SHADE: Deep Density-based Clustering

Anna Beer*[1], Pascal Weber*[1,2],
Lukas Miklautz[1], Collin Leiber[3,4], Walid Durani[3], Christian Böhm[1], Claudia Plant[1,5]
[1]*Faculty of Computer Science, University of Vienna*, Vienna, Austria
[2]*UniVie Doctoral School Computer Science*, Vienna, Austria
[3]*Database Systems and Data Mining, LMU Munich*, Munich, Germany
[4]*Munich Center for Machine Learning*, Munich, Germany
[5]*ds:UniVie, University of Vienna*, Vienna, Austria
{firstname.lastname}@univie.ac.at, {lastname}@dbs.ifi.lmu.de

***Abstract*—Detecting arbitrarily shaped clusters in high-dimensional noisy data is challenging for current clustering methods. We introduce SHADE, the first deep clustering algorithm that incorporates density-connectivity into its loss function. Similar to existing deep clustering algorithms, SHADE supports high-dimensional and large data sets with the expressive power of a deep autoencoder. In contrast to most existing deep clustering methods that rely on a centroid-based clustering objective, SHADE incorporates a novel loss function that captures density-connectivity. It thereby learns a representation that enhances the separation of density-connected clusters. SHADE detects a stable clustering and noise points fully automatically without any user input. It outperforms existing methods in clustering quality, especially on data that contain non-Gaussian clusters, such as video data. Moreover, the embedded space of SHADE is suitable for visualization and interpretation of the clustering results as the individual shapes of the clusters are preserved.**

***Index Terms*—Clustering, Deep Clustering, Density-based Clustering, DBSCAN**

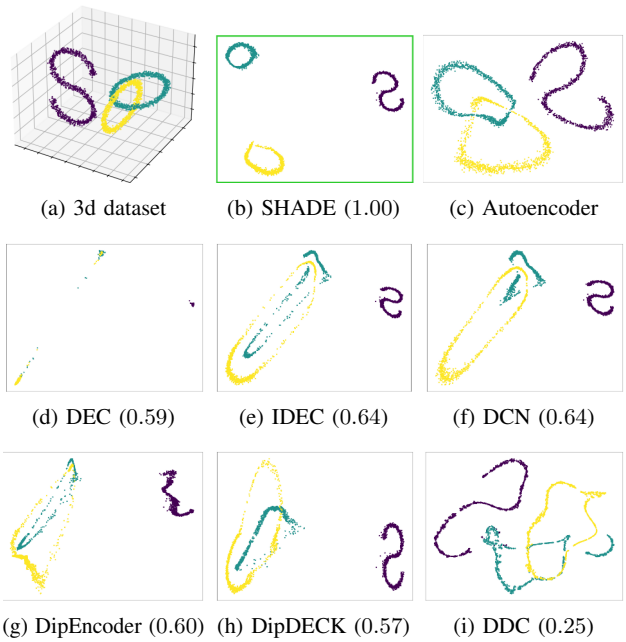
I. Introduction

Density-based clustering considers clusters as areas of high object density that are separated by areas of low object density, e.g., the 3d dataset in Fig. 1(a) with two intertwined rings and one s-shaped cluster. In contrast to our synthetic 3d toy data set, real-world data typically has a much higher dimensionality, which exacerbates finding density-based clusters.

We introduce SHADE (**S**tructure-preserving **H**igh-dimensional **A**nalysis with **D**ensity-based **E**xploration), a novel clustering method that integrates a density-connectivity loss into the training of a deep autoencoder. SHADE is the first method that learns a representation that enhances *density-based* clusters. Fig. 1(b) shows the 2d embedding of our 3d toy data learned by SHADE. SHADE enhances the density-based cluster structure by augmenting the separation of the three clusters while preserving their individual shapes. Using this embedding, SHADE finds – in contrast to all our deep clustering competitors – the perfect clustering.

Why not just use an autoencoder for representation learning followed by density-based clustering? Autoencoders minimize the reconstruction error of individual data points. In contrast, clustering aims at grouping similar points, which is a different goal: the two separate rings of our 3d example are connected

*First authors with equal contribution in alphabetical order.



(a) 3d dataset (b) SHADE (1.00) (c) Autoencoder
(d) DEC (0.59) (e) IDEC (0.64) (f) DCN (0.64)
(g) DipEncoder (0.60) (h) DipDECK (0.57) (i) DDC (0.25)

Fig. 1: Data set (a) and its 2d embedding created by our method SHADE (b), a regular autoencoder (c), and our competitors (d)-(i); colors imply ground-truth clusters, numbers give the ARI. SHADE separates the clusters and keeps their shapes, whereas other methods merge them or change the shape entirely.

in the 2d embedding of an autoencoder as Fig. 1(c) shows. Thus, using a classic autoencoder inhibits a correct clustering.

