325`.

[6] R. Krishnapuram, J. M. Keller, The possibilistic c-means algorithm: insights and recommendations, IEEE transactions on Fuzzy Systems 4 (3) (1996) 385–393.

[7] Y. Li, E. Schofield, M. Gönen, A tutorial on dirichlet process mixture modeling, J. Math. Phsychol. 91 (2019) 128–144. doi:10.1016/j.jmp.2019.04.004.

[8] A. L. Fred, A. K. Jain, Combining multiple clusterings using evidence accumulation, IEEE Trans. Pattern Anal. Mach. Intell. 27 (6) (2005) 835–850. doi:10.1109/TPAMI.2005.113.

[9] E. Ramasso, V. Placet, M. L. Boubakar, Unsupervised consensus clustering of acoustic emission time-series for robust damage sequence estimation in composites, IEEE Transactions on Instrumentation and Measurement 64 (12) (2015) 3297–3307. doi:10.1109/TIM.2015.2450354.

[10] J. R. Regatti, A. A. Deshmukh, E. Manavoglu, U. Dogan, Consensus clustering with unsupervised representation learning, in: 2021 International Joint Conference on Neural Networks (IJCNN), 2021, pp. 1–9. doi:10.1109/IJCNN52387.2021.9533714.

[11] G. Shafer, A mathematical theory of evidence, Vol. 42, Princeton university press, 1976. doi:10.2307/j.ctv10vm1qb.

[12] Z. Zhang, Y. Zhang, H. Tian, A. Martin, Z. Liu, W. Ding, A survey of evidential clustering: Definitions, methods, and applications, Information Fusion (2024) 102736doi:https://doi.org/10.1016/j.inffus.2024.102736.

[13] T. Denœux, O. Kanjanatarakul, Evidential clustering: A review, in: V.-N. Huynh, M. Inuiguchi, B. Le, B. N. Le, T. Denœux (Eds.), Integrated Uncertainty in Knowledge Modelling and Decision Making, Springer International Publishing, Cham, 2016, pp. 24–35. doi:10.1007/978-3-319-49046-5_3.

[14] M.-H. Masson, T. Denœux, ECM: an evidential version of the fuzzy c-means algorithm, Pattern Recognition 41 (4) (2008) 1384–1397. doi:10.1016/j.patcog.2007.08.014.

[15] P. Vannoorenberghe, P. Smets, Partially supervised learning by a credal EM approach, in: Symbolic and Quantitative Approaches to Reasoning with Uncertainty: 8th European Conference, ECSQARU 2005, Barcelona, Spain, July 6-8, 2005. Proceedings 8, Springer, 2005, pp. 956–967. doi:10.1007/11518655_80.

[16] L. Jiao, T. Denœux, Z.-G. Liu, Q. Pan, EGMM: an evidential version of the gaussian mixture model for clustering, Applied Soft Computing 129 (2022) 109619. doi:10.1016/j.asoc.2022.109619.

[17] E. Ramasso, Inference and learning in evidential discrete latent markov models, IEEE Transactions on Fuzzy Systems 25 (5) (2016) 1102–1114. doi:10.1109/TFUZZ.2016.2598361.

[18] Z.-G. Su, T. Denœux, BPEC: belief-peaks evidential clustering, IEEE Transactions on Fuzzy Systems 27 (1) (2019) 111–123. doi:10.1109/TFUZZ.2018.2869125.

[19] T. Denœux, Calibrated model-based evidential clustering using bootstrapping, Information Sciences 528 (2020) 17–45. doi:10.1016/j.ins.2020.04.014.

[20] T. Denœux, evclust: An R Package for Evidential Clustering, Université de technologie de Compiègne, available at https://cran.r-project.org/web/packages/evclust/index.html (2023).

[21] T. Denœux, M.-H. Masson, EVCLUS: evidential clustering of proximity data, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 34 (1) (2004) 95–109.

