

SHADE: Deep Density-based Clustering

Anna Beer^{*1}, Pascal Weber^{*1,2}

Lukas Miklautz¹, Collin Leiber^{3,4}, Walid Durani³, Christian Böhm¹, Claudia Plant^{1,5}

¹Data Mining and Machine Learning, University of Vienna, Vienna, Austria

²UniVie Doctoral School Computer Science, Vienna, Austria

³Database Systems and Data Mining, LMU Munich, Munich, Germany

⁴Munich Center for Machine Learning, Munich, Germany

⁵ds:UniVie, Vienna, Austria

{firstname.lastname}@univie.ac.at, {lastname}@dbs.ifl.lmu.de

Abstract—Detecting arbitrarily shaped clusters in high-dimensional noisy data is challenging for current clustering methods. We introduce SHADE (Structure-preserving High-dimensional Analysis with Density-based Exploration), 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. SHADE 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; see, for example, the 3d dataset in Figure 1(a) with two intertwined rings and one s-shaped cluster. While this is a synthetic example, density-based clusters are also common in real-world data: e.g., clusters following geographical structures or evolutionary development processes [5]. Real-world data typically exhibits a much higher dimensionality than our synthetic example, often in the order of hundreds or thousands of dimensions. Density-based structures are especially hard to find in such high-dimensional spaces because of the empty-space problem [35].

We introduce SHADE (Structure-preserving High-dimensional Analysis with Density-based Exploration), a novel clustering method that integrates a density-connectivity loss into the training of a deep autoencoder. Many recent deep clustering methods, such as [12], [23], [41], exploit the expressive power of deep autoencoders to learn cluster-friendly representations of high-dimensional data. SHADE is the first method that focuses on learning a representation that enhances density-based clusters.

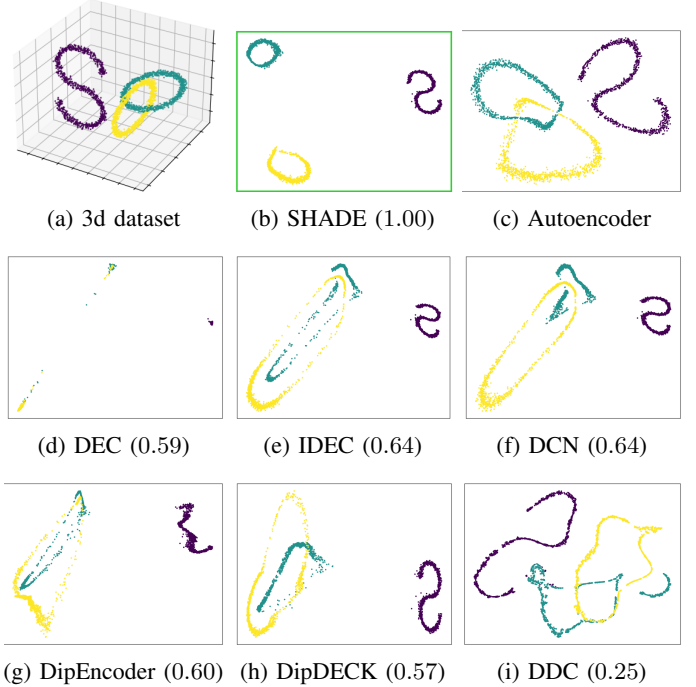


Fig. 1: 3d dataset (a) and its 2d embedding created by our algorithm SHADE (b), a regular autoencoder (c), and its competitors (d)-(i); colors imply ground truth clusters and numbers in brackets show the clustering quality measured by ARI. SHADE separates the clusters and keeps their shapes, whereas other methods merge them or change the shape entirely.

Figure 1(b) shows the representation learned by SHADE when reducing the dimensionality of our 3d synthetic dataset to 2d. SHADE enhances the density-based cluster structure by improving the separation of the three clusters while preserving their individual shapes. Using the density-connectivity information that is part of SHADE’s loss function, our algorithm also outputs a density-based clustering, which is perfect on this dataset with an ARI of 1.0.

Note that real-world data often has more dimensions than traditional density-based methods as, e.g., DBSCAN [9] and OPTICS [1], can handle – however, they work well on our 3d toy

*First authors with equal contribution in alphabetical order.

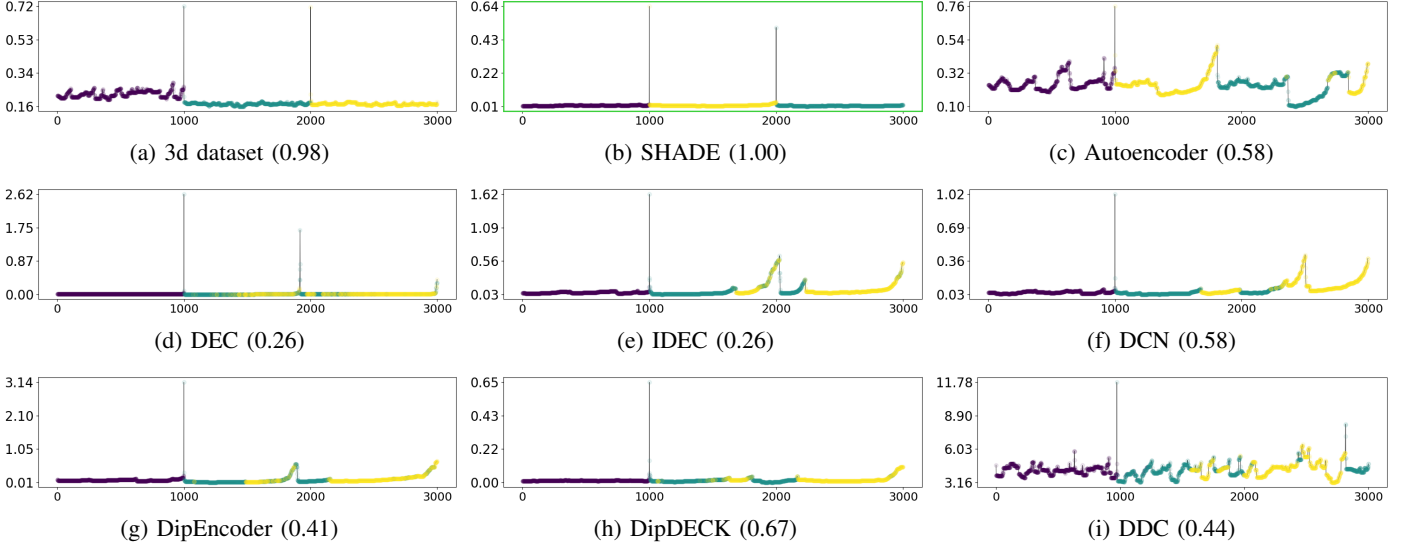


Fig. 2: OPTICS [1] reachability plots of the 3d dataset (a) and of its 2d embedding created by our algorithm SHADE (b), a regular autoencoder (c), and our competitors (d)-(i); colors imply ground truth clusters and numbers in brackets show the cluster separability measured by the density cluster separability index (DCSI). SHADE retains the original 3d dataset’s overall reachability structure and enhances cluster separability. In contrast, other methods merge the two intertwined clusters or fail to separate the intertwined clusters correctly. Additionally, the other methods reduce the overall cluster separability.

dataset. Figure 2(a) shows the OPTICS reachability plot of the 3d data. The colors correspond to the ground truth classes, see also Figure 1. All three clusters are visible as class-pure valleys in the plot that are separated by high reachability distances. To add a quantitative measure, we also report the DCSI [11], which is an internal validity measure for density-based clustering that scales between 0 and 1. DCSI measures the density-connectedness within classes and the separation between classes. In the original 3d space, the ground truth clusters are well density-separated, expressed by a very high DCSI of 0.98.

Figure 2(b) shows that SHADE improves the density-connectivity within the classes, which results in a perfect DCSI measure of 1.0. At the same time, SHADE preserves the shapes of the original clusters, see Figure 1(b).

Why not just use an autoencoder for representation learning followed by density-based clustering? Autoencoders minimize the reconstruction error of individual data points using some objective function. However, in clustering, we want to group similar points, which is a different objective. In the 2d embedding of the autoencoder in Figure 1(c), the two rings are connected, leading to a rather low DCSI of 0.58. The corresponding OPTICS plot in Figure 2(c) highlights this problem: some of the yellow points are in the same valley as some of the green points, i.e., they have a closer connection to points of the wrong ground truth cluster. Thus, the 2d embedding of the data by a classic autoencoder does not allow a correct, density-based clustering anymore.