Recent deep clustering methods integrate specific clustering losses into the objective function of autoencoders. However, most approaches rely on a centroid-based cluster notion, similar to k-Means, e.g., [8], [18], [28], [29]. Figs. 1(d)-(f) demonstrate that these methods tend to fail on data with density-based clusters. In the embedded space of DEC [28], the clusters seem to be well separated, but the shapes are completely destroyed by forcing them to fit Gaussian distributions. The ARI measure of 0.59 shows that DEC cannot separate the two rings. IDEC and DCN do not enforce Gaussianity as strictly as DEC but also employ centroid-based losses. Both methods

2024 IEEE International Conference on Data Mining (ICDM) | 979-8-3315-0668-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICDM59182.2024.00075

2374-8486/24/$31.00 ©2024 IEEE
DOI 10.1109/ICDM59182.2024.00075

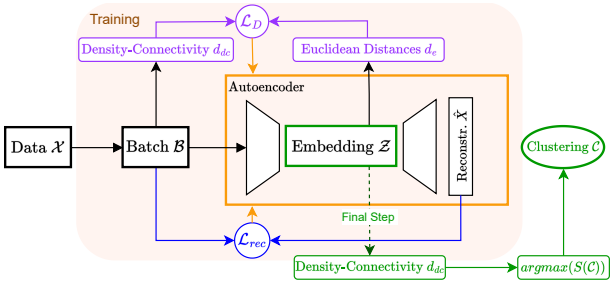


Fig. 2: SHADE optimizes the loss functions $\mathcal{L}_D$ and $\mathcal{L}_{rec}$ simultaneously in a batch-wise manner. $\mathcal{L}_D$ aligns the density-connectivity in the original space with the Euclidean distances in the embedding. $\mathcal{L}_{rec}$ enforces the preservation of the clusters' original shapes in the embedding, allowing an accurate reconstruction. SHADE returns the clustering $\mathcal{C}$ with the highest stability $S(\mathcal{C})$ based on the density-connectivity metric d_{dc}.

preserve the 's' shape but fail to separate the rings. DipDECK [15] and DipEncoder [14] support a more flexible cluster model by relying only on modalities instead of specific distributions. DipDECK further offers the benefit that users do not need to select the number of clusters as an input parameter. However, both methods perform similarly to DEC and IDEC, cf. Figs. 1(g) and 1(h): In the embedded space of the autoencoder, both rings heavily overlap. Fig. 1(i) shows the result of DDC [20], a recent sequential approach aimed at density-based clusters. However, DDC cannot separate the intertwined rings either, as it does not incorporate the density-based notion into the embedding. In consequence, it only yields a low ARI of 0.25.

Thus, we propose SHADE, the first approach that inherently combines the benefits of density-based and deep clustering. A version of this work with more elaborations and experiments can be found in [3] (http://arxiv.org/abs/2410.06265).

Our main contributions are as follows:

1) SHADE is the first deep density-based clustering algorithm. It learns a representation that enhances the separation of clusters while preserving their density-connectivity.
2) Empowered by an autoencoder, SHADE supports density-based clustering of high-dimensional data such as videos.
3) Inherited from density-based clustering, SHADE naturally supports noise without requiring its percentage beforehand.
4) SHADE automatically finds the right number of clusters.

II. A Novel Deep Density-Based Clustering Method

Density-based methods can find important structures that are not detectable with centroid-based methods, e.g., arbitrarily shaped clusters. However, finding such clusters in high-dimensional space is hard and can be slow. Hence, we first learn a low-dimensional embedding that enhances density-connected structures, where we then cluster the data. SHADE's main steps are shown in Fig. 2 and explained in the following.

A. Capturing Density-Connectivity

To enhance density-connected structures, we introduce a novel density-connectivity loss $\mathcal{L}_d$ in Section II-B. It captures the density-connectivity and enables the AE to transfer the relevant information from the original space into the embedding. It is based on the theoretical background given in the following.

1) Classic Density-Connectivity: For given parameters $\varepsilon \in \mathbb{R}_{>0}$ and $\mu \in \mathbb{N}$ (often also known as $minPts$), *core points* are points with at least μ points in their ε-range. Density-connected clusters are then maximal subsets of connected core points [7]. The core distance of a point $x \in \mathcal{X}$, $core\_dist(x)$ gives the distance to its μ-th nearest neighbor. Core points are *connected* if they are closer than ε. The transitive hull of density-connected core points defines a cluster and contains all points that are *density-reachable*, i.e., that have a 'chain' of core points connecting them. These concepts are used in a variety of literature, e.g., [7], [23]. The mutual reachability distance d_m is often used in methods building on top of it (e.g., [1], [5], [22]) and is defined as $d_m(x,y) = \max(d_{eucl}(x,y), core\_dist(x), core\_dist(y))$ for $x, y \in \mathcal{X}$, where d_{eucl} is the Euclidean distance.