[22] V. Antoine, B. Quost, M.-H. Masson, T. Denœux, CEVCLUS: evidential clustering with instance-level constraints for relational data, Soft Computing 18 (2014) 1321–1335.

[23] F. Li, S. Li, T. Denœux, k-CEVCLUS: Constrained evidential clustering of large dissimilarity data, Knowledge-Based Systems 142 (2018) 29–44. doi:10.1016/j.knosys.2017.11.023.

[24] T. Denœux, NN-EVCLUS: Neural network-based evidential clustering, Information Sciences 572 (2021) 297–330. doi:10.1016/j.ins.2021.05.011.

[25] G. E. Hinton, R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, in: Proceedings of the 2006 IEEE Symposium on Foundations of Computer Science (FOCS'06), IEEE, 2006, pp. 363–369. doi:10.1126/science.1127647.

[26] D. P. Kingma, M. Welling, Auto-encoding variational bayes, in: International Conference on Learning Representations (ICLR), Banff, Canada, 2014. doi:10.48550/arXiv.1312.6114.

[27] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, A. Lerchner, beta-vae: Learning basic visual concepts with a constrained variational framework, in: International Conference on Learning Representations, 2022.

[28] D. P. Kingma, M. Welling, An introduction to variational autoencoders, Foundations and Trends in Machine Learning 12 (4) (2019) 307–392. doi:10.1561/2200000056.

[29] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, R. Shah, Signature verification using a "siamese" time delay neural network, Advances in neural information processing systems 6 (1993). doi:10.1142/S0218001493000339.

[30] F. Schroff, D. Kalenichenko, J. Philbin, Facenet: A unified embedding for face recognition and clustering, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 815–823. doi:10.48550/arXiv.1503.03832.

[31] J. Xie, R. Girshick, A. Farhadi, Unsupervised deep embedding for clustering analysis, in: International conference on machine learning, PMLR, 2016, pp. 478–487. doi:10.48550/arXiv.1511.06335.

[32] X. Guo, X. Liu, E. Zhu, J. Yin, Deep clustering with convolutional autoencoders, in: Neural Information Processing: 24th International Conference, ICONIP 2017, Guangzhou, China, November 14-18, 2017, Proceedings, Part II 24, Springer, 2017, pp. 373–382. doi:10.1007/978-3-319-70096-0_39.

[33] Y. Ren, J. Pu, Z. Yang, J. Xu, G. Li, X. Pu, S. Y. Philip, L. He, Deep clustering: A comprehensive survey, IEEE Transactions on Neural Networks and Learning Systems (2024) 1–21.

[34] L. Van der Maaten, G. Hinton, Visualizing data using t-sne., Journal of machine learning research 9 (11) (2008).

[35] L. McInnes, J. Healy, J. Melville, Umap: Uniform manifold approximation and projection for dimension reduction, arXiv preprint arXiv:1802.03426 (2018).

[36] R. McConville, R. Santos-Rodriguez, R. J. Piechocki, I. Craddock, N2d:(not too) deep clustering via clustering the local manifold of an autoencoded embedding, in: 2020 25th international conference on pattern recognition (ICPR), IEEE, 2021, pp. 5145–5152. `doi:10.1109/ICPR48806.2021.9413131`.

[37] X. Yang, C. Deng, F. Zheng, J. Yan, W. Liu, Deep spectral clustering using dual autoencoder network, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 4066–4075. `doi:10.1109/CVPR.2019.00419`.

[38] S. Affeldt, L. Labiod, M. Nadif, Spectral clustering via ensemble deep autoencoder learning (SC-EDAE), Pattern Recognition 108 (2020) 107522. `doi:10.1016/j.patcog.2020.107522`.

[39] T. Chen, S. Kornblith, M. Norouzi, G. Hinton, A simple framework for contrastive learning of visual representations, in: Proceedings of the 37th International Conference on Machine Learning, ICML'20, JMLR.org, 2020, pp. 1597–1607. `doi:10.48550/arXiv.2002.05709`.