Therefore, recent research papers on deep clustering have integrated specific clustering losses into the objective function of autoencoders. The vast majority of approaches rely on a centroid-

based cluster notion, similar to k -Means, e.g., [10], [12], [27], [41], [42]. Figures 1(d)-(f) and 2(d)-(f) demonstrate that these methods tend to fail on data with density-based clusters. In the embedded space of DEC [41], 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 and the OPTICS plot in the embedded space demonstrate that DEC cannot separate the two rings. The corresponding reachability plot has two valleys, but these clusters consist of a mixture of points from both classes. Consequently, the DCSI measure is very low, with a value of only 0.26. IDEC and DCN do not enforce Gaussianity as strictly as DEC but also employ centroid-based losses. Both methods preserve the ‘s’ shape but fail to separate the rings.

The recently proposed methods DipDECK [23] and DipEncoder [22] 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 in terms of ARI and DCSI, cf. Figures 1(g), 1(h), 2(g), and 2(h). In the embedded space of the autoencoder, both rings heavily overlap. Therefore, considering modalities does not help.

In Figures 1(i) and 2(i), we see the result of DDC [30], a recent sequential approach that is more elaborated than just using density-based clustering on the autoencoder embedding. After autoencoder pretraining, t-SNE is applied to enhance cluster structures and to reduce the dimensionality. As its last step, DDC clusters the data in the latent space with a variation of the DensityPeaks algorithm. On our example dataset, this approach

cannot separate the intertwined rings either. The initial problems emerge from the mismatch of the objective function of the autoencoder and the goal of density-based clustering, which cannot be cured by t-SNE, resulting in an ARI of 0.25 and a DCSI of 0.44.

Therefore, we propose SHADE, the first approach combining the benefits of density-based and deep clustering. Our main contributions are as follows:

- 1) SHADE is the first deep density-based clustering algorithm, enabling joint representation learning and density-based clustering.
- 2) The representation learned by SHADE enhances the separation of density-based clusters while preserving their density-connectivity.
- 3) Empowered by a deep autoencoder, SHADE supports density-based clustering of high-dimensional data such as images and videos.
- 4) Inherited from density-based clustering, SHADE naturally supports noise.
- 5) SHADE detects density-based clusters and noise without requiring the user to know the number of clusters or percentage of noise beforehand

II. A NOVEL DEEP DENSITY-BASED CLUSTERING METHOD

Aiming to find clusters that consist of a series of similar objects (instead of objects that are *all* similar to only *one* centroid) makes fewer assumptions about the data: for a cluster assignment, a point just needs to be close to a few of *any* other points in the cluster. The shapes of clusters are less restricted, and thus, density-based methods can detect important structures that are not detectable with centroid-based methods. However, capturing density-connected clusters in high-dimensional space is hard and can be slow. Hence, we first learn a low-dimensional embedding where we capture and preserve the density-connected structures within the high-dimensional space. Afterward, we cluster the embedded data points in the low-dimensional space.

SHADE's main steps are shown in Figure 3, which we explain in more detail throughout this section.

A. Challenges of Density-Connected Structures

We identify three main challenges that come with density-connected structures in high-dimensional data:

- 1) *Separability between intra- and inter-cluster distances:* For density-connected clusters, the (pairwise) Euclidean distance between points does not necessarily correlate with cluster membership: Clusters of arbitrary shape can be elongated, yielding large Euclidean intra-cluster distances that are potentially higher than inter-cluster distances. E.g., the Euclidean distance between points within a circular cluster as in Figure 1 can be as large as the diameter of the circle. However, the Euclidean distance between the closest points of different clusters is only half the diameter.

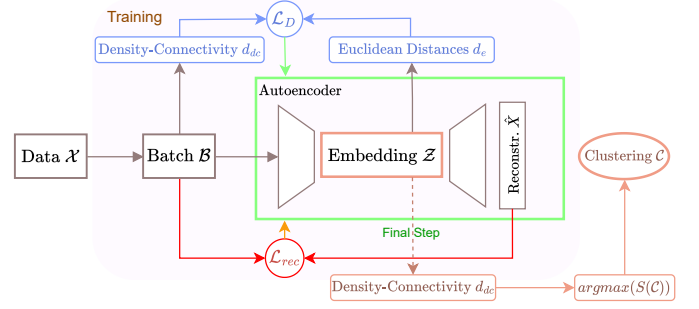


Fig. 3: SHADE optimizes the loss functions \mathcal{L}_D and \mathcal{L}_{rec} simultaneously in a batch-wise manner. \mathcal{L}_D aligns the density-connectivity in the original space with the Euclidean distances in the embedding. \mathcal{L}_{rec} , on the other hand, enforces that the original high-dimensional spatial structure or shape of the clusters is preserved in the learned embedding, allowing an accurate reconstruction. The final clustering \mathcal{C} is obtained by selecting the clustering with the highest stability $S(\mathcal{C})$ based on the density-connectivity metric d_{dc} .

- 2) *Preserving distances between structurally relevant points:* Especially for points that lie between two clusters, slight offsets in the embedding can have grave consequences regarding the density-connected cluster structure. In contrast, points at the border of the data space might not change the clustering result at all. Thus, the preservation of distances between points that are crucial for the clustering structure should be prioritized.

- 3) *Non-contractible and intertwined clusters:* Density-connected clusters might consist of intertwined topological structures that cannot be separated by AEs that produce a smooth transition of the data to the embedding. AEs based on the reconstruction loss are powerful tools for clustering if the points of each cluster lie in a *contractible* space [14]. Intuitively, a topological space is contractible if it can be shrunk to a point by a continuous function. A smooth AE could find such a function or embedding when given an according loss function. After contracting each cluster's space to a point within that space, i.e., embedding the points, the clusters are trivially separable in the embedding. (e.g., centroid-based deep clustering approaches might map all points of a cluster onto its center). Density-connected clusters imply *simply connected* spaces that might be non-contractible, e.g., an annulus or a ring, as shown in Figure 1. These clusters cannot be contracted onto a point, and if they are intertwined as in Figure 1, they are not easily separable in any embedding given by a smooth function.

B. Capturing Density-Connectivity

To preserve and enhance density-connected structures and tackle the aforementioned challenges, we introduce a novel density-connectivity loss \mathcal{L}_d in Section II-C. This loss needs to capture the density-connectivity and enable the AE to transfer the relevant information from the high-dimensional space into the low-dimensional embedding. To define it, we need

some theoretical background knowledge on density-connectivity (Section II-B1 and Section II-B2).

1) *Background: Classic Density-Connectivity*: Density-connected clusters are defined as maximal subsets of connected *core points* [9]. For parameters $\varepsilon \in \mathbb{R}_{>0}$ and $\mu \in \mathbb{N}$ that are usually given by users¹, core points are defined as points with at least μ points in their ε -range. 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 density-connected 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., [9], [25], [32], [33]. The mutual reachability distance d_m is often used in methods building on top of it (e.g., [1], [6], [7], [32]) 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) *Background: Hierarchical Density-Connected Structures*: Analogous to hierarchical single-linkage clustering [29], the minimum spanning tree (MST) captures the hierarchical structure of density-connected clusters. While single-linkage clustering uses the Euclidean distance graph as a basis for the MST, for density-connected structures, the hierarchy is based on the MST of the graph given by the mutual reachability distance [3], [39].

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 [3]. We can store the values of d_{dc} in a tree \mathcal{T}_d , as d_{dc} is a tree-metric. 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 without loss of generality, we can assume \mathcal{T}_d to be a binary tree. Capturing the density-connectivity via \mathcal{T}_d has several advantages: It can leverage some problems connected to the curse of dimensionality, e.g., after constructing the MST, all subsequent computations regarding the density are independent of the original dimensionality. Furthermore, \mathcal{T}_d captures the full hierarchical structure of the data instead of a single partitioning, allowing it to capture noise and clusters of different densities.

C. Density-Connectivity Loss

We propose a novel loss function that captures the relevant information of density-connected structures in the high-dimensional space and transfers them into the embedding. In the following, we describe the key ideas behind our proposed density-connectivity loss \mathcal{L}_d (Equation 1) that is able to tackle all these challenges.

The d_{dc} metric captures density-connectivity, with inter-cluster distances consistently larger than intra-cluster distances [3], a promising solution for **challenge (1)**. However, using this property effectively for the loss function is not trivial. As d_{dc} is an ultrametric, any data \mathcal{X} can be trivially embedded into the one-dimensional space with full preservation of all pairwise d_{dc} . Thus, preserving the d_{dc} from the original space to the

embedding would not yield meaningful structures. Thus, we use d_{dc} only in the original space while using the Euclidean distance in the embedded space. This allows learning an embedding where points within a density-connected structure are close, and points in different clusters are far apart.