2) Hierarchical Density-Connected Structures: Analogous to hierarchical single-linkage (SL) clustering [19], the minimum spanning tree (MST) captures the hierarchical structure of density-connected clusters. While the MST for SL clustering is based on the Euclidean distance, the MST for density-based clustering is based on the mutual reachability distance [2], [26]. The minimax path distances on this MST imply a tree metric d_{dc} that captures the smallest ε s.t. two points are density-reachable [2] and whose values can be stored in a tree $\mathcal{T}_d$. Its root node contains the length of the largest edge in the MST, and its leaves represent the points in the dataset. The distance $d_{dc}(x,y)$ between two points is stored in their lowest common ancestor in $\mathcal{T}_d$, and w.l.o.g., we can assume that $\mathcal{T}_d$ is binary.

B. Density-Connectivity Loss

We capture density-connectivity in the high-dimensional space with d_{dc} and enhance the structure by using the Euclidean distance in the low-dimensional embedding. As d_{dc} is path-based, it hides information about the entanglement of structures and reduces it to information about their connectivity. Thus, we minimize the following density-connectivity loss:

$$\mathcal{L}_d(\mathcal{X}) = \frac{1}{|\mathcal{X}|^2} \sum_{x_i, x_j \in \mathcal{X}} \left(d_{dc}(x_i, x_j) - d_{eucl}(z_i, z_j) \right)^2 \quad (1)$$

for any points x_i, x_j in $\mathcal{X}$ and their corresponding embedded points $z_i = enc(x_i), z_j = enc(x_j)$. Minimizing $\mathcal{L}_d$ preserves the inter-cluster distances and decreases intra-cluster distances, yielding a higher separability between clusters. As using the whole dataset at once is often not feasible, we optimize the loss function in a batch-wise manner, i.e., we regard $\mathcal{L}_d(\mathcal{B})$ for any points x_i, x_j in the batch $\mathcal{B}$. In Section III-C2, we show the stability across various batch sizes.

C. Preserving the Structure in the Embedding

To preserve the shapes of the clusters in the low-dimensional embedding, we additionally minimize the reconstruction error:

$$\mathcal{L}_{rec} = \frac{1}{|\mathcal{B}|} \sum_{x_i \in \mathcal{B}} \|x_i - \hat{x}_i\|_2^2 \quad (2)$$

where $\hat{x_i} = dec(enc(x_i))$ is the reconstruction of x_i.

We can enforce the separation of the different density-connected clusters and preserve the cluster structure by including $\mathcal{L}_d$, s.t. our overall loss function is: $\mathcal{L} = \mathcal{L}_{rec} + \mathcal{L}_d$

Training on this combined loss function creates an embedding that enhances density-connected structures while preserving their shapes, making it very suitable for visualizing data. It is, furthermore, the basis for finding density-based clusters with our novel clustering technique described in Section II-D. This technique unites multiple desirable and hard-to-reach properties: it returns a stable clustering, it detects noise, it finds the number of clusters fully automatically, and it allows clusters with different densities.

D. Fully Automatic Clustering and Noise Detection

We can now perform the final clustering step in the embedded space. As the clusters' potentially non-convex shapes are preserved in the embedding, we cannot simply apply k-Means like some of our competitors. Instead, we detect the density-based structures in the low-dimensional embedding based on the cluster hierarchy given by $\mathcal{T}_d$ (see Section II-A2). Note that $\mathcal{T}_d$ offers a variety of possible clusterings, some of which are better than others. While DBSCAN-like clustering can be obtained by thresholding the tree at a user-given ε, this would prevent the detection of clusters with different densities, and ε is hard to choose. Thus, automatically detecting the best partitioning for a given tree is desirable. We impose two additional challenges that are especially important for exploratory data analysis, as it is typical for our unsupervised setting: 1) the number of clusters should be detected automatically, and 2) points that do not belong to a cluster should not be assigned to a cluster, i.e., the algorithm needs to be able to detect noise. Note that no other deep clustering algorithm fulfills these requirements, yet. Our novel method, SHADE, is the first deep clustering method that inherently detects noise and is simultaneously part of a small group of deep clustering algorithms that can automatically identify the number of clusters. SHADE solves both problems by introducing a novel measure for stability and a method to find the most *stable* clustering.

1) Stability of Clusters: For SHADE, we define *stable* clusters as clusters that persist for a large variety of densities. Our stability value corresponds to the range of densities that produce the same cluster besides bordering or noise points. We compute it by first reducing the cluster hierarchy to nodes relevant to the structure of the tree and then regard the values of nodes in consecutive levels in this simplified *structure tree*.