[40] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, et al., Bootstrap your own latent-a new approach to self-supervised learning, Advances in neural information processing systems 33 (2020) 21271–21284. `doi:10.48550/arXiv.2006.07733`.

[41] L. Guiziou, E. Ramasso, S. Thibaud, S. Denneulin, Deep evidential clustering of images, in: International Conference on Belief Functions, Springer, 2024, pp. 3–12. `doi:10.1007/978-3-031-67977-3\_1`.

[42] J. L. Suárez, S. García, F. Herrera, A tutorial on distance metric learning: Mathematical foundations, algorithms, experimental analysis, prospects and challenges, Neurocomputing 425 (2021) 300–322. `doi:10.1016/j.neucom.2020.08.017`.

[43] K. Sentz, S. Ferson, Combination of evidence in dempster-shafer theory, Tech. Rep. SAND2002-0835, 800792, Sandia National Lab. (SNL-NM & SNL-CA) and US Department of Energy (2002). doi:10.2172/800792.

[44] P. Smets, R. Kennes, The transferable belief model, Artificial Intelligence 66 (2) (1994) 191–234. doi:10.1016/0004-3702(94)90026-4.

[45] F. Pichon, Belief functions: canonical decompositions and combination rules, Ph.D. thesis, University of Technology of Compiègne, Compiègne, France (March 2009).

[46] A. Ayoun, P. Smets, Data association in multi-target detection using the transferable belief model, International Journal of Intelligent Systems 16 (10) (2001) 1167–1182. doi:10.1002/int.1054.

[47] E. Ramasso, Contribution of belief functions to hidden markov models with an application to fault diagnosis, in: 2009 IEEE International Workshop on Machine Learning for Signal Processing, IEEE, 2009, pp. 1–6. doi:10.1109/MLSP.2009.5306209.

[48] I. Borg, P. Groenen, I. Borg, P. Groenen, The four purposes of multidimensional scaling, Modern Multidimensional Scaling: Theory and Applications (1997) 3–14.

[49] L. Jiao, F. Wang, Z.-G. Liu, Q. Pan, TDEC: evidential clustering based on transfer learning and deep autoencoder, IEEE Trans. Fuzzy Syst. 32 (10) (2024) 5585–5597. doi:https://doi.org/10.1109/TFUZZ.2024.3421564.

[50] T. Denœux, S. Sriboonchitta, O. Kanjanatarakul, Evidential clustering of large dissimilarity data, Knowledge-Based Systems 106 (2016) 179–195.

[51] M. Bartholomew-Biggs, S. Brown, B. Christianson, L. Dixon, Automatic differentiation of algorithms, Journal of Computational and Applied Mathematics 124 (1) (2000) 171–190. doi:10.1016/S0377-0427(00)00422-2.

[52] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and

pattern recognition, 2016, pp. 770–778. doi:10.48550/arXiv.1512.03385.

[53] V. Antoine, B. Quost, M.-H. Masson, T. Denœux, CECM: constrained evidential c-means algorithm, Computational Statistics & Data Analysis 56 (4) (2012) 894–914. doi:10.1016/j.csda.2010.09.021.

[54] D. Kingma, J. Ba, Adam: A method for stochastic optimization, in: International Conference on Learning Representations (ICLR), San Diega, CA, USA, 2015. doi:10.48550/arXiv.1412.6980.

[55] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, Image quality assessment: From error visibility to structural similarity, IEEE Transactions on Image Processing 13 (4) (2004) 600–612. doi:10.1109/TIP.2003.819861.

[56] D. G. Lowe, Object recognition from local scale-invariant features, in: Proceedings of the seventh IEEE international conference on computer vision, Vol. 2, Ieee, 1999, pp. 1150–1157.

[57] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-up robust features (surf), Computer vision and image understanding 110 (3) (2008) 346–359.

[58] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, Orb: An efficient alternative to sift or surf, in: 2011 International conference on computer vision, Ieee, 2011, pp. 2564–2571.