To address the varying relevance of points, **challenge (2)**, note that only distances along the MST of the mutual reachability distance are important for the hierarchy of high-dimensional density-connected structures. Employing d_{dc} in the original space automatically omits irrelevant distance values between points that are not directly connected in the MST. Thus, only the distances relevant to the (hierarchical) clustering structure are explicitly used by the tree metric. Therefore, the relevant points’ influence is captured in d_{dc} .

Challenge (3), the problem of non-contractible and intertwined clusters, is solved by reducing the fully connected distance graph to the MST: Capturing the density-connected structure as a path-based distance hides the information about the entanglement of structures and reduces it to information about their connectivity. E.g., regard the two intertwined rings from Figure 1, where both circles have the same radius r , which also corresponds to the distance of the points of one ring to its center. The only information stored about the relationship between the two rings is the length of the link in the MST between them. Thus, any point of one ring has exactly the same $d_{dc} = r$ to any point of the other ring. Note that a dataset with two rings that are *not* intertwined and have an Euclidean distance of $3r$ between their centers has the exact same distance matrix regarding d_{dc} . Therefore, relying on the MST allows the autoencoder to split apart the rings and similar intertwined structures. Thus, d_{dc} carries all the information that we need for successfully tackling challenges (1) to (3), and we get an embedding with the desired properties by minimizing the following density-connectivity loss:

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

for any points x_i, x_j in \mathcal{X} and their corresponding embedded points $z_i = enc(x_i), z_j = enc(x_j)$.

However, using the whole dataset at once is often not feasible when training the loss function. Hence, it is optimized in a batch-wise manner. We can then approximate our original loss functions with the following loss:

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

for any points x_i, x_j in the batch \mathcal{B} and their corresponding embedded points $z_i = enc(x_i), z_j = enc(x_j)$. In Section III-D2, we show that this approach yields stable results across various batch sizes, making it valuable for real-world applications.

D. Preserving the Structure in the Embedding

By minimizing the density-connectivity loss, we preserve the inter-cluster distances and compress the points within one cluster together such that the distance between them corresponds to the

¹ μ is also known as *minPts* in some literature

d_{dc} distance in the high-dimensional space. That way, we also achieve better separability between clusters.

To preserve the overall cluster structure in the low-dimensional embedding, e.g., their shapes, we additionally minimize the reconstruction error:

$$\mathcal{L}_{rec} = \frac{1}{|\mathcal{B}|} \sum_{x_i \in \mathcal{B}} \|x_i - \hat{x}_i\|_2^2 \quad (3)$$

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

As shown in Figure 1, the AE, which only minimizes the reconstruction loss \mathcal{L}_{rec} , already retains the cluster structure to a certain degree but fails to separate the intertwined clusters. We can enforce the separation of the different density-connected clusters and preserve the cluster structure by including our novel density-connectivity loss \mathcal{L}_d .

Training on this combined loss function creates an embedding that preserves and enhances density-connected structures. This embedding is very suitable for visualizing data, as it preserves the shapes and cluster structure. It is, furthermore, the foundation for finding density-connected structures, which we do by applying a novel clustering technique described in Section II-E in the embedding. 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. Combining deep learning with an incorporated density-connectivity loss and this technique, we are able to cluster very high-dimensional data by enhancing the density-connectivity and the shape of clusters in the embedding space.

E. Fully Automatic Clustering and Noise Detection in the Embedded Space

We can now perform the final clustering step in the embedded space obtained with the steps described above. Note that we cannot simply apply k -Means in the embedded space like some of our competitors, as the density-connected structures from the high-dimensional space are preserved. As a result, clusters are typically not convex, rendering centroid-based clustering methods ineffective. 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-B2). Note that \mathcal{T}_d offers a variety of possible clusterings, some of which are better than others. For example, DBSCAN-like clustering results in the embedding can be obtained by thresholding the tree at a user-given ε . However, this approach prevents 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 to the best of our knowledge, no other deep clustering algorithm fulfills these requirements. 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* as explained in the following. For other notions of stability and delimitation to those, we refer to Section IV-D.

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 $ch(a)$. The nodes $l(a) \in \mathcal{T}_d$ build a proper² density-connected cluster for all $\varepsilon \geq \mathcal{T}_d(a)$.

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. $\mathcal{T} = \{a \in \mathcal{T}_d \mid \forall ch \in ch(a) : |l(ch)| \geq \mu\}$. 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 number of points in a cluster c gives us the stability $S(c)$ ³:

$$S(c) = \left(\frac{1}{\mathcal{T}_d(a)} - \frac{1}{\mathcal{T}_d(p(a))} \right) \cdot |l(a)| \quad (4)$$

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

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 [6] with a two-step approach where we only need

²According to the definitions of DBSCAN(*) [9]

³Note 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).

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 $S(c_a) > \sum_{b \in \text{ch}(a)} S(c_b)$ then c_a is flagged. If not, we store the best possible stability so far in c_a and continue, i.e., $S(c_a) := \sum_{b \in \text{ch}(a)} S(c_b)$.

Top-down: Every flagged node with no flagged ancestor defines a cluster. For this, we can simply go through the tree in a depth-first pass 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, perform a systematic analysis on synthetic data in Section III-C, perform an ablation study in Section III-D, show our method’s quality on benchmark data in Section III-E, and discuss its limitations in Section III-F.

A. Experiment Details

a) Algorithms: We compare SHADE with the following deep clustering methods (cf. Section IV): The k -means like deep clustering methods DCN [42], DEC [41], and IDEC [12]; DipDECK [23] and DipEncoder [22], deep clustering methods that are more flexible w.r.t. cluster shapes; DDC [30] 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 our method, 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 the ClustPy package implementation [24] for all their respective hyperparameters. Our full code is available at <https://github.com/pasiweber/SHADE>.

c) Data: We evaluate all algorithms on synthetic data (e.g., Synth_low, Synth_high) generated and accessible by the high-dimensional density-connected data generator DENSIRE [18], tabular data from UCI [26] (HAR, letterrecognition, htru2, Mice, TCGA, Pendigits), video data (Weizmann [4], Keck [43]) and image data from UCI [26] (Coil20, Coil100, cmu_faces). A list of all datasets, including their properties, can be found in Appendix B. Furthermore, we also applied SHADE on popular (deep) clustering benchmark datasets whose clusters usually are of Gaussian shape, as Ostdigits [26], USPS [16], MNIST [21], FMNIST [40], and KMNIST [8].

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 [36]. We report the ARI and NMI values of non-noise points and our algorithm’s detected noise percentage in Appendix E.

B. Comparison of Competitors on Toy Example

To provide some intuition about the strengths and weaknesses of current deep clustering methods, let us regard our motivational example in Figure 1: The attempt of DEC to force all clusters to Gaussian blobs in 2D space fails as Figure 1(d) demonstrates: The two rings are not well separated, and the clusters are transformed into Gaussian blobs, losing all of its original structure. Combining the centroid-based clustering loss with the reconstruction loss of the AE, IDEC preserves most of the structure of the s-shaped cluster but fails to separate the rings (cf. 1(e)). DCN [42] shows a very similar behavior. Although it fails to separate the two rings, it preserves the overall structure quite well, as seen in Figure 1(f). As seen in Figure 1(g), the DipEncoder [22] completely loses the original structure of the clusters as it tries to achieve an unimodal distribution within the embedded clusters. However, this strategy is incompatible with circular or s-shaped clusters, as these always have at least two modes regardless of the perspective. DipDECK [23] can also not separate the rings and also splits up one ring, as seen in Figure 1(h). DDC [30] clusters on the TSNE-embedding of an already trained AE embedding. Figure 1(i) shows that even this two-stage process fails to separate the two rings. However, it mostly preserves the density structure of the clusters.

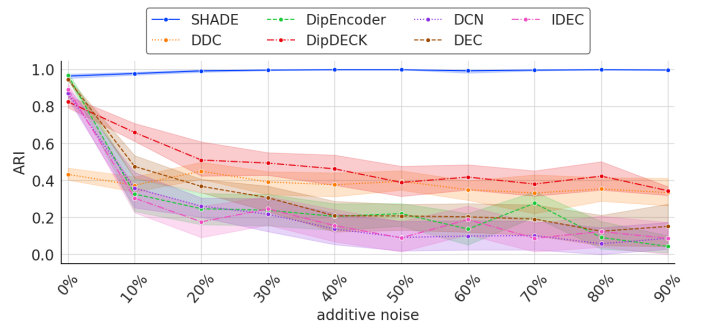


Fig. 4: ARI on synthetic data with varying noise ratio and 100 dimensions. SHADE consistently surpasses all competitors. Note that SHADE reliably returns the highest ARI values, whereas our competitors have a high variance in clustering quality across ten runs. This shows the importance of inherent noise handling for deep clustering algorithms.