Notation: A group of leaves $l(a)$ is defined by its lowest common ancestor node a in a tree $\mathcal{T}$ and has $|l(a)|$ many members. The node a stores the distance value $\mathcal{T}(a)$. We refer to the parent node of a with $p(a)$ and the children with $ch(a)$. For all $\varepsilon \geq \mathcal{T}_d(a)$ the nodes $l(a) \in \mathcal{T}_d$ build a proper density-connected cluster as known from DBSCAN$^{(*)}$ [7].

a) Structure Tree: To capture the relevant cluster splits (rather than splits between a cluster and bordering noise), we define a structure tree $\mathcal{T}$. It exclusively contains nodes of $\mathcal{T}_d$ where both children have at least μ leaves. $\mathcal{T} = \{a \in \mathcal{T}_d | \ \forall ch \in ch(a) : |l(ch)| \geq \mu\}$. Nodes $b \in \mathcal{T}_d$ that do not fulfill this property are skipped in $\mathcal{T}$, and if they have children, their leaves $l(b)$ are assigned to the topmost child of b that fulfills the property. This assigns bordering noise points within a distance $< \mathcal{T}_d(p(a))$ to their closest cluster. If, after this step, a node only has one remaining child, we merge the child node with its parent and keep the d_{dc} value of the child node.

b) Stability: The range of densities for which points $\in l(a)$ build a density-connected cluster c is the absolute difference between $\mathcal{T}_d(a)^{-1}$ and $\mathcal{T}_d(p(a))^{-1}$, considering the density given by $1/\varepsilon$. Regarding the structure tree $\mathcal{T}$ and including the sizes of the clusters gives us the stability $S(c)$[1]:

$$S(c) = \left(\frac{1}{\mathcal{T}_d(a)} - \frac{1}{\mathcal{T}_d(p(a))} \right) \cdot |l(a)| \tag{3}$$

The stability of a clustering $\mathcal{C}$ is the sum of its cluster's stabilities: $S(\mathcal{C}) = \sum_{c \in C} S(c)$

2) Automatically finding most stable clustering: Finding the most stable clustering efficiently is not trivial, as there are many potential clusterings for a given hierarchy. Note that we look for a flat clustering, where each cluster is defined by a root node containing all descendant leaves of the root node. Furthermore, especially for exploratory use cases, it is important to detect the number of clusters automatically – a property that most deep clustering methods do not have. Thus, instead of listing every possible clustering and computing its stability or aiming for a certain number of clusters, we find the most stable clustering similarly as [5] with a two-step approach where we only need one bottom-up pass through the structure tree followed by a top-down pass:

Bottom-up: We regard a cluster c_a implied by a common ancestor a. If a is a leaf node of $\mathcal{T}$, we flag the node a as a base case containing at least $2\mu - 2$ points. Else, if the stability $S(c_a)$ of c_a is larger than the sum of its children's stabilities, we also flag the node a: the cluster c_a leads to higher overall stability than choosing the clusters implied by a's children and is thus preferable: If $S(c_a) > \sum_{b \in ch(a)} S(c_b)$ then c_a is flagged. If not, we store the best possible stability so far in c_a and continue, i.e., $S(c_a) := \sum_{b \in ch(a)} S(c_b)$.

Top-down: Every flagged node with no flagged ancestor defines a cluster. For this, we can traverse the tree depth-first and return whenever we hit a flagged node. Nodes that are not flagged and have no flagged ancestor are noise.

III. Experiments

We describe the setup, data, and evaluation in Section III-A. We show analyses on synthetic data in Section III-B, an ablation study in Section III-C, our method's quality in Section III-D, and discuss its limitations in Section III-E.

A. Experiment Details

a) Algorithms: We compare SHADE with the following deep clustering methods (cf. Section IV): The k-means like DCN [29], DEC [28], and IDEC [8]; DipDECK [15] and

[1]Note the ultrametric property of $\mathcal{T}_d$ which determines that $\mathcal{T}_d(p(a)) \geq \mathcal{T}_d(a)$ for any a (besides the root node).

DipEncoder [14], deep clustering methods that are more flexible w.r.t. cluster shapes; and DDC [20] a sequential deep density-based clustering method. Note that except for SHADE, DipDECK, and DDC, all the other algorithms need the ground truth number of clusters given by the user in advance.

b) Setup: All methods were trained with a batch size of 500, a target embedding size of 10, Adam as the optimizer, and 50 pretrain epochs for the used AE for all methods, except for SHADE, which does not need a pretrained AE. All methods were then (further) optimized on their respective loss function for 100 epochs. We report average results over ten runs for each experiment. For preprocessing, we applied a z-normalization for all image data and a feature-wise z-normalization for all tabular data. For SHADE, we consistently use $\mu = 5$ for every experiment. As for the other methods, we used the default settings of [16] for all hyperparameters.
Our full code and supplementary material are available at https://github.com/pasiweber/SHADE.