[59] A. Hermans, L. Beyer, B. Leibe, In defense of the triplet loss for person re-identification, arXiv preprint arXiv:1703.07737 (2017).

[60] M. Allaoui, Considerably improving clustering algorithms using umap dimensionality reduction technique: A comparative study., in: Image and Signal Processing: 9th International Conference, ICISP, 2020, pp. 317–325. doi:10.1007/978-3-030-51935-3_34.

[61] J. Masci, U. Meier, D. Cireşan, J. Schmidhuber, Stacked convolutional auto-encoders for hierarchical feature extraction, in: Artificial Neural Networks and Machine Learning–ICANN 2011: 21st International

Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I 21, Springer, 2011, pp. 52–59. `doi:10.1007/978-3-642-21735-7\_7`.

[62] F. Li, H. Qiao, B. Zhang, Discriminatively boosted image clustering with fully convolutional auto-encoders, Pattern Recognition 83 (2018) 161–173. `doi:10.1016/j.patcog.2018.05.019`.

[63] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl, et al., Constrained k-means clustering with background knowledge, in: Icml, Vol. 1, 2001, pp. 577–584.

Appendix A. Criteria

Appendix A.1. Adjusted rand index

$$\text{ARI} = \frac{\sum_{ij} \binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}$$

where:

- n_{ij} is the number of elements in both cluster i and true class j,
- a_i is the number of elements in cluster i,
- b_j is the number of elements in true class j,
- n is the total number of elements, and
- $\binom{x}{2}$ is the binomial coefficient.

Appendix A.2. Accuracy

$$\text{ACC} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\{y_i = \text{map}(c_i)\}$$

where:

- n is the total number of data points,
- y_i is the true label,
- c_i is the predicted label,
- $\mathbf{1}\{\cdot\}$ is the indicator function, and
- $\text{map}(c_i)$ is the mapping of cluster c_i to the true class y_i.

Appendix A.3. Normalised mutual information

$$\mathrm{NMI}(Y, C) = \frac{2 \cdot I(Y; C)}{H(Y) + H(C)}$$

where:

- $I(Y; C)$ is the mutual information between the true labels Y and the predicted clusters C,
- $H(Y)$ and $H(C)$ are the entropies of the true labels and clusters, respectively.

The mutual information $I(Y; C)$ is given by:

$$I(Y; C) = \sum_{i=1}^{k} \sum_{j=1}^{l} p(y_i, c_j) \log \frac{p(y_i, c_j)}{p(y_i) p(c_j)}$$

Appendix B. Default parameters

Table B.15: Default parameters of t-SNE.

Algorithm	barneshut
CacheSize	1,000
Distance	euclidean
Exaggeration	4
NumPCAComponents	0
Perplexity	30
Standardize	false
InitialY	$10^{-4} * randn(N, NumDimensions)$
LearnRate	500
NumPrint	20
Theta – Barnes-Hut tradeoff parameter	0.5
Verbose	0

Table B.16: Default parameters of UMAP (1st part).

n_neighbors	15
n_components	2
metric	euclidean
metric_kwds	None
output_metric	euclidean
output_metric_kwds	None
n_epochs	None
learning_rate	1
init	spectral
min_dist	0.1
spread	1
low_memory	True
n_jobs	-1
set_op_mix_ratio	1
local_connectivity	1
repulsion_strength	1
negative_sample_rate	5
transform_queue_size	4
a	None
b	None
random_state	None
angular_rp_forest	False
target_n_neighbors	-1
target_metric	categorical
target_metric_kwds	None
target_weight	0.5
transform_seed	42
transform_mode	embedding
force_approximation_algorithm	False
verbose	False
tqdm_kwds	None
unique	False
densmap	False
dens_lambda	2
dens_frac	0.3

Table B.17: Default parameters of UMAP (2nd part).

dens_var_shift	0.1
output_dens	False
disconnection_distance	None
precomputed_knn	(None, None, None)