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

Dataset	Metric	$\mu = 3$	$\mu = 4$	$\mu = 5$	$\mu = 6$	$\mu = 7$
Synth_high	ARI	97.5 \pm 1.1	98.6 \pm 1.1	98.4 \pm 1.2	98.3 \pm 0.9	98.0 \pm 1.5
	NMI	97.0 \pm 1.1	98.2 \pm 1.2	98.1 \pm 1.2	98.1 \pm 0.7	97.4 \pm 1.7
	noise	3.1 \pm 1.4	2.8 \pm 0.9	2.4 \pm 1.2	3.1 \pm 0.9	3.2 \pm 0.9
	k	14.6 \pm 1.7	13.0 \pm 1.8	13.0 \pm 1.5	12.9 \pm 0.7	14.4 \pm 2.7
TCGA	ARI	79.5 \pm 11.5	84.1 \pm 8.2	86.6 \pm 7.8	85.2 \pm 6.7	84.7 \pm 6.6
	NMI	86.8 \pm 6.2	89.2 \pm 5.5	90.8 \pm 4.4	89.3 \pm 5.2	89.6 \pm 4.0
	noise	20.6 \pm 8.8	22.1 \pm 7.7	21.8 \pm 9.0	21.6 \pm 2.8	23.5 \pm 8.4
	k	5.8 \pm 1.7	6.2 \pm 1.2	5.7 \pm 0.9	5.8 \pm 1.0	6.1 \pm 0.7
COIL20	ARI	83.6 \pm 4.8	81.3 \pm 8.4	82.5 \pm 4.5	84.5 \pm 3.1	83.4 \pm 4.2
	NMI	94.2 \pm 1.7	93.3 \pm 2.0	93.6 \pm 1.7	94.2 \pm 1.1	93.9 \pm 1.0
	noise	11.4 \pm 2.2	10.9 \pm 2.2	12.5 \pm 1.9	14.2 \pm 1.8	14.8 \pm 3.4
	k	16.8 \pm 0.6	16.6 \pm 0.8	16.5 \pm 1.0	16.9 \pm 0.8	17.3 \pm 0.8

C. Systematic Evaluation regarding Noise

We systematically evaluate our algorithm SHADE against the competitor methods w.r.t. varying levels of *noise*. We generated multiple synthetic datasets with 5,000 data points and 100 dimensions with the data generator DENSIREN [18]. We let the data generator add additive, uniform noise up to 90% and investigated the clustering performance on the non-noise points. As shown in Figure 4, SHADE is the only deep clustering algorithm that can still cluster the non-noise points correctly. All the other methods are affected by the additional noise points, and their cluster performance drops drastically.

D. 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 . For choosing μ , there are several heuristics, e.g., $\mu > d$ is often advised, and one of the most recent and widespread heuristics [33] is $\mu = 2d - 1$. Note that this is aimed at datasets with significantly more points than dimensions, which can no longer be considered the default. The parameter μ is mainly needed for robustness and countering the single link effect that merges different clusters because of chains of noise in between them [15]. However, already in the original DBSCAN paper [9], it is set to $\mu = 4$ for all experiments, as it does not significantly change the results. This aligns with established [39] and recent research [33].

We investigated SHADE’s sensitivity to the parameter μ and show that the clustering quality, noise detection, and the

TABLE III: Individual performance and the average score of SHADE on some datasets 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}
Synth_low	ARI	99.7 \pm 0.5	86.6 \pm 0.4	97.1 \pm 3.3
	NMI	99.6 \pm 0.4	90.4 \pm 0.1	98.2 \pm 1.5
	DCSI	97.5 \pm 0.5	98.8 \pm 0.2	97.1 \pm 0.5
HAR	ARI	37.8 \pm 7.2	33.0 \pm 4.8	33.2 \pm 0.0
	NMI	58.8 \pm 5.7	52.1 \pm 2.8	55.6 \pm 0.1
	DCSI	48.3 \pm 5.2	23.7 \pm 2.5	43.8 \pm 7.9
letterrec.	ARI	23.0 \pm 2.0	23.3 \pm 1.2	23.0 \pm 1.2
	NMI	57.6 \pm 0.7	58.1 \pm 0.6	56.1 \pm 0.6
	DCSI	23.0 \pm 1.2	19.4 \pm 1.7	22.0 \pm 1.6
TCGA	ARI	82.3 \pm 9.9	53.7 \pm 13.4	82.9 \pm 6.0
	NMI	88.8 \pm 5.3	69.5 \pm 11.6	88.5 \pm 3.1
	DCSI	77.3 \pm 3.7	60.5 \pm 1.3	66.0 \pm 3.2
Average	ARI	60.7 \pm 4.9	59.0 \pm 2.6	49.2 \pm 4.9
	NMI	76.2 \pm 3.0	74.6 \pm 1.3	67.5 \pm 3.8
	DCSI	61.5 \pm 2.7	57.2 \pm 3.3	50.6 \pm 1.4

estimation of the number of clusters k do not vary much for different values of μ . We present the results of our algorithm for $\mu \in [3, 4, 5, 6, 7]$, demonstrated on three of the benchmark datasets in Table I. We note that for very small datasets, like TCGA with only 801 points, the choice of μ can cause slightly varying clustering results. However, for the typical deep learning setting, clustering quality is robust w.r.t. μ . Thus, we set a low value for $\mu = 5$ for *all experiments*, as this is enough to avoid single links, and similar values are often recommended in the literature [9], [33].

2) *Varying Batch Size*: The batch size in the training phase of our algorithm is more important than in other deep clustering algorithms. By using batches, we only approximately optimize our original loss function stated in Equation (1). Hence, we conducted an ablation study on different batch sizes. As shown in Table II, our algorithm is stable over a wide range of different batch sizes. Sometimes, a larger batch size can improve performance, as seen on the TCGA dataset, but generally, the performance is similar for any batch size that is not too small (e.g., ≥ 200 for Synth_high or ≥ 300 for COIL20). Hence, we choose a batch size of 500 in the comparison experiment with our competitors.

3) *Different Parts of our Loss Function*: We performed an ablation study regarding the effect of the different parts of our

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

Dataset	Metric	100	200	300	400	500	600	700	800	900
Synth_high	ARI	96.8 \pm 1.3	98.4 \pm 1.2	98.5 \pm 0.6	98.3 \pm 1.0	98.4 \pm 0.6	98.1 \pm 1.3	97.9 \pm 1.1	98.3 \pm 1.1	97.7 \pm 1.2
	NMI	95.3 \pm 2.1	98.3 \pm 1.2	98.0 \pm 0.7	97.9 \pm 1.0	97.9 \pm 0.6	97.8 \pm 1.3	97.4 \pm 1.3	97.8 \pm 1.4	97.1 \pm 1.4
	noise	4.3 \pm 2.0	2.6 \pm 0.9	2.8 \pm 1.4	3.5 \pm 1.1	3.0 \pm 1.7	3.0 \pm 1.2	4.1 \pm 1.4	3.8 \pm 1.7	3.1 \pm 1.1
	k	20.6 \pm 5.1	12.6 \pm 2.2	13.7 \pm 1.4	13.4 \pm 1.6	13.7 \pm 1.5	13.1 \pm 1.6	14.1 \pm 2.5	13.7 \pm 2.2	14.4 \pm 1.8
TCGA	ARI	86.7 \pm 7.2	84.5 \pm 4.9	82.8 \pm 6.0	83.5 \pm 7.9	85.8 \pm 6.2	84.8 \pm 9.7	88.2 \pm 4.6	90.2 \pm 7.0	90.0 \pm 4.2
	NMI	91.3 \pm 4.3	89.3 \pm 3.4	87.6 \pm 4.6	88.7 \pm 5.3	89.5 \pm 3.8	89.0 \pm 6.3	91.6 \pm 3.2	93.5 \pm 4.0	93.5 \pm 2.5
	noise	22.3 \pm 5.0	28.2 \pm 4.5	23.3 \pm 9.2	27.6 \pm 6.6	27.3 \pm 5.7	25.8 \pm 9.4	26.2 \pm 7.9	24.5 \pm 7.0	22.1 \pm 5.1
	k	6.4 \pm 0.9	6.1 \pm 1.4	5.7 \pm 1.0	6.2 \pm 1.4	6.0 \pm 1.0	6.1 \pm 0.9	5.8 \pm 0.6	5.2 \pm 1.0	6.3 \pm 0.8
COIL20	ARI	77.6 \pm 8.4	79.6 \pm 7.8	84.1 \pm 5.0	84.3 \pm 3.3	85.2 \pm 1.9	82.4 \pm 4.2	82.7 \pm 4.3	84.2 \pm 3.2	83.2 \pm 4.8
	NMI	91.3 \pm 2.6	92.6 \pm 2.1	94.0 \pm 1.6	94.3 \pm 1.1	94.3 \pm 1.0	93.6 \pm 1.2	93.7 \pm 1.4	94.0 \pm 1.4	93.6 \pm 1.7
	noise	14.0 \pm 3.4	15.0 \pm 3.9	15.7 \pm 2.1	13.1 \pm 2.6	13.6 \pm 2.0	13.2 \pm 2.5	14.8 \pm 1.9	13.9 \pm 1.3	13.0 \pm 2.2
	k	17.9 \pm 2.1	17.4 \pm 1.3	17.9 \pm 1.6	17.1 \pm 1.2	16.9 \pm 0.7	16.5 \pm 1.2	16.7 \pm 1.6	16.7 \pm 0.6	16.7 \pm 0.9

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-E and explain the performance for HAR in more detail in Section III-F and Figure 5.