c) Data: We evaluate all algorithms on synthetic, tabular, video, and image data (see Table IV). Furthermore, we also applied SHADE on popular (deep) clustering benchmark datasets whose clusters usually are of Gaussian shape, as Optdigits [17], USPS [9], MNIST [13], FMNIST [27], and KMNIST [6] and report the results in [3].

d) Evaluation: As SHADE is the first deep density-based clustering method that handles noise, the comparison to methods that do not detect noise is not trivial. Hence, in all tables and figures where we compare the results of our algorithm to other methods, we assign every detected noise point to its closest cluster. We label this comparison variant 'SHADE_1nn'. However, assigning every point to a cluster, instead of allowing noise, is neither our goal nor meaningful in exploratory data analysis. This assignment serves the sole purpose of enabling a fair, non-biased assessment of the clustering quality compared to our competitors according to best practices [24].

B. Systematic Evaluation regarding Noise

We evaluate the impact of up to 90% additive uniform noise on synthetic datasets with 5000 data points and 100 dimensions.

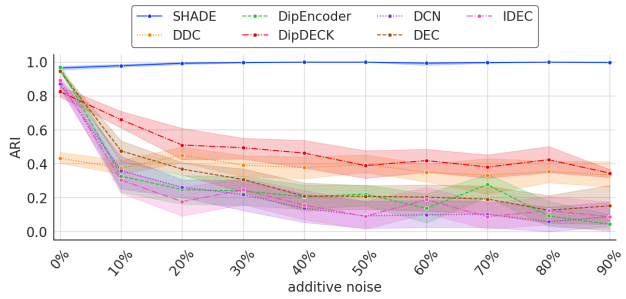


Fig. 3: ARI on synthetic data with varying noise ratio. Note that SHADE yields the highest quality and simultaneously the lowest variances across ten runs. This shows the importance of inherent noise handling for deep clustering algorithms.

Fig. 3 shows the clustering performance on the non-noise points, where SHADE excels as it is the only deep method that can detect noise points.

C. Ablation Study

1) Varying Parameter μ: Density-connectivity often depends on hyperparameters like μ and ε. As $\mathcal{T}_d$ captures all possible density-connected clusterings in dependence of ε, users do not need to predefine ε. While μ influences the robustness against the single link effect, it is often fixed to some low default value (e.g., in [5], [7]), as it does not significantly change the results. This aligns with established [26] and recent research [23]. In Table I, we exemplarily show SHADE's robustness w.r.t. μ on COIL20 as a representative dataset. Thus, we set $\mu = 5$ for *all experiments*, as this is enough to avoid single links, and similar values are often recommended (e.g., [7], [23]).

TABLE I: SHADE's average clustering results for varying μ. The default value for SHADE is $\mu = 5$ (gray column).

Dataset	Metric	$\mu = 3$	$\mu = 4$	$\mu = 5$	$\mu = 6$	$\mu = 7$
COIL20	ARI	83.6 ± 4.8	81.3 ± 8.4	82.5 ± 4.5	84.5 ± 3.1	83.4 ± 4.2
	noise	11.4 ± 2.2	10.9 ± 2.2	12.5 ± 1.9	14.2 ± 1.8	14.8 ± 3.4
	k	16.8 ± 0.6	16.6 ± 0.8	16.5 ± 1.0	16.9 ± 0.8	17.3 ± 0.8

2) Varying Batch Size: The batch size in the training phase of our algorithm is more important than in other deep clustering algorithms as it restricts us from learning an approximate optimum of our original loss function stated in Equation (1). Hence, we conducted an ablation study to show SHADE's robustness against different batch sizes, see Table II. We exemplarily report results for COIL20 here. We choose a batch size of 500 for all other experiments.

TABLE II: SHADE with different batch sizes. The default value is $batchsize = 500$ (gray column).

Dataset	Metric	100	300	500	700	900
COIL20	ARI	77.6 ± 8.4	84.1 ± 5.0	85.2 ± 1.9	82.7 ± 4.3	83.2 ± 4.8
	noise	14.0 ± 3.4	15.7 ± 2.1	13.6 ± 2.0	14.8 ± 1.9	13.0 ± 2.2
	k	17.9 ± 2.1	17.9 ± 1.6	16.9 ± 0.7	16.7 ± 1.6	16.7 ± 0.9

3) Different Parts of our Loss Function: Table III shows an ablation study regarding the different parts of our loss function: we test the original $\mathcal{L}_D$ combined with $\mathcal{L}_{rec}$, only the $\mathcal{L}_D$ loss, and only the $\mathcal{L}_{rec}$ loss. We report the average across all tested datasets. The combination of the $\mathcal{L}_D$ and $\mathcal{L}_{rec}$ loss performs best according to all three evaluation measures.