	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	[18]
	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	[26]
	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	
Video	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	
	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]
Image	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	[43]
	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	[26]
	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	

loss function. For this, we apply the clustering step explained in Section II-E in three different embeddings resulting from different loss functions: the loss \mathcal{L}_D combined with \mathcal{L}_{rec} , only the \mathcal{L}_D loss, and only the \mathcal{L}_{rec} loss. Table III shows the results on a selection of datasets and the average results of them. The individual results on each dataset are provided in Appendix C. On average, the combination of the \mathcal{L}_D and \mathcal{L}_{rec} loss performs best, followed by using only our novel \mathcal{L}_D loss. This holds true for all three evaluation measures: ARI and NMI indicate the correspondence to ground truth clusters. DCSI captures the separation between ground truth clusters and the connectedness within ground truth clusters in the respective embedding. A high DCSI indicates that the ground truth clusters can be found in the embedding using a density-based clustering approach.

E. Performance Benchmark

Our benchmark study results are shown in Table IV, where we used the competitors and datasets described in Section III-A. For most datasets, SHADE has the best or second-best quality.

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. On that note, we want to emphasize that we tested data that *potentially* contains density-connected structures – frequently used benchmark datasets like MNIST do not contain *density-connected* clusters, but spherical-shaped ones and are thus not listed here. To find such Gaussian clusters, we recommend using existing centroid-based deep clustering algorithms. Nevertheless, we provide the results of our algorithm on various MNIST versions in Appendix F.

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. Not surprisingly, for both datasets, the runner-up

is DDC – our only competitor including a concept of density. We provide further information like NMI values and the detected number of clusters in Appendix E.

F. Limitations

We showed superior performance on data containing density-connected structures in the high-dimensional space. However, like any clustering algorithm, SHADE is not perfect in all cases. An illustrative example is the Human Action Recognition (HAR) dataset. Our competitors performed consistently better than SHADE, especially for the methods where the number of clusters is given. However, the confusion matrix in Figure 5 and a brief analysis of the class labels reveal the reason behind this behavior: the dataset contains motion data for six different activities. However, three of them are activities where people move, and the others are resting activities. These classes are density-connected, as the transition between the activities *walk*, *walk upstairs*, and *walk downstairs* are smooth. SHADE, thus, almost perfectly differentiates between moving and not moving activities, which might not yield competitive ARI values but is a reasonable clustering nevertheless.

Not moving	1	1	8	1777	1906	1944
Moving	1721	1543	1398	0	0	0
	walk	walk up	walk down	sitting	standing	laying

Fig. 5: HAR Confusion Matrix. x-axis gives ground truth classes, y-axis clusters detected by SHADE

IV. DISCUSSION AND RELATED WORK

In the following, we provide some background on methods that detect density-based clusters and on current existing deep clustering methods. Furthermore, we discuss the existing approach that applies density-based clustering on the resulting representation of applying several dimensional reduction techniques sequentially.

A. Density-based Clustering

We distinguish two different types of density-based clustering methods: 1) Those that define a cluster as one dense area that has a certain density, i.e., points per volume (e.g., Density Peaks [31]). 2) Those that rely on the concept of *density-connectivity* as known from DBSCAN-like [9] methods. This well-established concept, first introduced in the 70s [13], [39], serves as the basis for SHADE.

Density-based algorithms 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, e.g., for pioneering research in science. In these cases, the expected results or properties of the data are usually not known beforehand. Where centroid-based clustering methods are often based on strong assumptions about the data, like assuming underlying Gaussian distributions (e.g., the EM-algorithm [28]), 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 inherently slow as they are based on complex graph-based computations (even if they are only implicitly used). The *curse of dimensionality* [17] is another issue that makes finding density-based structures in high-dimensional space hard, as the pairwise distances between points become increasingly similar. Thus, the inter- and intra-cluster distances also become harder to distinguish. The *empty space problem* [35] consolidates with increasing dimensionality, making dense areas in the data space more and more unlikely. Thus, finding density-connected structures is, in general, no trivial task, and even though density-based clustering is an established concept, it is still highly discussed, and the research community constantly advances its knowledge [3], [33].

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 [19] or convolutional [20] 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 [41] and IDEC [12] or hard assignments, like DCN [42], DKM [10] and ACe/DeC [27] for centroid-based deep clustering. The centroid-based objective pulls clusters together such that they can collapse onto their cluster centroid. While this can be beneficial for clustering accuracy, it completely destroys the

density-based structure of a dataset. Another problem with these approaches is that they assume that distances within the clusters are smaller than distances between clusters - an assumption that is not true for density-connected clusters. Other methods attempt to address this problem. For example, the DipEncoder [22] considers modalities instead of distances. Therefore, each cluster can obtain an individual spread, leading to more diverse patterns within the embedding. Despite the increased flexibility, it can still only identify convex cluster shapes. DipDECK [23] 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. Furthermore, this strategy allows them to estimate the number of clusters.

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. This stands in strong contrast to our method and the deep clustering methods described above, which optimize the clustering and representation simultaneously with a unified objective. An example of a sequential deep density-based clustering method is DDC (Deep Density-based image Clustering) [30]. It performs three steps. First, an autoencoder is pretrained to obtain a lower-dimensional representation. Second, t-SNE [37] is employed to further reduce the dimensionality. Third, a variant of Density-Peak Clustering (DPC) [31] is employed to obtain cluster assignments from the doubly reduced representation. This process allows DDC to exploit the advantages of both transformations, but at the same time, it inherits the disadvantages. If the AE learns a poor initial embedding, undesired structures are reinforced by t-SNE, resulting in an insufficient final representation. In addition, outliers can strongly distort the result of t-SNE. As DDC follows a sequential, three-step approach, it neglects the integration of density-based clustering and deep learning under a common objective and, therefore, can not improve from a bad representation. Note that by applying DPC as a last step, found clusters are *density-based*, but this does not imply that the algorithm finds *density-connected* structures (see Section IV-A): e.g., DDC does not find elongated density-connected structures like in our motivational example in Figure 1. Nevertheless, DDC achieves an overall strong cluster performance, even when used without augmentations. Thus, we use it as a comparison method in our experiments in Section III.

D. Notions of Stability

Of the various definitions for the 'stability' of a cluster, most are complex, arbitrary, or come with other severe disadvantages. They often rely on sampling or generating perturbed versions of the dataset, which is then clustered and compared to other clustering results [38]. These approaches are computationally expensive, as several clusterings have to be computed, and are

usually not deterministic. E.g., [2] defines a clustering as stable if all clusterings that yield a similarly good solution w.r.t. the optimization function are similar to it. [34] rely on random splits of the data and compare clusterings on the parts with those on the full dataset. This is computationally complex as several partitionings need to be computed and compared. HDBSCAN [7] defines stability based on an Euclidean MST defining the 'death' and 'birth' of clusters. While our definitions have a similar intuition, note that our MST is based on the mutual reachability distance, as it is a closer fit to the original concepts of density-connectivity as given by DBSCAN^(*) [3], [9].

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. The embedding furthermore enables SHADE to find arbitrarily shaped clusters and even topologically intertwined structures while detecting the number of clusters fully automatically. We performed extensive experiments on various datasets, showing SHADE's superiority on various datasets and noisy data.

ACKNOWLEDGEMENTS

We gratefully acknowledge financial support from the Vienna Science and Technology Fund (WWTF-ICT19-041) and the Austrian funding agency for business-oriented research, development, and innovation (FFG-903641⁴).

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] M.-F. Balcan, A. Blum, and A. Gupta, "Clustering under approximation stability," *Journal of the ACM (JACM)*, vol. 60, no. 2, pp. 1–34, 2013.
- [3] 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.
- [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, P. Kröger, J. Sander, and A. Zimek, "Density-based clustering," *WIREs Data Mining Knowl. Discov.*, vol. 10, no. 2, 2020. [Online]. Available: <https://doi.org/10.1002/widm.1343>
- [6] 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.
- [7] R. J. G. B. Campello, D. Moulavi, A. Zimek, and J. Sander, "Hierarchical density estimates for data clustering, visualization, and outlier detection," *ACM Trans. Knowl. Discov. Data*, vol. 10, no. 1, pp. 5:1–5:51, 2015.
- [8] T. Clauwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha, "Deep learning for classical japanese literature," *arXiv preprint arXiv:1812.01718*, 2018.
- [9] 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.
- [10] M. M. Fard, T. Thonet, and É. Gaussier, "Deep k -means: Jointly clustering with k -means and learning representations," *Pattern Recognit. Lett.*, vol. 138, pp. 185–192, 2020.
- [11] J. Gauss, F. Scheipl, and M. Herrmann, "Dcsi—an improved measure of cluster separability based on separation and connectedness," *arXiv preprint arXiv:2310.12806*, 2023.
- [12] 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.
- [13] J. A. Hartigan, *Clustering algorithms*. John Wiley & Sons, Inc., 1975.
- [14] A. Hatcher, *Algebraic topology*. Cambridge: Cambridge University Press, 2002.
- [15] J. Held, A. Beer, and T. Seidl, "Chain-detection for DBSCAN," in *BTW (Workshops)*, ser. LNI, vol. P-290. Gesellschaft für Informatik, Bonn, 2019, pp. 173–183.
- [16] 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.
- [17] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, 1998, pp. 604–613.
- [18] 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.
- [19] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AICHE journal*, vol. 37, no. 2, pp. 233–243, 1991.
- [20] 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.
- [21] 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.
- [22] 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.
- [23] 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.
- [24] C. Leiber, L. Miklautz, C. Plant, and C. Böhm, "Benchmarking deep clustering algorithms with clustpy," in *ICDM (Workshops)*. IEEE, 2023, pp. 625–632.
- [25] S. T. Mai, I. Assent, and M. Storgaard, "Anydbc: An efficient anytime density-based clustering algorithm for very large complex datasets," in *KDD*. ACM, 2016, pp. 1025–1034.
- [26] K. N. Markelle Kelly, Rachel Longjohn, "The uci machine learning repository," 2023. [Online]. Available: <http://archive.ics.uci.edu/ml>
- [27] L. Miklautz, L. G. M. Bauer, D. Mautz, S. Tschitschek, C. Böhm, and C. Plant, "Details (don't) matter: Isolating cluster information in deep embedded spaces," in *IJCAI*. ijcai.org, 2021, pp. 2826–2832.
- [28] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal processing magazine*, vol. 13, no. 6, pp. 47–60, 1996.
- [29] F. Murtagh, "A survey of recent advances in hierarchical clustering algorithms," *The computer journal*, vol. 26, no. 4, pp. 354–359, 1983.
- [30] Y. Ren, N. Wang, M. Li, and Z. Xu, "Deep density-based image clustering," *Knowl. Based Syst.*, vol. 197, p. 105841, 2020.
- [31] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *science*, vol. 344, no. 6191, pp. 1492–1496, 2014.
- [32] E. Schubert, S. Hess, and K. Morik, "The relationship of dbscan to matrix factorization and spectral clustering," in *LWDA*, 2018, pp. 330–334.
- [33] 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.
- [34] S. P. Smith and R. Dubes, "Stability of a hierarchical clustering," *Pattern Recognition*, vol. 12, no. 3, pp. 177–187, 1980.
- [35] M. Steinbach, L. Ertöz, and V. Kumar, "The challenges of clustering high dimensional data," in *New directions in statistical physics: econophysics, bioinformatics, and pattern recognition*. Springer, 2004, pp. 273–309.
- [36] 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.

⁴<https://projekte.ffg.at/projekt/4814676>

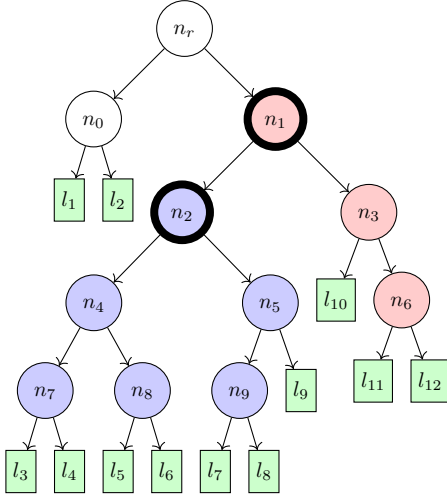
- [37] L. Van der Maaten and G. Hinton, “Visualizing data using t-sne.” *Journal of machine learning research*, vol. 9, no. 11, 2008.
- [38] U. Von Luxburg *et al.*, “Clustering stability: an overview,” *Foundations and Trends® in Machine Learning*, vol. 2, no. 3, pp. 235–274, 2010.
- [39] D. Wishart, “Numerical classification method for deriving natural classes,” *Nature*, vol. 221, no. 5175, pp. 97–98, 1969.
- [40] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms,” *arXiv preprint arXiv:1708.07747*, 2017.
- [41] 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.
- [42] 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.
- [43] 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.

APPENDIX A

DENSITY-CONNECTIVITY TREE AND STRUCTURE TREE

Figure 6 shows a small example of a density-connectivity tree \mathcal{T}_d for $\mu = 3$. The nodes with bold borders are nodes that exist in the structure tree, too, as both their children have more than $\mu = 3$ leaves. Colors imply the smallest cluster to which a node belongs in the structure tree.

Fig. 6: Tree capturing density-connectivity. Leaves correspond to data points and nodes to the distances d_{dc} between them.



APPENDIX B

OVERVIEW OF DATASET PROPERTIES FOR EXPERIMENTS

Table V summarizes all datasets that we used, their number of points n , dimensionality d , number of classes k , number of noise points $\#noise$, and their source. The noise in Synth_low and Synth_high was sampled from a uniform distribution, where noise has to show a certain distance to the clusters.

TABLE V: Dataset properties

	Dataset	n	d	k	$\#noise$	Source
Tabular data	Synth_low	5000	100	10	500	[18]
	Synth_high	5000	100	10	500	[18]
	HAR	10,299	561	6	0	[26]
	letterrecognition	20,000	16	26	0	[26]
	htru2	17,898	8	2	0	[26]
	Mice	1,077	68	8	0	[26]
	TCGA	801	20,264	5	0	[26]
Video	Pendigits	10,992	16	10	0	[26]
	Weizmann	5,701	77,760	90	0	[4]
Image data	Keck	25,457	120,000	60	0	[43]
	COIL20	1,440	16,384	20	0	[26]
	COIL100	7,200	49,152	100	0	[26]
	cmu_faces	624	960	20	0	[26]
	Optdigits	5,620	64	10	0	[26]
	USPS	9,298	256	10	0	[16]
	MNIST	70,000	784	10	0	[21]
	FMNIST	70,000	784	10	0	[40]
	KMNIST	70,000	784	10	0	[8]

TABLE VI: Parameters for the Synth data set

Parameters	Values
n	5000
dim	100
ratio_noise	[0.0, 0.1, ..., 0.9]
max_retry	5
dens_factors	[1, 1, 0.5, 0.3, 2, 1.2, 0.9, 0.6, 1.4, 1.1]
square	True
clunum	10
seed	6
core_num	200
momentum	0.8
step	1.5
branch	0.1
star	1
verbose	False
safety	False
domain_size	20
random_start	False

APPENDIX C
ABLATION – LOSS FUNCTION

In Table VII, the performance of using the individual loss terms and both together are reported for all datasets. This is an extension of Table III.

TABLE VII: Extensive ablation study regarding the loss function, including more datasets and more evaluation measures than in the main part.