TABLE III: Individual performance and average scores of SHADE with varying loss functions.
Per default, both $\mathcal{L}_{rec}$ and $\mathcal{L}_d$ are included (gray column).

Dataset	Metric	$\mathcal{L}_D + \mathcal{L}_{rec}$	$\mathcal{L}_D$	$\mathcal{L}_{rec}$
Average	ARI	$\mathbf{62.6 \pm 5.9}$	61.9 ± 5.0	61.0 ± 5.4
	NMI	$\mathbf{79.0 \pm 4.4}$	78.0 ± 3.8	78.4 ± 3.3
	DCSI	$\mathbf{55.7 \pm 2.13}$	54.2 ± 2.4	53.4 ± 1.3

D. Performance Benchmark

Table IV shows our benchmark study with the competitors and datasets described in Section III-A. SHADE performs very

TABLE IV: Clustering results measured by ARI of SHADE_1nn, where we assign all noise points to their nearest cluster for better comparability with our competitors. Note that we discuss results for TCGA in Section III-D and for HAR in Section III-E.

	Dataset	SHADE_1nn	DDC	DipDECK	DipEncoder	DCN	DEC	IDEC	src
Tabular data	Synth_low	**98.9 ± 2.0**	56.9 ± 5.5	33.9 ± 6.4	10.1 ± 9.9	9.6 ± 9.7	40.2 ± 3.0	15.3 ± 8.8	[10]
	Synth_high	**97.5 ± 1.4**	33.9 ± 11.1	29.9 ± 13.6	9.3 ± 10.7	8.8 ± 10.5	30.3 ± 3.5	17.9 ± 6.6	
	HAR	36.4 ± 6.4	49.4 ± 3.3	51.3 ± 4.0	60.0 ± 6.9	**66.1 ± 1.3**	63.4 ± 2.4	64.9 ± 0.9	[17]
	letterrec.	23.0 ± 0.9	9.9 ± 2.9	7.3 ± 3.5	24.7 ± 1.3	22.6 ± 1.0	23.9 ± 1.6	**25.2 ± 1.8**	
	htru2	**65.0 ± 19.5**	49.4 ± 13.0	9.7 ± 19.4	4.3 ± 0.8	49.7 ± 2.8	3.0 ± 0.5	3.2 ± 0.6	
	Mice	**27.7 ± 2.9**	25.2 ± 1.9	22.7 ± 4.3	21.6 ± 2.6	21.7 ± 1.4	22.0 ± 1.5	21.8 ± 1.4	
	TCGA	80.0 ± 13.7	87.5 ± 0.8	88.8 ± 4.4	**93.4 ± 6.0**	87.2 ± 5.3	85.1 ± 2.7	82.6 ± 0.9	
	Pendigits	75.1 ± 0.8	**76.9 ± 2.0**	74.3 ± 1.1	64.6 ± 3.0	61.6 ± 1.9	65.7 ± 3.3	64.9 ± 2.6	
Video	Weizmann	**48.2 ± 3.6**	14.7 ± 1.8	12.0 ± 1.9	23.3 ± 1.2	24.6 ± 1.1	24.9 ± 1.2	24.7 ± 1.2	[4]
	Keck	**7.5 ± 0.4**	−0.2 ± 1.1	6.9 ± 0.8	7.1 ± 0.3	6.4 ± 0.5	6.1 ± 0.9	6.2 ± 0.9	[30]
Image	COIL20	**68.7 ± 3.5**	62.0 ± 5.5	50.5 ± 7.8	64.0 ± 3.0	62.4 ± 2.8	63.7 ± 2.8	62.9 ± 2.9	[17]
	COIL100	56.8 ± 5.0	16.4 ± 3.8	21.4 ± 3.0	54.3 ± 1.9	55.9 ± 3.0	55.8 ± 2.0	**56.9 ± 2.0**	
	cmu_faces	34.6 ± 6.2	35.0 ± 3.5	29.8 ± 9.8	37.9 ± 2.2	**40.3 ± 2.0**	35.8 ± 2.8	39.4 ± 3.3	

well on almost all datasets. One exception is the TCGA dataset, which contains five different tumor types as clusters. Based on the good results of centroid-based methods in Table IV, we believe that TCGA contains Gaussian-like clusters, which are well-represented with prototypes for each tumor type. Note that we tested data that *potentially* contains density-connected structures – many benchmark datasets like MNIST do not contain *density-connected*, but convex clusters and should rather be clustered with centroid-based deep clustering algorithms.

Table IV shows that SHADE is remarkably successful on the datasets where we know that they contain density-based structures, i.e., the synthetic datasets and the Weizmann video data. On synthetic datasets, our algorithm reaches an ARI above 97.0, whereas the second-best method merely reaches an ARI of 56.9 resp. 33.9. For both datasets, the runner-up is DDC – our only competitor including a concept of density.

E. Limitations

On HAR, our competitors outperformed SHADE. However, the confusion matrix in Fig. 4 and a brief analysis of the class labels shows why: HAR contains motion data for six activities. Three of them describe motions and three are resting activities. These classes are density-connected, as the transition between the activities *walk*, *walk upstairs*, and *walk downstairs* are smooth. SHADE almost perfectly differentiates between moving and not moving activities, which might not yield high ARI values but is a reasonable clustering nevertheless.

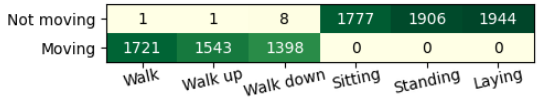


Fig. 4: HAR Confusion Matrix. x-axis gives ground truth classes, y-axis clusters detected by SHADE

IV. Discussion and Related Work

A. Density-based Clustering

Density-based algorithms (e.g., DBSCAN [7] or Density Peaks [21]) have several important advantages compared to other clustering paradigms. Most of them can find clusters of arbitrary shape, detect the number of clusters automatically, and can handle noise. These are invaluable properties for real-world use cases where the expected results or properties of the data are not known beforehand. Where centroid-based clustering methods are often based on strong assumptions about the data, like assuming underlying Gaussian distributions, density-based methods are more flexible and data-driven. The associated hyperparameters, however, are often hard to set.

Furthermore, density-based clustering methods are often slow and do not perform well in high-dimensional data. Thus, finding density-connected structures is, in general, no trivial task, especially for large and high-dimensional data.

B. Deep Clustering

Deep clustering describes the combination of deep learning techniques with specific clustering objectives. Here, various deep learning architectures can be used, with most deep clustering methods based on feedforward [11] or convolutional [12] autoencoders (AEs). AEs transform the data into a lower-dimensional space by applying the encoding function $enc(\cdot)$ and try to reconstruct the original representation using the decoding function $dec(\cdot)$. Deep clustering approaches utilizing AEs usually optimize the loss function $\mathcal{L} = \lambda_1 \mathcal{L}_{rec} + \lambda_2 \mathcal{L}_{clust}$, where $\mathcal{L}_{rec}$ validates the quality of the embedding by comparing the input x and output $enc(dec(x))$ of the AE and $\mathcal{L}_{clust}$ strengthens the existing clustering structures. Most established deep clustering methods use a centroid-based objective as the basis for $\mathcal{L}_{clust}$. They are often related to k-Means using either soft assignments, like DEC [28] and IDEC [8] or hard assignments, like DCN [29], and ACe/DeC [18] for centroid-based deep clustering. The centroid-based objective pulls clusters together such that they can collapse onto their cluster centroid. While this can improve clustering accuracy, it destroys the density-based structure. Another problem is the assumption that distances within the clusters are smaller than distances between clusters – which is not true for density-connected clusters. Other methods attempt to address this problem. E.g., DipEncoder [14] considers modalities instead of distances. Here, each cluster can obtain an individual spread, leading

to more diverse patterns within the embedding. However, it can still only identify convex cluster shapes. DipDECK [15] overcomes this limitation by initially overestimating the number of clusters, which are later merged if they show a coherent structure. Here, multiple k-means-like clusters can be combined to capture more complex patterns.

C. Sequential Deep Density-based Clustering

Current methods at the intersection of deep and density-based clustering only work sequentially. They first learn a representation (independently of the density-based clustering objective) using a neural network and, afterward, apply a (non-deep) density-based clustering method on top of this representation. E.g., DDC [20] sequentially applies an AE, t-SNE [25], and a variant of Density-Peak Clustering (DPC) [21] in the embedding. While DDC profits from the advantages of both the AE and t-SNE, it also inherits the drawbacks of both. If the AE learns a poor initial embedding, undesired structures are reinforced by t-SNE, resulting in an insufficient final representation. Outliers can strongly distort the result of t-SNE. As DDC follows a sequential approach, it cannot improve the representation w.r.t. the density-based clustering concept. Note that by applying DPC as a last step, DDC finds density-*based* clusters, but not necessarily density-*connected* structures (see Section IV-A): e.g., DDC does not find elongated density-connected structures as shown in Fig. 1. Nevertheless, DDC achieves an overall strong cluster performance.

V. Conclusion

We present SHADE, the first deep clustering method that incorporates density in its loss function. It preserves density-connected structures in low-dimensional embeddings, allowing a meaningful visualization that is especially useful for exploratory data analysis. SHADE finds arbitrarily shaped clusters and even topologically intertwined structures while automatically detecting the number of clusters. Our experiments show SHADE's superiority on various datasets and noisy data. For more elaborations and experiments, we refer to https://arxiv.org/abs/2410.06265.