Dataset	Metric	$\mathcal{L}_D + \mathcal{L}_{rec}$	\mathcal{L}_{rec}	\mathcal{L}_D
Synth_low	ARI	99.7 ± 0.5	86.6 ± 0.4	97.1 ± 3.3
	NMI	99.6 ± 0.4	90.4 ± 0.1	98.2 ± 1.5
	DCSI	97.5 ± 0.5	98.8 ± 0.2	97.1 ± 0.5
Synth_high	ARI	97.3 ± 1.5	85.8 ± 0.2	96.4 ± 1.8
	NMI	97.2 ± 1.4	87.9 ± 0.0	96.2 ± 1.8
	DCSI	95.9 ± 0.9	96.8 ± 1.0	93.4 ± 1.2
HAR	ARI	37.8 ± 7.2	33.0 ± 4.8	33.2 ± 0.0
	NMI	58.8 ± 5.7	52.1 ± 2.8	55.6 ± 0.1
	DCSI	48.3 ± 5.2	23.7 ± 2.5	43.8 ± 7.9
letterrec.	ARI	23.0 ± 2.0	23.3 ± 1.2	23.0 ± 1.2
	NMI	57.6 ± 0.7	58.1 ± 0.6	56.1 ± 0.6
	DCSI	23.0 ± 1.2	19.4 ± 1.7	22.0 ± 1.6
htu2	ARI	52.4 ± 33.5	55.2 ± 7.3	49.7 ± 22.9
	NMI	39.0 ± 25.5	38.6 ± 4.0	33.4 ± 15.8
	DCSI	0.4 ± 0.1	0.2 ± 0.1	0.1 ± 0.1
Mice	ARI	27.9 ± 4.8	27.3 ± 2.5	23.4 ± 8.4
	NMI	47.7 ± 5.4	51.2 ± 1.4	43.9 ± 14.1
	DCSI	24.9 ± 1.3	25.5 ± 1.7	23.4 ± 1.5
TCGA	ARI	82.3 ± 9.9	53.7 ± 13.4	82.9 ± 6.0
	NMI	88.8 ± 5.3	69.5 ± 11.6	88.5 ± 3.1
	DCSI	77.3 ± 3.7	60.5 ± 1.3	66.0 ± 3.2
Pendigits	ARI	75.6 ± 0.7	76.2 ± 0.6	74.6 ± 1.2
	NMI	83.2 ± 0.6	83.6 ± 0.5	82.2 ± 0.9
	DCSI	52.2 ± 1.3	33.8 ± 0.9	53.4 ± 1.0
Weizmann	ARI	48.7 ± 2.1	49.8 ± 5.6	48.2 ± 5.0
	NMI	79.9 ± 0.9	81.4 ± 0.8	80.3 ± 1.1
	DCSI	72.5 ± 2.3	77.0 ± 2.3	72.8 ± 3.6
Keck	ARI	7.2 ± 0.4	7.0 ± 0.2	7.5 ± 0.4
	NMI	60.6 ± 0.5	62.2 ± 0.2	61.1 ± 0.5
	DCSI	26.9 ± 2.2	25.6 ± 1.1	27.1 ± 3.3
COIL20	ARI	70.2 ± 4.7	74.1 ± 1.9	69.6 ± 4.2
	NMI	86.0 ± 2.0	89.2 ± 0.4	84.9 ± 2.4
	DCSI	80.6 ± 3.5	86.7 ± 1.7	77.6 ± 2.1
COIL100	ARI	58.0 ± 3.9	76.5 ± 1.4	56.7 ± 2.4
	NMI	85.9 ± 1.2	92.0 ± 0.4	85.5 ± 0.6
	DCSI	79.5 ± 2.4	89.6 ± 0.8	78.7 ± 2.1
cmu_faces	ARI	37.4 ± 4.8	41.2 ± 13.3	40.3 ± 4.2
	NMI	67.6 ± 3.6	73.7 ± 10.5	70.7 ± 2.7
	DCSI	45.0 ± 3.0	56.3 ± 1.1	48.6 ± 2.5

APPENDIX D
REAL-WORLD VIDEODATA

We evaluate our approach on two video datasets. The first dataset is the Weizmann video dataset [4]. This dataset consists of 93 videos showing nine persons performing ten different activities like walking or jumping. This results in 90 clusters, one for each combination of person and activity. We extract 5,701 $144 \times 180 \times 3$ color frames from the videos and interpret each frame as an individual sample. The second dataset is the Keck Gesture video dataset [43]. It contains 98 videos of four persons performing 14 different gestures. As there are pauses between the gestures, we add ‘no gesture’ as a separate label, resulting in 60 clusters. We extract 25,457 $480 \times 640 \times 3$ color frames from the videos, which are then downsized to $200 \times 200 \times 3$ due to space limitations. Each frame is again interpreted as an individual sample.

APPENDIX E
EXTENDED TABLE

In addition to the ARI scores, as stated in Table IV, we evaluate the algorithms using NMI and show the identified number of clusters. These results are given in Table VIII.

TABLE VIII: Overview of clustering results. As SHADE is the only algorithm handling noise, we report its detected noise ratio in column 1 for clarity (*noise: $xx.x \pm x.x$*). k gives the number of ground truth clusters in the first column and the number of detected clusters in the other columns (SHADE - DipDECK); otherwise, algorithms have received the ground truth number of classes as an input parameter (-). We do not include evaluation measures for SHADE in the color coding as these values are computed on labelings including noise, which could lead to an overoptimistic assessment of our method. Note, however, that the clustering quality that SHADE reaches for non-noise points (column 'SHADE') is of exceptionally good quality for most datasets. (*DipEncoder is given the number of clusters, but can 'lose' them, leading to a lower k)