Acknowledgments

We acknowledge financial support from the Vienna Science and Technology Fund (WWTF-ICT19-041) and the Austrian funding agency (FFG-903641).

References

[1] M. Ankerst, M. M. Breunig, H. Kriegel, and J. Sander, "OPTICS: ordering points to identify the clustering structure," in *SIGMOD Conference*. ACM Press, 1999, pp. 49–60.

[2] A. Beer, A. Draganov, E. Hohma, P. Jahn, C. M. Frey, and I. Assent, "Connecting the dots–density-connectivity distance unifies dbscan, k-center and spectral clustering," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 80–92.

[3] A. Beer, P. Weber, L. Miklautz, C. Leiber, W. Durani, C. Böhm, and C. Plant, "Shade: Deep density-based clustering," 2024. [Online]. Available: https://arxiv.org/abs/2410.06265

[4] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," in *The Tenth IEEE International Conference on Computer Vision (ICCV'05)*, 2005, pp. 1395–1402.

[5] R. J. G. B. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Advances in Knowledge Discovery and Data Mining*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 160–172.

[6] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha, "Deep learning for classical japanese literature," *arXiv preprint arXiv:1812.01718*, 2018.

[7] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[8] X. Guo, L. Gao, X. Liu, and J. Yin, "Improved deep embedded clustering with local structure preservation," in *IJCAI*. ijcai.org, 2017, pp. 1753–1759.

[9] J. J. Hull, "A database for handwritten text recognition research," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 16, no. 5, pp. 550–554, 1994.

[10] P. Jahn, C. M. M. Frey, A. Beer, C. Leiber, and T. Seidl, "Data with density-based clusters: A generator for systematic evaluation of clustering algorithms," in *ECML/PKDD (7)*, ser. Lecture Notes in Computer Science, vol. 14947. Springer, 2024, pp. 3–21.

[11] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE journal*, vol. 37, no. 2, pp. 233–243, 1991.

[12] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989.

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[14] C. Leiber, L. G. M. Bauer, M. Neumayr, C. Plant, and C. Böhm, "The dipencoder: Enforcing multimodality in autoencoders," in *KDD*. ACM, 2022, pp. 846–856.

[15] C. Leiber, L. G. M. Bauer, B. Schelling, C. Böhm, and C. Plant, "Dip-based deep embedded clustering with k-estimation," in *KDD*. ACM, 2021, pp. 903–913.

[16] C. Leiber, L. Miklautz, C. Plant, and C. Böhm, "Benchmarking deep clustering algorithms with clustpy," in *ICDM (Workshops)*. IEEE, 2023, pp. 625–632.

[17] K. N. Markelle Kelly, Rachel Longjohn, "The uci machine learning repository," 2023. [Online]. Available: http://archive.ics.uci.edu/ml

[18] L. Miklautz, L. G. M. Bauer, D. Mautz, S. Tschiatschek, C. Böhm, and C. Plant, "Details (don't) matter: Isolating cluster information in deep embedded spaces," in *IJCAI*. ijcai.org, 2021, pp. 2826–2832.

[19] F. Murtagh, "A survey of recent advances in hierarchical clustering algorithms," *The computer journal*, vol. 26, no. 4, pp. 354–359, 1983.

[20] Y. Ren, N. Wang, M. Li, and Z. Xu, "Deep density-based image clustering," *Knowl. Based Syst.*, vol. 197, p. 105841, 2020.

[21] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *science*, vol. 344, no. 6191, pp. 1492–1496, 2014.

[22] E. Schubert, S. Hess, and K. Morik, "The relationship of dbscan to matrix factorization and spectral clustering." in *LWDA*, 2018, pp. 330–334.

[23] E. Schubert, J. Sander, M. Ester, H. Kriegel, and X. Xu, "DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN," *ACM Trans. Database Syst.*, vol. 42, no. 3, pp. 19:1–19:21, 2017.

[24] T. Ullmann, A. Beer, M. Hünemörder, T. Seidl, and A. Boulesteix, "Over-optimistic evaluation and reporting of novel cluster algorithms: an illustrative study," *Adv. Data Anal. Classif.*, vol. 17, no. 1, pp. 211–238, 2023.

[25] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[26] D. Wishart, "Numerical classification method for deriving natural classes," *Nature*, vol. 221, no. 5175, pp. 97–98, 1969.

[27] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[28] J. Xie, R. B. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *ICML*, ser. JMLR Workshop and Conference Proceedings, vol. 48. JMLR.org, 2016, pp. 478–487.

[29] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards k-means-friendly spaces: Simultaneous deep learning and clustering," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 70. PMLR, 2017, pp. 3861–3870.

[30] L. Zhe, J. Zhuolin, and L. Davis, "Recognizing actions by shape-motion prototype trees," in *IEEE International Conference on Computer Vision (ICCV)*, vol. 2009, 2009, pp. 444–451.