Dataset	Metric	SHADE	SHADE_1nn	DDC	DipDECK	DipEncoder	DCN	DEC	IDEC
Tabular data	Synth_low (noise: 1.1 ± 1.3) $k = 10$	ARI NMI k	99.4 ± 1.8 99.7 ± 1.0 10.0 ± 0.0	98.9 ± 2.0 99.2 ± 1.2 10.0 ± 0.0	56.9 ± 5.5 82.9 ± 3.2 16.8 ± 1.0	33.9 ± 6.4 58.0 ± 5.0 4.1 ± 0.5	10.1 ± 9.9 31.3 ± 13.6 -	9.6 ± 9.7 29.4 ± 13.4 -	40.2 ± 3.0 69.4 ± 3.0 44.3 ± 8.8
	Synth_high (noise: 2.4 ± 1.2) $k = 10$	ARI NMI k	98.4 ± 1.2 98.1 ± 1.2 13.0 ± 1.5	97.5 ± 1.4 97.3 ± 1.3 13.0 ± 1.5	33.9 ± 11.1 61.8 ± 6.1 10.3 ± 2.1	29.9 ± 13.6 53.2 ± 12.0 4.7 ± 1.3	9.3 ± 10.7 28.9 ± 13.4 -	8.8 ± 10.5 26.9 ± 13.2 -	30.3 ± 3.5 61.4 ± 4.8 46.7 ± 5.0
	HAR (noise: 3.2 ± 5.2) $k = 6$	ARI NMI k	36.0 ± 5.6 58.5 ± 5.8 3.3 ± 2.0	36.4 ± 6.4 58.2 ± 5.5 3.3 ± 2.0	49.4 ± 3.3 68.2 ± 2.1 4.2 ± 1.0	51.3 ± 4.0 71.3 ± 2.0 3.1 ± 0.3	60.0 ± 6.9 73.6 ± 4.1 -	66.1 ± 1.3 75.4 ± 1.2 -	63.4 ± 2.4 75.0 ± 1.8 64.9 ± 0.9
	letterrec. (noise: 50.3 ± 1.8) $k = 26$	ARI NMI k	43.2 ± 1.7 75.6 ± 1.0 111.4 ± 4.9	23.0 ± 0.9 57.4 ± 0.5 111.4 ± 4.9	9.9 ± 2.9 43.5 ± 2.5 14.0 ± 1.0	7.3 ± 3.5 34.4 ± 3.8 13.0 ± 2.4	24.7 ± 1.3 49.7 ± 0.9 -	22.6 ± 1.0 46.4 ± 1.0 -	23.9 ± 1.6 50.8 ± 1.0 25.2 ± 1.8
	htru2 (noise: 14.0 ± 3.8) $k = 2$	ARI NMI k	72.8 ± 23.4 59.9 ± 19.6 5.9 ± 2.1	65.0 ± 19.5 47.4 ± 14.4 5.9 ± 2.1	49.4 ± 13.0 42.1 ± 7.0 3.7 ± 0.6	9.7 ± 19.4 5.5 ± 11.0 1.2 ± 0.4	4.3 ± 0.8 10.5 ± 1.4 -	49.7 ± 2.8 31.6 ± 7.0 -	3.0 ± 0.5 10.8 ± 0.6 3.2 ± 0.6
	Mice (noise: 30.0 ± 5.5) $k = 8$	ARI NMI k	32.5 ± 3.8 56.2 ± 4.6 11.9 ± 2.8	27.7 ± 2.9 48.7 ± 2.3 11.9 ± 2.8	25.2 ± 1.9 49.6 ± 2.0 15.3 ± 1.2	22.7 ± 4.3 48.0 ± 5.3 13.4 ± 5.3	21.6 ± 2.6 38.7 ± 2.6 -	21.7 ± 1.4 38.3 ± 1.5 -	22.0 ± 1.5 39.4 ± 1.8 21.8 ± 1.4
	TCGA (noise: 21.8 ± 9.0) $k = 5$	ARI NMI k	86.6 ± 7.8 90.8 ± 4.4 5.7 ± 0.9	80.0 ± 13.7 87.4 ± 7.2 5.7 ± 0.9	87.5 ± 0.8 91.9 ± 0.7 6.0 ± 0.0	88.8 ± 4.4 91.9 ± 2.1 5.9 ± 0.5	93.4 ± 6.0 94.0 ± 3.8 -	87.2 ± 5.3 89.1 ± 3.1 -	85.1 ± 2.7 89.5 ± 1.2 82.6 ± 0.9
	Pendigits (noise: 17.5 ± 2.2) $k = 10$	ARI NMI k	85.1 ± 1.4 89.3 ± 0.9 20.0 ± 1.8	75.1 ± 0.8 82.7 ± 0.6 20.0 ± 1.8	76.9 ± 2.0 84.1 ± 1.0 13.0 ± 0.4	74.3 ± 1.1 82.0 ± 0.8 14.8 ± 0.7	64.6 ± 3.0 75.2 ± 1.3 -	61.6 ± 1.9 72.7 ± 0.6 -	65.7 ± 3.3 76.7 ± 1.4 64.9 ± 2.6
	Weizmann (noise: 14.6 ± 2.6) $k = 90$	ARI NMI k	57.1 ± 5.9 86.6 ± 0.6 57.1 ± 2.1	48.2 ± 3.6 80.2 ± 1.1 57.1 ± 2.1	14.7 ± 1.8 57.5 ± 2.0 20.8 ± 2.4	12.0 ± 1.9 52.8 ± 1.9 24.5 ± 2.2	23.3 ± 1.2 61.9 ± 0.8 $88.4 \pm 1.2^*$	24.6 ± 1.1 62.3 ± 0.9 -	24.9 ± 1.2 63.7 ± 1.0 24.7 ± 1.2
	Keck (noise: 25.4 ± 1.1) $k = 60$	ARI NMI k	9.3 ± 0.5 66.3 ± 0.3 226.6 ± 10.4	7.5 ± 0.4 61.6 ± 0.3 226.6 ± 10.4	-0.2 ± 1.1 18.1 ± 4.3 10.6 ± 3.1	6.9 ± 0.8 34.9 ± 6.0 32.3 ± 7.4	7.1 ± 0.3 42.4 ± 0.6 -	6.4 ± 0.5 39.4 ± 2.5 -	6.1 ± 0.9 39.9 ± 3.5 6.2 ± 0.9
	COIL20 (noise: 12.5 ± 1.9) $k = 20$	ARI NMI k	82.5 ± 4.5 93.6 ± 1.7 16.5 ± 1.0	68.7 ± 3.5 85.6 ± 1.3 16.5 ± 1.0	62.0 ± 5.5 85.5 ± 0.9 14.3 ± 0.8	50.5 ± 7.8 79.9 ± 2.4 18.9 ± 1.0	64.0 ± 3.0 80.2 ± 1.1 -	62.4 ± 2.8 79.6 ± 1.3 -	63.7 ± 2.8 80.6 ± 1.0 62.9 ± 2.9
Image data	COIL100 (noise: 24.9 ± 1.7) $k = 100$	ARI NMI k	78.1 ± 7.3 94.4 ± 0.9 77.8 ± 3.2	56.8 ± 5.0 85.4 ± 0.6 77.8 ± 3.2	16.4 ± 3.8 69.5 ± 3.0 18.3 ± 3.3	21.4 ± 3.0 69.5 ± 1.7 26.8 ± 1.9	54.3 ± 1.9 82.6 ± 0.6 99.8 ± 0.4	55.9 ± 3.0 84.2 ± 0.7 -	55.8 ± 2.0 85.8 ± 0.5 56.9 ± 2.0
	cmu_faces (noise: 15.1 ± 4.1) $k = 20$	ARI NMI k	38.9 ± 7.6 69.1 ± 5.3 10.6 ± 1.7	34.6 ± 6.2 65.0 ± 4.9 10.6 ± 1.7	35.0 ± 3.5 64.4 ± 2.3 12.0 ± 0.9	29.8 ± 9.8 62.3 ± 7.6 8.9 ± 2.2	37.9 ± 2.2 67.7 ± 1.6 -	40.3 ± 2.0 68.5 ± 1.0 -	35.8 ± 2.8 66.1 ± 2.0 39.4 ± 3.3

APPENDIX F
BENCHMARK DATASETS WITH GAUSSIAN CLUSTERS

In Table IX, we include results on the most common benchmark datasets in the area: Optdigits, Pendigits, USPS, MNIST, FMNIST, and KMNIST. Note that they have in common that their classes are centroid-based: each object of a class is similar to a concept given by the label, e.g., the picture of a written digit. This results in a typical, roughly Gaussian-like distribution where most points of a class are rather similar to the center, and fewer points are far away from the center. While SHADE returns competitive NMI and ARI values on the clustered data points (column 'SHADE'), it detects objects far away from their concept as noise, which can be the majority of the data in the worst case.

TABLE IX: Analogous to Table VIII, we show further clustering results for data sets that do not contain density-based structures, but are, however, often expected as benchmarks for deep clustering. SHADE detects high percentages of these data sets as noise as especially points at the border of Gaussian clusters have often no density-connection to the rest of the cluster.

Dataset	Metric	SHADE	SHADE_1nn	DDC	DipDECK	DipEncoder	DCN	DEC	IDEC
Optdigits (noise: 38.2 ± 5.3) $k = 10$	ARI	93.3 ± 1.9	78.0 ± 7.4	88.9 ± 2.3	82.2 ± 2.3	80.2 ± 4.0	77.0 ± 3.5	80.4 ± 3.1	80.7 ± 3.6
	NMI	94.6 ± 1.0	83.4 ± 3.9	91.7 ± 1.3	86.2 ± 0.9	85.6 ± 2.0	82.9 ± 1.6	<u>86.3 ± 1.4</u>	86.2 ± 1.5
	k	12.4 ± 1.0	12.4 ± 1.0	11.1 ± 0.5	11.0 ± 1.2	-	-	-	-
USPS (noise: 45.7 ± 2.8) $k = 10$	ARI	88.1 ± 2.7	68.1 ± 1.5	92.2 ± 2.2	68.2 ± 5.1	72.5 ± 2.0	66.2 ± 1.3	73.4 ± 0.9	74.0 ± 0.9
	NMI	91.8 ± 1.1	75.9 ± 1.1	91.0 ± 0.8	78.5 ± 2.5	79.9 ± 1.6	74.5 ± 1.0	81.0 ± 0.5	<u>81.3 ± 0.6</u>
	k	9.5 ± 0.9	9.5 ± 0.9	9.8 ± 0.4	8.0 ± 1.2	-	-	-	-
MNIST (noise: 58.9 ± 5.3) $k = 10$	ARI	86.1 ± 8.4	54.1 ± 2.8	94.5 ± 1.2	80.9 ± 3.0	79.6 ± 2.8	82.0 ± 4.5	79.5 ± 2.1	81.9 ± 2.0
	NMI	84.5 ± 2.4	63.7 ± 1.3	93.7 ± 0.8	84.5 ± 1.5	84.9 ± 1.6	84.6 ± 1.9	85.1 ± 1.0	<u>87.5 ± 0.8</u>
	k	38.0 ± 9.9	38.0 ± 9.9	10.0 ± 0.0	11.3 ± 0.8	-	-	-	-
FMNIST (noise: 66.3 ± 2.3) $k = 10$	ARI	72.2 ± 2.7	35.7 ± 1.4	35.5 ± 7.1	46.5 ± 0.9	43.5 ± 3.6	45.8 ± 2.9	42.3 ± 4.1	46.9 ± 3.6
	NMI	73.4 ± 1.0	53.9 ± 0.9	64.1 ± 3.3	65.3 ± 0.7	59.4 ± 2.4	62.4 ± 2.2	59.3 ± 2.7	63.7 ± 2.5
	k	68.7 ± 9.2	68.7 ± 9.2	7.1 ± 0.9	9.6 ± 1.1	-	-	-	-
KMNIST (noise: 73.4 ± 2.2) $k = 10$	ARI	58.6 ± 9.4	27.5 ± 1.7	60.3 ± 3.5	33.8 ± 9.9	43.6 ± 1.1	40.3 ± 1.6	42.4 ± 0.9	42.8 ± 1.1
	NMI	66.2 ± 3.1	46.8 ± 0.6	73.6 ± 1.6	55.9 ± 5.0	<u>56.7 ± 1.3</u>	53.6 ± 1.4	55.7 ± 1.1	<u>56.7 ± 1.2</u>
	k	64.1 ± 10.4	64.1 ± 10.4	16.2 ± 0.7	11.7 ± 2.8	-	-	-	-